



UiT The Arctic University of Norway

Faculty of Science and Technology

Department of Physics and Technology

Deep Generative Models in Credit Scoring

Rogelio Andrade Mancisor

A dissertation for the degree of Philosophiae Doctor – October 2020



Deep Generative Models
in
Credit Scoring

Rogelio Andrade Mancisidor

Abstract

Banks need to develop effective credit scoring models to better understand the relationship between customer information and the customer's ability to repay the loan. The output of such a model is called the default probability and is used to rank loan applications in terms of their creditworthiness. The main focus of this thesis is to develop novel credit scoring methodologies that solve well-known problems in the field and that bridge the gap between simple neural networks and advanced methodologies in deep learning applied to credit scoring.

In the research conducted in this thesis, we propose a new methodology to learn useful data representations of bank customers introducing a supervision stage, where we group the input data using the Weight of Evidence transformation, into the Variational Autoencoder framework. Our proposed method learns data representations that are able to capture the customers' creditworthiness in a well-defined clustering structure. Further, the learned data representations preserve the spatial coherence of customers' creditworthiness and are well suited for marketing campaigns and credit risk assessment.

We develop two novel Deep Generative Models that are able to infer the unknown customers' creditworthiness of rejected loan applications. Our proposed models use probabilistic theory to infer the unknown customers' creditworthiness, which is a clear advantage over traditional approaches. Adding rejected applications improves the classification accuracy of our proposed models, and potentially solves the selection bias problem. We parametrize a Gaussian mixture model with neural networks to further improve the latent representation of customers information.

Finally, we address credit scoring as a multi-modal learning problem. That is, banks have multiple measurement-modalities that provide complementary information about customers. Hence, we develop a novel Deep Generative Model that learns shared data representations, which are useful to generate future credit data and for classification. Our proposed model generates future credit data, based on application data, which can be used to support bank activities other than credit scoring. Finally, we introduce a novel objective function that improves the generative process and classification in our proposed model by maximizing mutual information between future credit data and its shared representation.

Acknowledgments

I would like to express my deepest gratitude to my main supervisor Professor Robert Jenssen. His guidance, support, and valuable insight have played a significant role in conducting this research. Thank you for guiding me through this journey and shaping my research development.

I would also like to extend my sincere thanks to my co-supervisor Dr. Kjersti Aas, who always had time to guide me and look over the challenges we faced in this research work. I really appreciate your involvement and attention to detail in the research work of this thesis.

I am also grateful to Dr. Michael Kampffmeyer, who always had time to discuss any challenges I faced during the research conducted in this thesis. It has been an incredible experience to be part of the UiT Machine Learning Group where I had the opportunity to interact with a group of highly skilled researchers. It was always nice to be there for some days at the institute and I want to thank you all. In particular, I would like to acknowledge the help I received from Sigurd and Thomas who were always available for answering all my technical questions.

I am also grateful to Santander Consumer Bank - Nordics for funding this research project, specially to Per Kolbjørnsen and Andres Diez for supporting this research project. I would also like to thank Dr. Biliana Alexandrova Kabadjova for mentoring me during the time that I spent in the research internship at Banco de México.

I am deeply in debt for the time that I could not spend with you Silje, Elena, and Matheo due to studying, traveling or working with this research project. Your support, patience, and understanding mean a lot to me. Finally, my sincere gratitude to my parents, siblings, family, and friends for your unconditional love and support.

Contents

Abstract	i
Acknowledgments	ii
1 Introduction	1
1.1 Credit Scoring	5
1.2 Challenges in Credit Scoring	6
1.3 Research objectives	8
1.4 Approach adopted	9
1.5 Brief summary of papers	10
1.6 Reading guide	14
I Background Theory and Methodology	16
2 Credit Scoring Models	17
2.1 Linear Discriminant Analysis	19
2.2 Linear Programming	20
2.3 Decision Trees	22
2.4 Logistic Regression	24
2.5 Weight of Evidence	25
3 Probabilistic Graphical Models	27
3.1 Conditional Probability and The Bayes' Theorem	27
3.2 Directed Graphical Models	30
3.3 Variational Inference	32
3.3.1 Mean Field Approximation	37
3.3.2 Stochastic Variational Inference	39
3.3.3 Non-conjugate Variational Inference	41

3.3.4	Amortized Inference	42
4	Deep Generative Models	44
4.1	Variational Autoencoder	44
4.1.1	Connection with autoencoders	46
4.1.2	Generative properties	46
4.2	Deriving the Lower Bound	48
4.3	The Reparameterization Trick	50
4.3.1	Reparameterization Gradients	50
4.3.2	Backpropagate gradients through a deterministic reparameterization	53
4.4	Improving DGMs	54
4.4.1	Tightness of the ELBO	54
4.4.2	Beyond the mean-field assumption	55
4.4.3	The Kullback-Leibler divergence is restrictive	56
4.4.4	Learning expressive latent representations	57
II	Summary of research	59
5	Paper I - Learning Latent Representations of Bank Customers with the Variational Autoencoder	60
5.1	Contributions by the author	61
6	Paper II - Deep Generative Models for Reject Inference in Credit Scoring	63
6.1	Contributions by the author	64
7	Paper III - Generating Customer's Credit Behavior with Deep Generative Models	66
7.1	Contributions by the author	67
8	Concluding remarks	68
8.1	Weaknesses and future work	69
III	Included papers	72
9	Paper I	73

10 Paper II	88
11 Paper III	107
A Multilayer Perceptron Model and the Backpropagation algorithm	128
B Segment based credit scoring	134
B.0.1 Methodology	134
B.0.2 Results	135
B.0.3 Conclusion	137

List of Figures

1.1	Default definition	6
1.2	Challenges in credit scoring	9
1.3	Deep generative models develop in this research work	13
2.1	Two-dimensional linear discriminant function	18
2.2	Linear programming model	21
2.3	Decision trees recursive partitioning	24
3.1	Probabilistic graphical model for credit scoring	31
3.2	Bayesian Gaussian Mixture Model	33
3.3	Variational approximation	36
3.4	Mean field approximation and the Variational Autoencoder	43
4.1	Variational Autoencoder and Autoencoder	47
4.2	Variance of score and reparametrized derivatives	53
4.3	The reparameterization trick	54
5.1	Learned data representation in Paper I	61
6.1	Models developed in Paper II	64
7.1	Multi-modal credit data in Paper III	67
A.1	Multilayer Perceptron MLP	129
A.2	Gradient descent optimization	130
B.1	Segment-based credit scoring	135

Chapter 1

Introduction

Retail banks, as well as other financial institutions, decide whether to grant credit to applicants based on their ability to repay the loan. More than forty years ago, banks' analysts decided whether to grant a loan application based on the four *C*'s of credit (Altman and Saunders, 1997). That is, based on the borrower's *character*, *capital*, *capacity* and *collateral*. Thomas (2000) adds *conditions* as the fifth *C*.

However, over the past decades, the Bank of International Settlements (BIS)¹ has encouraged retail banks to develop internal models to measure credit risk. Therefore, banks have focused on developing effective mathematical models to decide whether to grant credit, increase an existing credit line, and to predict the recovery amount on a given defaulted loan, among others. The research in this thesis focuses on the first type of models, which are called credit scoring models (Thomas, 2000).

According to The Financial Supervisory Authority (FSA) of Norway, the total credit loan losses in the first half of 2019 for all Norwegian banks is 2.1 billion NOK². Further, Khandani et al. (2010) show that by using advance credit scoring techniques it is possible to reduce credit losses by 12%-24%. This means that the Norwegian banks could have potential savings ranging

¹In 1988 the banking supervision authorities agreed upon some rules for banking regulation called The Basel Capital Accord (Basel I for short). Then in 2004 a new accord was published (Basel II for short) with a more sophisticated method for calculating the risk weighted assets and allowing for internal-based ratings models.

²FSA financial report for the 1st. half of 2019. Norwegian only.

from 252 millions NOK to 504 millions NOK by developing and implementing advanced credit scoring models. Therefore, it is important to develop effective mathematical models to grant credit.

The history of credit scoring started back in 1938 in the National Bureau of Economic Research in New York, USA with the work developed by Durand (1941). He used statistical measures to discover which applicants' features are more relevant to quantify risk. Since then, different credit scoring models have been developed and the most popular techniques are discriminant analysis, logistic regression, classification trees, and linear programming (Thomas, 2000). Further, neural networks models have gained popularity in credit scoring and the first model was proposed by Tam and Kiang (1990). They trained neural networks using the backpropagation algorithm (Rumelhart et al., 1988) to classify bank default data. Their initial research was further improved in Tam and Kiang (1992).

The experiments conducted by Tam and Kiang (1992) show that neural networks achieve higher predictive accuracy than discriminant analysis, logistic regression, decision trees, and k -nearest neighbors. In addition, their study provides insight into the potentials of neural networks, e.g. i) neural networks are better approximations of the sample distribution given their nonlinear activation functions, ii) neural networks have the ability to adjust the model, hence they react to changes in the real world, and iii) neural networks do not assume any data distribution. They also named some challenges associated, at that time, with neural networks models, e.g. i) difficult to choose the network architecture, ii) training is computationally demanding, and iii) model interpretability is not straightforward.

Over the past decades the research in credit scoring with neural networks have grown rapidly and different benchmark studies have been published. Specifically, neural networks for credit scoring have been compared with linear discriminant analysis, logistic regression, genetic algorithms, decision trees, k -nearest neighbor, support vector machines, and probit models (Desai et al., 1997; West, 2000; Yobas et al., 2000; Malhotra and Malhotra, 2003; Zekic-Susac et al., 2004; Abdou et al., 2008; Angelini et al., 2008). The empirical results show that neural networks offer, on average, relative high model performance compared to other methods to classify bank default data. Both Baesens et al. (2003) and Lessmann et al. (2015) present detailed comparisons of different machine learning methods for credit scoring.

Inspired by the promising results achieved by neural networks in credit scoring, novel approaches have been developed. Jensen (1992) analyses neural networks for scenarios where the outcome of the loan can take three different values. Continuing in the scenario with multiple outcomes, Desai et al. (1997) use ensemble neural networks models for credit scoring. Lee et al. (2002), Lee and Chen (2005), and Zhang et al. (2018b) introduce more ensemble models using hybrid models that combine discriminant analysis, multivariate adaptive regression splines with neural networks, and multiple simple classifiers, respectively. Another example of ensemble modeling is presented in Hsieh (2005), where self-organizing maps, k-means, and neural networks are combined into a unified framework, or in Tsai and Wu (2008); Munkhdalai et al. (2020) where multiple neural networks are assembled. Both Lai et al. (2006) and Shen et al. (2019) propose methods to deal with limited training data in credit scoring by using a neural network metamodel and an ensemble approach using synthetic variables, respectively. Another type of ensemble models are introduced in Chuang and Huang (2011), where they combine neural networks and case-based reasoning in a two-step approach or in Pławiak et al. (2020) where the authors combine probabilistic neural networks with data normalization methods, feature extraction techniques, and kernel functions in a unified framework.

Some research has focused on identifying optimal network architectures and optimal data proportions for training, validation and test (Khashman, 2010; Zhao et al., 2015). Other research analyzes the performance of neural networks using alternative data for credit scoring, e.g. microfinance (Blanco et al., 2013; Byanjankar et al., 2015) and accounting data (Šuštersič et al., 2009). Abdou et al. (2019) use data for the Indian banking sector and actual misclassification costs to measure the performance of neural networks and traditional approaches for credit scoring. Neural networks have also been used in a different context than credit scoring, for example Mbuva et al. (2019) use Bayesian neural networks for feature selection, Baesens et al. (2005) use neural networks for survival analysis to estimate when customers default on their bank loans, or Baesens et al. (2003) develop extraction rules to explain classification results.

More recently, research has emerged on credit scoring using deep learning models. Neagoe et al. (2018) compare the accuracy of feed forward and convolutional neural networks (CNNs)(LeCun et al., 1998). Zhu et al. (2018) couples CNNs with feature selection algorithms to achieve superior perfor-

mance compared to logistic regression and random forests models. Sun and Vasarhelyi (2018) show that deeper neural networks architectures achieve higher predictive performance compared to decision trees, logistic regression and the Bayes classifier. Finally, Wang et al. (2018) obtain significant improvements in the classification of peer-to-peer lending using operation behavior data and coupling the attention mechanism (Mnih et al., 2014) with long short term memory neural networks (Hochreiter and Schmidhuber, 1997).

In all the previous examples, except Baesens et al. (2005), neural networks are used to classify whether a customer will repay a loan. However, neural networks can be used in a broader fashion, for example to approximate probability functions. Kingma and Welling (2013) and Rezende et al. (2014) use neural networks to approximate the log-likelihood function in models without an analytical solution. Their proposed approach offers a flexible and efficient methodology, which is often referred to as Deep Generative Models (DGMs)³.

DGMs use deep learning, which is a field of machine learning that allows algorithms to improve with data (Goodfellow et al., 2016). Another way to understand deep learning is by imagine a system built of a cascade of trainable modules, where we train all modules end-to-end and each of the modules adjust itself to produce the right answer (LeCun, 2018). Deep learning is the current state of machine learning that started back in the 40's with the cybernetic wave, followed by the connectionism in the 80's (Goodfellow et al., 2016).

DGMs have gained popularity across different research fields. For example in health analytics (Rampasek and Goldenberg, 2017; Titus et al., 2018; Way and Greene, 2017a,b), speech emotion recognition (Latif et al., 2017), natural language processing (Bowman et al., 2015; Su et al., 2018), image classification (Kingma et al., 2014; Maaløe et al., 2016), sentiment analysis (Wu et al., 2019; Fu et al., 2019), and clustering (Zheng et al., 2016).

Some of the advantages of the DGMs' methodology are as follows: **Representation learning:** The generative process is based on latent representa-

³Deep Belief Networks, restricted Boltzmann machines, and Generative Adversarial Networks are also examples of DGMs. However, nowadays, models with objective functions based on Variational Inference and parameterized with neural networks are called DGMs.

tions that contain powerful information of the input data. **Dimensionality reduction:** Given that the dimension of the latent space is chosen to be less than in the input space. **Probabilistic ground:** Quantities of interest are modeled directly using probability density functions, allowing to infer queries such as posterior probabilities after the data is observed. **Generative properties:** The model approximates the likelihood of the data using neural networks, which are used for generating new instances of the data.

The main focus of the research conducted in this thesis is to develop novel approaches using the aforementioned properties in DGMs to improve the performance of credit scoring models, to provide solutions to challenges in credit scoring, and to close the gap between plain-vanilla neural networks and DGMs for credit scoring. Further, this doctoral project is an industrial Ph.D. in collaboration with Santander Consumer Bank (SCB) - Nordics. Credit scoring is a core activity for SCB and with this research project SCB wants to expand their expertise in credit scoring with machine learning.

Details about challenges in credit scoring, research objectives, and the approach taken are given in Section 1.2, 1.3, 1.4 respectively.

1.1 Credit Scoring

Credit scoring models transform applicants' information, e.g. economic or demographic factors, into a score, which ranks applicants in terms of their creditworthiness. Then, retail banks use this metric, among other things, to decide whether to grant a loan and to set the eventual price of the loan.

While the applicants' data \mathbf{x} can be obtained in the application form for the loan, the outcome of the loan y , which can be default ($y = 1$) or non-default ($y = 0$), has to be assigned by the bank. This assignment is commonly based on the Basel II accord⁴. That is, any current account which is 90 days past due for any obligation (90+ for short), or if it is already known that there is a high probability of financial loss, or if the debt is written off, is considered a defaulted loan. The 90+ condition must be met within 12 months after the loan contract is signed (see Figure 1.1). It is worth mentioning that the default condition can only be assigned to the current bank's customers since

⁴Banks have some flexibility to decide the exact definition of the outcome of the loan and the definition explained in here is just general.

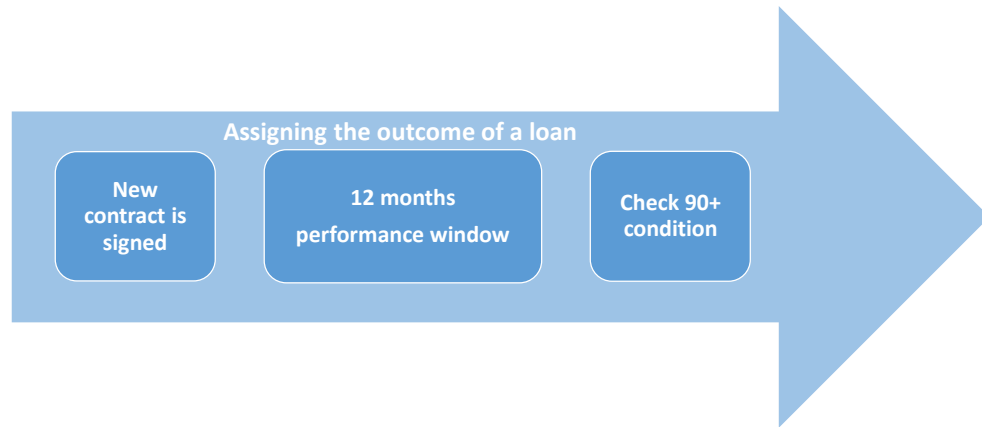


Figure 1.1: Graphical representation of the assignment of the outcome of the loan y based on the 90 days past due criteria. Given that a new contract is signed at some point in time, we monitor the contract for following 12 months and at the end we check whether the contract was at any time 90 days past due. If the contract was indeed 90 days past due, then $y = 1$, else $y = 0$.

the bank does not know whether the 90+ condition is met for the rejected applications.

In statistical terms, credit scoring models aim to discover the relationship between the customers' data \mathbf{x} and the categorical variable y to estimate the conditional probability $Pr(y = 1|\mathbf{x})$. This is the probability for a default application in the following 12 months, or simply default probability, and it is used to rank applications in terms of their ability to repay the loan, i.e. in terms of their creditworthiness.

1.2 Challenges in Credit Scoring

Credit scoring has a major importance not only for retail banks, but also for the people who need access to credit. If the applicants' creditworthiness is underestimated, someone may not have access to credit or the price of credit can be higher than it should be. From a risk management point of view, overestimating the customers' creditworthiness means that a bank is bearing more risk than assumed. Additionally, overestimating the applicants'

creditworthiness impacts the profitability of the bank since the price of the loan is lower than it should be.

Hence, it is important to estimate the default probability as accurate as possible by modeling the relationship between the applicants' data \boldsymbol{x} and the outcome of the loan y . However, there are different factors that make this task rather challenging, for example:

1. **The applicants' data is high-dimensional with complex relationships:** Nowadays, in addition to the data captured in the loan application form, banks can obtain more data in national registers or buy data from credit bureaus. In addition, data engineering can generate more data. Dealing with high-dimensional data is a double-edged sword since models are prone to overfitting.
2. **The through-the-door sample is heterogeneous:** In credit scoring, the through-the-door sample refers to all the people that apply for a loan, despite if the loan was granted or not. In this sample exists specific sub-samples, e.g. students, professionals and pensioners, whose creditworthiness may be affected differently by the same stimulus or, even worse, may not be affected at all.
3. **The outcome of the loan is only known for the current banks' customers:** The outcome of the loan (default or non-default) is the dependent variable y in the statistical model, which is also called the label of the data. Given that banks assign this label based on the actual repayment behavior during the performance window (see Figure 1.1), the labeled data is not the entire through-the-door sample. In other words, the data that can be used for modeling purposes is censored. This means that the unlabeled data has been excluded systematically generating a selection bias problem.
4. **Applicants' or customers' data can be obtained at different points in time:** Banks obtain information about a given applicant in the application form for a loan or buying data from credit bureaus. Further, applicants that obtain the loan generate new information throughout the loan period, e.g. payment and purchase behavior. Therefore, the bank has multiple measurement-modalities providing complementary information. This is an example of multi-modality data.
5. **The relation between the applicants' data \boldsymbol{x} and the outcome**

of the loan y is time dependent: Like in most areas of economics, the customers' payment behavior depends on economic conditions, e.g. unemployment rate, interest rate, taxes, which clearly are time dependent. Therefore, the predictive power of scoring models can be affected by economic shocks or by structural changes in the input data or in macroeconomic variables.

1.3 Research objectives

We leverage Deep Generative Models to provide solutions to some of the challenges mentioned in Section 1.2. Previous studies in credit scoring using neural networks have focused on comparing the model performance for traditional classifiers and neural networks, but to the best of our knowledge no one has develop novel methodologies for credit scoring using DGMs. Therefore, inspired by the results that DGMs have achieved in different research fields, the main focus of this research is to improve model performance in credit scoring models by developing novel methods using DGMs, which provide solutions to well-known challenges in credit scoring. Our main objectives are:

1. Learn low-dimensional data representations of bank customers, which captures the customers' creditworthiness and can support banking activities.
2. Develop segment-based models that can take into account the heterogeneous data sources in credit scoring.
3. Propose new approaches that acknowledge the selection bias problem in credit scoring and can extract information from unlabeled and labeled data.
4. Design novel scoring models that can learn the data modalities in credit data. For example, models that can generate future credit data based on the information captured in the application form, and can use a shared latent data representation for downstream classification tasks.

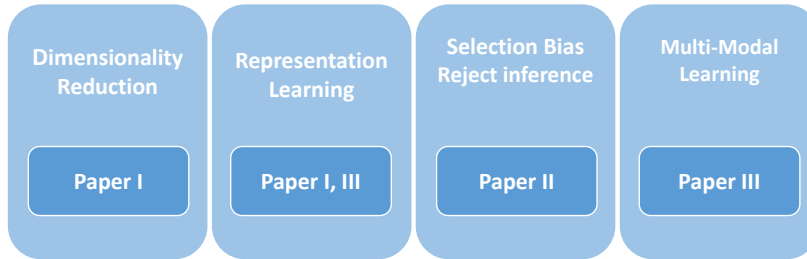


Figure 1.2: Different challenges in the credit scoring literature addressed in the manuscripts done as part of this research project. Specifically, in Paper I we developed a new methodology to learn a data representation for credit data, and we use the variational autoencoder to visualize the representation in a two-dimensional space. Paper II introduces a novel methodology for reject inference in credit scoring to infer the creditworthiness of rejected applications to improve the classification of new loan applications. Finally, in Paper III we developed a new novel model based on multi-modal learning. Our proposed model generates future credit data and classifies loan applications.

1.4 Approach adopted

We use conditional probability to specify the interaction in joint distributions. For example, the joint distribution $p(\mathbf{x}, \mathbf{z})$ of the applicants' data \mathbf{x} and its latent representation \mathbf{z} is given by the product $p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. Concretely, we introduce a supervision stage in the Variational Autoencoder (VAE) (Kingma and Welling, 2013), which is a DGM, to learn useful latent representations that are able to capture the natural clustering of the data in well-defined structures. Further, our proposed methodology is able to capture the spatial coherence of customers' creditworthiness in the latent space of the VAE and can be used in marketing campaigns and in credit risk assessment (Paper I).

We deal with the selection bias problem in traditional credit scoring models by adding the rejected applications to the classification exercise. In this way, the through-the-door sample is used for modeling purposes. However, adding the rejected applications brings a new challenge. The outcome of the loan for rejected applications is unknown and must be inferred. Hence, we infer the unknown label y using Deep Generative Models (DGMs) and Variational Inference (VI). Finally, we use the powerful information embedded in the

latent space of DGMs, in addition to the actual data \mathbf{x} , to estimate the applicants' creditworthiness (Paper II).

Banks have access to multiple measurement-modalities that provide complementary information about customers. To make use of the different modalities in credit data, we developed a novel DGM that is able to learn a shared data representation, which is useful to generate future credit data and for classification. Furthermore, we introduced a novel lower bound that optimizes mutual information between the future view of data and the shared latent representation. Our proposed objective function helps to improve the generative process in our proposed model and also helps to improve the classification of new loan applications (Paper III).

Figure 1.2 shows an overview over the challenges addressed in the different papers produced in this research.

1.5 Brief summary of papers

The papers included in this thesis are:

- I. Mancisidor, R. A., Kampffmeyer, M., Aas, K., and Jenssen, R. (2020b). Learning Latent Representations of Bank Customers with the Variational Autoencoder. *Expert Systems with Applications*, <https://doi.org/10.1016/j.eswa.2020.114020>.
- II. Mancisidor, R. A., Kampffmeyer, M., Aas, K., and Jenssen, R. (2020a). Deep Generative Models for Reject Inference in Credit Scoring. *Knowledge Based Systems*, <https://doi.org/10.1016/j.knosys.2020.105758>.
- III. Mancisidor, R. A., Kampffmeyer, M., Aas, K., and Jenssen, R. (2020c). Generating Customer's Credit Behavior with Deep Generative Models. Submitted to Knowledge Based Systems, September 2020.

Paper I: Uses our proposed semi-supervised version of a VAE for dimensionality reduction of the input data and for representation learning. The dimensionality reduction of the input data is achieved straightforwardly by specifying the dimension in the latent space of the VAE. Furthermore, our contributions in Paper I are as follows:

- We introduce a supervision stage in the VAE framework to learn data

representations that captures the customers' creditworthiness. Specifically, we group the input data using the Weight of Evidence (WoE).

- The learned data representations capture the natural clustering structure of the data and preserve the spatial coherence of creditworthiness.
- The different groups of customers in the well-defined clustering structure of the learned data representation have different levels of creditworthiness and are well suited for marketing campaigns and for credit risk assessment

Paper II: Develops two novel semi-supervised models for reject inference in credit scoring using DGMs. Reject inference attempts to infer the unknown creditworthiness of the rejected applications, which are included in the modeling exercise to fix the selection bias in traditional credit scoring models. The main contributions in Paper II are:

- We combine auxiliary variables and Gaussian mixtures in a semi-supervised framework with DGMs.
- We derive the objective functions for our proposed models and show how they can be parametrized by neural networks and optimized with stochastic gradient descent.
- Our results show that our proposed models achieve higher performance compared to the state-of-the-art methods in credit scoring.

Paper III: Develops a novel model for credit scoring that uses multiple measurement-modalities for a given customer. Specifically, banks collect data at the time of application to decide whether to grant a loan. After the loan is granted, customers generate new data. These two data sets provide complementary information about the customers' creditworthiness, and it is an example of bi-modal data. Our contributions are as follows:

- We address multi-modal learning in credit scoring for the first time and we developed a novel multi-modal learning model that learns a shared latent data representation to generate future credit data and for downstream classification.
- We introduce a novel lower bound, which maximizes mutual information between latent representations and the view of data that is generated after a loan application is accepted. Our proposed lower bound

improves model performance, in terms of reconstruction and classification, compared to the classical lower bound in DGM.

- The latent representations learned by our proposed methodology help to reconstruct future credit data more accurately than other competitive models.

Figure 1.3 shows an overview over some of the models developed in the research conducted in this thesis.

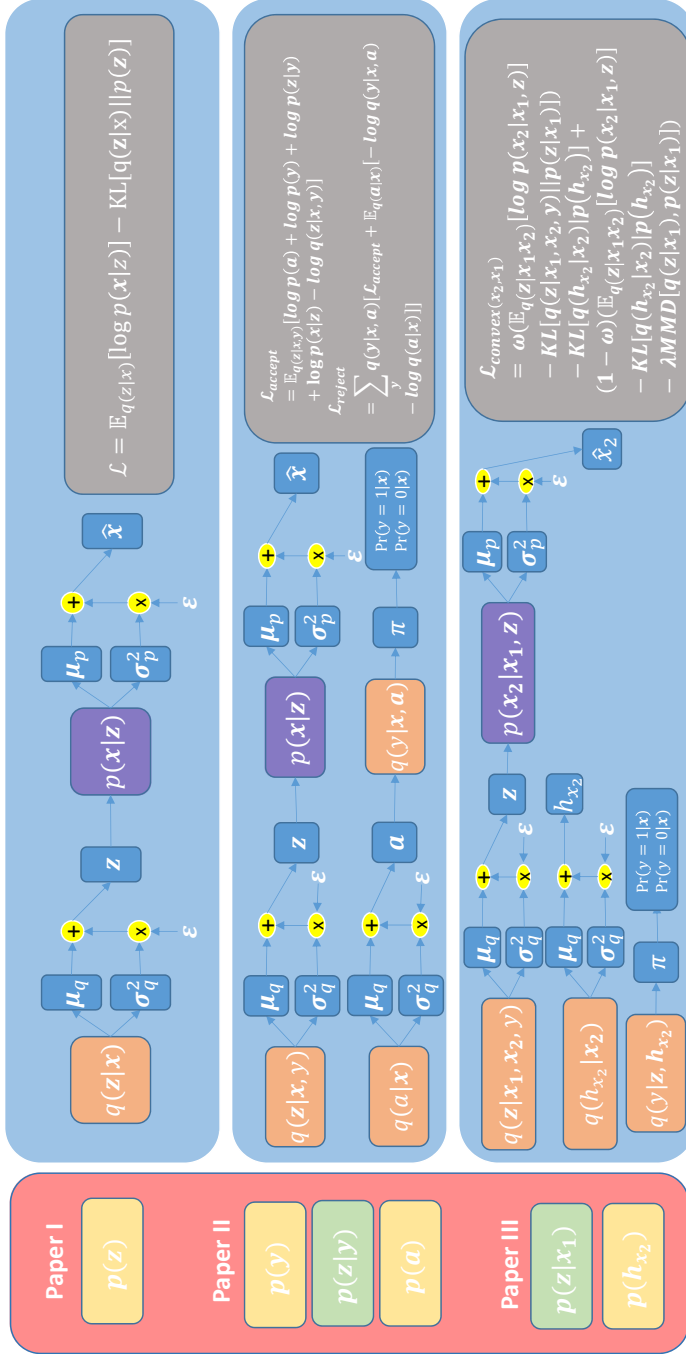


Figure 1.3: Graphical overview over the Deep Generative Models used and developed in this research. Yellow boxes denotes prior probabilities without density parameters, while green boxes are prior with parameters to be estimated. Similarly, orange boxes represent inferred variables and purple boxes depict generative process. All density parameters are parametrized with Multilayer Perceptron (MLP) models. The output of the MLPs and derived variables are represented by the blue boxes. Finally, the lower bound in the models is given in the grey boxes.

1.6 Reading guide

This thesis is divided in the following three parts: i) methodology, ii) summary of research, and iii) included papers.

The purpose with the *methodology* section is to provide the reader with the theoretical background that builds the foundation for what it is presented in this research on credit scoring. To that end, the methodology part includes the topics:

1. Credit Scoring Models
 - 1.1. Linear discriminant function
 - 1.2. Linear Discriminant Analysis
 - 1.3. Linear Programming
 - 1.4. Decision Trees
 - 1.5. Logistic Regression
 - 1.6. Weight of Evidence
2. Probabilistic Graphical Models
 - 2.1. Conditional Probability and The Bayes' Theorem
 - 2.2. Directed Graphical Models
 - 2.3. Variational Inference
3. Deep Generative Models
 - 3.1. Variational Autoencoder
 - 3.2. Deriving the Lower Bound
 - 3.3. The Reparametrization Trick
4. Multilayer Perceptron and the Backpropagation algorithm
5. Segment-based credit scoring

Chapter 1 formalizes the purpose of credit scoring in the bank industry. Further, it introduces some of the most popular statistical models for credit

scoring. Chapter 2 provides an introduction to Probabilistic Graphical Models. We start explaining key concepts in mathematical statistics to motivate the need for approximation approaches, such as Variational Inference. Chapter 3 uses the Variational Autoencoder to motivate the robust machinery in Deep Generative Models. Finally, we present multilayer perceptron and the backpropagation algorithm, together with an idea about a segment-based credit scoring approach in the appendix.

In the *summary of research* part, we discuss the main contributions of the different papers included in this research. Further, we provide concluding remarks and discuss future impact of deep generative models for credit scoring and credit risk. Finally, the papers are included in the *included papers* part.

Part I

Background Theory and Methodology

Chapter 2

Credit Scoring Models

In this chapter, we formalize the statistical concept of credit scoring modeling and present some of the most common methods for credit scoring according to Thomas (2000). Given that most of the models that we present are linear models, we start this chapter introducing linear discriminant functions. For the interested reader, we present a summary of the multilayer perceptron and the backpropagation algorithm in the appendix.

A credit scoring model aims to capture the relationship between the applicants' data $\mathbf{x} \in \mathbb{R}^\ell$ and the (forward-looking) outcome of the loan $y \in \{0, 1\}$ to estimate the probability $Pr(y = 1|\mathbf{x})$. In specific cases, the outcome of the loan can take more than two values, e.g. a bank can send some applications to manual check, hence those applications would get the label $y = 2$. However, in this research we only focus on the binary case.

The outcome of the loan is assigned based on the actual repayment behavior of current banks' customers. That is, if a customer is 90 days past due for any obligation, then that customer has label $y = 1$, otherwise $y = 0$. Additionally, if it is already known that there is a high probability of financial loss or the debt has been written off, the customer has also label $y = 1$. Finally, banks use credit scoring models to accept or reject applications, to set the pricing of the loan and for cross-sales (Anderson, 2007; Thomas, 2000).

According to Bishop (2006), any linear function of the data \mathbf{x} is a linear

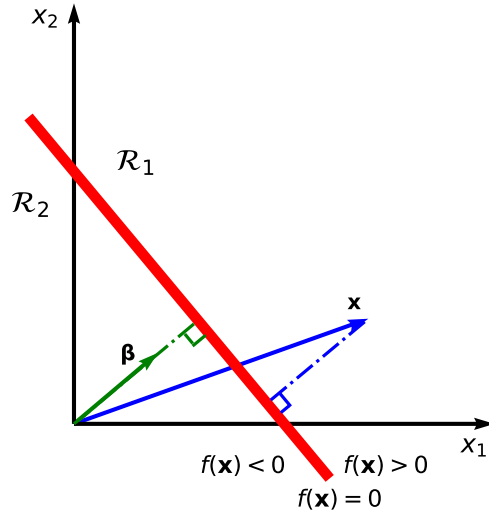


Figure 2.1: Two-dimensional linear discriminant function. The decision boundary is depicted by the red line. All vectors lying on the boundary satisfy the condition $f(\mathbf{x}) = 0$, while vectors satisfying the condition $f(\mathbf{x}) > 0$ are assigned to the class in region \mathcal{R}_1 , otherwise to the class in region \mathcal{R}_2 . Note that $\boldsymbol{\beta}$ determines the orientation of the decision boundary since $\boldsymbol{\beta}$ is orthogonal to any vector lying on the boundary.

discriminant function. Hence,

$$f(\mathbf{x}) = \sum_i \beta_i x_i + \beta_0 = \boldsymbol{\beta}^T \mathbf{x}, \quad (2.1)$$

where $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots, \beta_\ell, \beta_0)^T$ and $\mathbf{x} = (x_1, x_2, \dots, x_\ell, 1)^T$, is a linear discriminant function of \mathbf{x} and the decision boundary is given at $f(\mathbf{x}) = 0$. Further, note that $\boldsymbol{\beta}^T \mathbf{x}$ is the length of the projection of \mathbf{x} onto the decision hyperplane. Hence, for vectors lying on the decision hyperplane we have the condition

$$f(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x} = 0. \quad (2.2)$$

In binary classification problems, \mathbf{x} is assigned to the positive class if $f(\mathbf{x}) \geq 0$, otherwise it is assigned to the negative class. It is easy to show that

if the vectors \mathbf{x}_A and \mathbf{x}_B lie on the decision boundary, we have $\boldsymbol{\beta}^T(\mathbf{x}_A - \mathbf{x}_B) = 0$. This means that $\boldsymbol{\beta}$ is orthogonal to the decision surface and it determines its orientation as shown in Figure 2.1. The reason is that the dot product between vectors \mathbf{x}_A and \mathbf{x}_B can also be expressed as $\mathbf{x}_A \cdot \mathbf{x}_B = \|\mathbf{x}_A\| * \|\mathbf{x}_B\| * \cos \theta$, where $\|\cdot\|$ is the length of a vector and θ is the angle between the two vectors. Since the angle between two orthogonal vectors is 90° and $\cos 90^\circ = 0$, the dot product between orthogonal vectors is 0.

2.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) (Fisher, 1936) aims to find a linear transformation of the input data, which *best* discriminates the two classes in the credit scoring problem, i.e. separate $y = 1$ from $y = 0$ using the transformation $f(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x}$.

Let G_1 and G_2 correspond to groups including only customers with labels $y = 1$ and $y = 0$, and each group has n_1 and n_2 observations respectively. Hence, the mean vector of group G_1 and G_2 are

$$\boldsymbol{\mu}_1 = \frac{1}{n_1} \sum_{n \in G_1} \mathbf{x}_n \quad \text{and} \quad \boldsymbol{\mu}_2 = \frac{1}{n_2} \sum_{n \in G_2} \mathbf{x}_n.$$

We want to find $\boldsymbol{\beta}$ that maximizes the difference of the means in the projection $\boldsymbol{\beta}^T \boldsymbol{\mu}_1 - \boldsymbol{\beta}^T \boldsymbol{\mu}_2$. Hence, to best discriminate between the two classes $y = 1$ and $y = 0$ in credit scoring, the unknown vector $\boldsymbol{\beta}$ also needs to minimize the within variation of the projected data. That is, minimize

$$s_1 = \sum_{n \in G_1} (\boldsymbol{\beta}^T \mathbf{x}_n - \boldsymbol{\beta}^T \boldsymbol{\mu}_1)^2 \quad \text{and} \quad s_2 = \sum_{n \in G_2} (\boldsymbol{\beta}^T \mathbf{x}_n - \boldsymbol{\beta}^T \boldsymbol{\mu}_2)^2.$$

This leads to the maximization of the Rayleigh quotient

$$\begin{aligned} J(\boldsymbol{\beta}) &= \frac{(\boldsymbol{\beta}^T \boldsymbol{\mu}_1 - \boldsymbol{\beta}^T \boldsymbol{\mu}_2)^2}{\sum_{n \in G_1} (\boldsymbol{\beta}^T \mathbf{x}_n - \boldsymbol{\beta}^T \boldsymbol{\mu}_1)^2 + \sum_{n \in G_2} (\boldsymbol{\beta}^T \mathbf{x}_n - \boldsymbol{\beta}^T \boldsymbol{\mu}_2)^2} \\ J(\boldsymbol{\beta}) &= \frac{\boldsymbol{\beta}^T (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\beta}}{\boldsymbol{\beta}^T \sum_{n \in G_1} (\mathbf{x}_n - \boldsymbol{\mu}_1) (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\beta} + \boldsymbol{\beta}^T \sum_{n \in G_2} (\mathbf{x}_n - \boldsymbol{\mu}_2) (\mathbf{x}_n - \boldsymbol{\mu}_2)^T \boldsymbol{\beta}} \\ J(\boldsymbol{\beta}) &= \frac{\boldsymbol{\beta}^T \mathbf{S}_B \boldsymbol{\beta}}{\boldsymbol{\beta}^T \mathbf{S}_W \boldsymbol{\beta}}, \end{aligned} \tag{2.3}$$

where $\mathbf{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$ and $\mathbf{S}_W = \sum_{n \in G_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T + \sum_{n \in G_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T$. The former is usually referred to as the between class covariance matrix and the latter as the within class covariance matrix.

Adding to the constraint $\boldsymbol{\beta}^T \mathbf{S}_W \boldsymbol{\beta} = 1$ in the denominator of Equation 2.3, we obtain the Lagrangian function

$$\mathcal{L} = \boldsymbol{\beta}^T \mathbf{S}_B \boldsymbol{\beta} - \lambda[\boldsymbol{\beta}^T \mathbf{S}_W \boldsymbol{\beta} - 1]. \quad (2.4)$$

Taking the derivative of the Lagrangian function with respect to $\boldsymbol{\beta}$, at the solution we have that

$$\mathbf{S}_B \boldsymbol{\beta} = \lambda \mathbf{S}_W \boldsymbol{\beta}. \quad (2.5)$$

Note that the left hand side of Equation 2.5 is $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\beta}$ and that $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\beta}$ is a scalar. Hence, $\mathbf{S}_B \boldsymbol{\beta}$ points in the direction of $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$, and we only care about the direction of $\boldsymbol{\beta}$ since the direction of $\boldsymbol{\beta}$ determines the orientation of the decision surface, see Figure 2.1. Hence, after dropping the constant terms, the vector that maximizes the separation between the projected means and minimizes the within-class dispersion is given by

$$\boldsymbol{\beta} \propto \mathbf{S}_W^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2). \quad (2.6)$$

2.2 Linear Programming

A linear programming (LP) model maximizes, or minimizes, an objective function subject to some constraints. Both the objective and constraints are linear functions. For example, let the objective function be

$$\text{maximize } x_1 + x_2 \quad (2.7)$$

subject to:

$$\begin{aligned} x_1 + 2x_2 &\leq 4 \\ 4x_1 + 2x_2 &\leq 12 \\ -x_1 + x_2 &\leq 1 \\ x_1 &\geq 0 \\ x_2 &\geq 0. \end{aligned}$$

This LP problem can be solved by plotting the constraint set, i.e. the set of combinations between x_1 and x_2 that satisfy the inequalities in the constraints. The constraint set together with the five constraints is depicted in

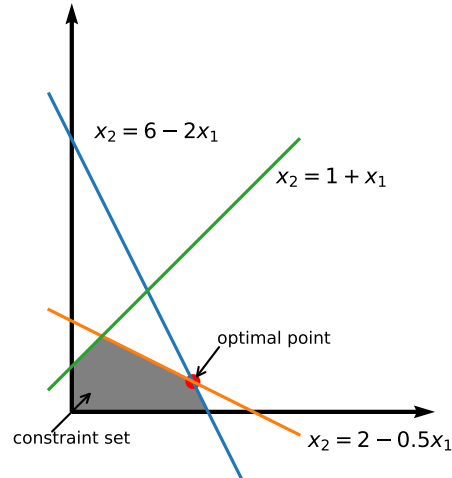


Figure 2.2: A linear programming model satisfying the conditions in Equation 2.7. The grayed area shows the constraint set, which is the set of possible combinations, and the three lines show the model constraints. In this example, it is easy to see that the optimal solution is at the orange dot since it is that point in the constraint set that maximizes the objective function.

Figure 2.2. Looking at the diagram, it is clear that the optimal combination lies on the edges of the constraint set, and for this particular problem the orange point maximizes the objective function.

There are several LP models for binary classification and others for multi-class classification. However, in this section we explain the model presented in Hardy Jr and Adrian Jr (1985), which is to our knowledge the first application of LP in credit scoring. This model is a variation of the one proposed by Freed and Glover (1981) and extended in Bajgier and Hill (1982).

The linear programming model for credit scoring aims to

$$\text{maximize } \sum_{j=1}^n a_j D_j^+ - \sum_{j=1}^n b_j D_j^- \quad (2.8)$$

subject to:

$$\begin{aligned} \boldsymbol{\beta}^T \mathbf{x}_j - D_j^+ + D_j^- &\geq C & \text{if } y_j = 0 \\ \boldsymbol{\beta}^T \mathbf{x}_j + D_j^+ - D_j^- &\leq C & \text{if } y_j = 1 \\ \sum_{j=1}^n D_j^+ &\leq nC, \end{aligned}$$

where D_j^+ is the distance to the cutoff score C_j for correctly classified observations, D_j^- is the distance to the cutoff score for misclassified observations, a_j is the relative penalization weights for correctly classified observations, b_j is the relative penalization weights for misclassified observations satisfying the condition $b_j > a_j$. Finally, $\boldsymbol{\beta}$ are the unknown parameters in the linear classifier.

Note that Equation 2.8 attempts to maximize correctly classified observations \mathbf{x}_j and minimize misclassified cases by finding the weights $\boldsymbol{\beta}$ that *best* separate the two class of customers ($y = 0$ and $y = 1$). It also incorporates a penalty term for misclassification.

2.3 Decision Trees

The automatic interaction detection (AID) framework (Morgan and Sonquist, 1963) is the first decision tree algorithm. This algorithm iteratively splits the dependent variable into two subgroups, or nodes, using one predictor variable. Note that if a predictor has k categories, the number of possible splits, in a given iteration, is $2^{k-1} - 1$. Further, AID assumes that predictors are discrete variables, either nominal or ordered categories. In case of continuous predictor variables, the categories should be formed beforehand.

The nodes that are formed at each iteration should maximize the explained

sum of squares (ESS)

$$\begin{aligned}
 ESS &= n_1 \bar{y}_1^2 + n_2 \bar{y}_2^2 \\
 &= \frac{\left(\sum_{n \in G_1} y_n\right)^2}{n_1} + \frac{\left(\sum_n y_n - \sum_{n \in G_1} y_n\right)^2}{N - n_1}
 \end{aligned} \tag{2.9}$$

where $\bar{y}_1 = \frac{1}{n_1} \sum_{n \in G_1} y_n$ and $\bar{y}_2 = \frac{1}{n_2} \sum_{n \in G_2} y_n$ are the means of the dependent variable y in the groups G_1 and G_2 , respectively, n_1 and n_2 are the number of observations in each group, and N is the total number of observations.

Therefore, it is enough to know the number of cases in one of the subgroups and the sum of the dependent variable to account for the reduction in error sum of squares¹. Finally, a new split is kept only if the reduction in error sum of squares is larger than 1% of the total sum of squares for the whole sample, otherwise the algorithm searches for another partition.

Apart from AID, there are several decision tree algorithms. The main difference between them are the objective function to optimized at each iteration and how do they decide upon whether to split a given node. For example, ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993) decision tree algorithms use entropy to decide the optimal split at each iteration. Specifically, ID3 defines the reduction in entropy, after a given split on the predictor x , as

$$\text{Gain}(D, x) = \text{Entropy}(D) - \sum_{G_i \in x} \text{Pr}(G_i) \text{Entropy}(G_i), \tag{2.10}$$

where G_i is the i 'th group of the entire data set D and Entropy is the Shannon entropy. Hence, ID3 searches for the predictor with the largest Gain as given in Equation 2.10. On the other hand, C4.5 uses a normalized version of Equation 2.10 to avoid splits on predictors with many unique categories. Further, the CART algorithm (Breiman et al., 1984) uses the Gini index

$$\text{Gini} = 1 - \sum_{G_i \in x} \text{Pr}(G_i)^2 \tag{2.11}$$

as objective function to create binary splits.

¹The reduction in error sum of squares is the same as the increase in ESS, just with the opposite sign.

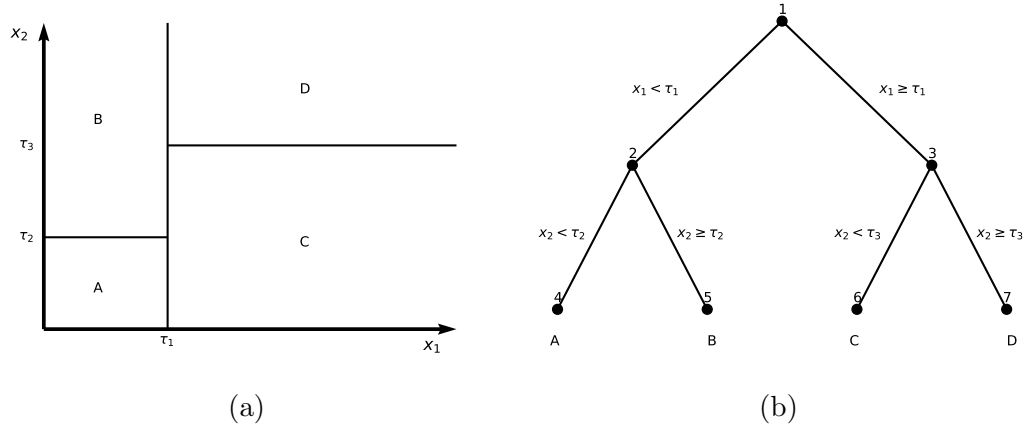


Figure 2.3: The left panel shows the regions created by the (axis-parallel) recursive partitioning in decision tree algorithms. The parent node 1, in the right panel, is partitioned on x_1 at the optimal threshold τ_1 . This iteration creates nodes 2 and 3. Further, node 2 is partitioned on x_2 at the threshold τ_2 creating the regions A and B. Finally, node 3 is further split on x_2 creating the regions C and D.

Figure 2.3 shows the recursive partitioning in decision trees. Note that decision trees are not linear discriminant models, as defined in Equation 2.1, since the classification of the input space happens in an axis-parallel fashion.

2.4 Logistic Regression

Logistic regression (LR) is the most popular technique for credit scoring (Thomas, 2000; Lessmann et al., 2013).

The obvious problem with Equation 2.1, if we want to model a probability $Pr(y = 1|\mathbf{x})$, is that it can take values of $(-\infty, +\infty)$. To circumvent this problem, logistic regression uses the logit transformation

$$\text{logit}(Pr(\cdot)) = \log \frac{Pr(y = 1|\mathbf{x})}{1 - Pr(y = 1|\mathbf{x})} = \boldsymbol{\beta}^T \mathbf{x}. \quad (2.12)$$

Note that under this setup, $Pr(y = 1|\mathbf{x})$ is given by the sigmoid function

$$\begin{aligned}
 \log \frac{Pr(y = 1|\mathbf{x})}{1 - Pr(y = 1|\mathbf{x})} &= \boldsymbol{\beta}^T \mathbf{x} \\
 \frac{Pr(y = 1|\mathbf{x})}{1 - Pr(y = 1|\mathbf{x})} &= \exp(\boldsymbol{\beta}^T \mathbf{x}) \\
 \frac{1}{1 - Pr(y = 1|\mathbf{x})} &= \exp(\boldsymbol{\beta}^T \mathbf{x}) + 1 \\
 Pr(y = 1|\mathbf{x}) &= \frac{\exp(\boldsymbol{\beta}^T \mathbf{x})}{1 + \exp(\boldsymbol{\beta}^T \mathbf{x})}. \tag{2.13}
 \end{aligned}$$

In the logistic regression we know that $Pr(y = 0|\mathbf{x}) = 1 - Pr(y = 1|\mathbf{x})$, therefore at the decision boundary the following condition must be true

$$\begin{aligned}
 Pr(y = 1|\mathbf{x}) &= Pr(y = 0|\mathbf{x}) \\
 \frac{\exp(\boldsymbol{\beta}^T \mathbf{x})}{1 + \exp(\boldsymbol{\beta}^T \mathbf{x})} &= \frac{1}{1 + \exp(\boldsymbol{\beta}^T \mathbf{x})} \\
 \exp(\boldsymbol{\beta}^T \mathbf{x}) &= 1 \\
 \boldsymbol{\beta}^T \mathbf{x} &= 0. \tag{2.14}
 \end{aligned}$$

Hence, Equation 2.14 shows that the logistic regression is a linear discriminant model.

2.5 Weight of Evidence

In credit scoring models, it is common to transform the input data \mathbf{x} into the weight of evidence (WoE), and this transformation has become the standard (Abdou, 2009). Further, it is important to stress that WoEs can be used as input features in any credit scoring model.

This transformation is done in the following way. Given that the m 'th feature $x_m \in \mathbf{x}$ is continuous, we divide its values into K bins B_1, B_2, \dots, B_K . In the case of categorical variables, the different categories are already these bins.

Fine classing approach								
Age	Count	Total Distribution	Goods	Distribution Goods	Bads	Distribution Bads	Bad Rate	WoE
Missing	1 000	2.5%	860	2.38%	140	3.65%	14.00 %	-0.4272
18-22	4 000	10%	3 040	8.41%	960	25.00%	24.00 %	-1.0898
23-26	6 000	15%	4 920	13.61%	1 080	28.13%	18.00 %	-0.7261
27-29	9 000	22.5%	8 100	22.40%	900	23.44%	10.00 %	-0.0453
30-35	10 000	25.0%	9 500	26.27%	500	13.02%	5.00 %	0.7019
36-44	7 000	17.5%	6 800	18.81%	200	5.21%	2.86 %	1.2839
44+	3 000	7.5%	2 940	8.13%	60	1.56%	2.00 %	1.6493
Total	40 000	100%	36 160	100%	3 840	100%	9.60 %	
Coarse classing approach								
Missing	1 000	2.5 %	860	2.38 %	140	3.65 %	14.00%	-0.4272
18-29	19 000	47.5 %	16 060	44.41%	2 940	76.56 %	15.47 %	-0.5445
30-44+	20 000	50%	19 240	53.20%	760	19.79 %	3.80 %	0.9889
Total	40 000	100%	36160	100 %	3840	100 %	9.60%	

Table 2.1: Weight of Evidence (WoE) transformation of the variable age. The top panel shows the fine classing approach, while the bottom panel shows the coarse approach where only three groups are created.

Then, the WoE for the k 'th bin of the m 'th feature is

$$\begin{aligned}
 \text{WoE}_{k,m} &= \log \frac{\text{Pr}(x_m \in B_k | y = 0)}{\text{Pr}(x_m \in B_k | y = 1)} \\
 &= \log \frac{\frac{1}{n} \sum_{i=1}^n [x_{i,m} \in B_{k,m} \text{ and } y_i = 0]}{\frac{1}{n} \sum_{i=1}^n [x_{i,m} \in B_{k,m} \text{ and } y_i = 1]}, \tag{2.15}
 \end{aligned}$$

where n is the total number of observations and $[\cdot]$ is the Iverson bracket. Note that the number of bins can vary for different features.

Table 2.1 shows the difference between fine and coarse classing. In the fine classing approach, we create K bins, which provide the finest granularity. Then, fine bins with similar risk are binned into smaller groups resulting in the coarse classing, see Anderson (2007) for more details. Note that the discretization induced by WoE has a couple of advantages. First, missing values have their own WoE and imputation is not needed. Second, WoE also induce a common scale in all predictors and it is the same scale as the dependent variable in LR. This advantage is useful in cases where both age (in tens) and salary (probably in hundreds or even in thousands) are used in a given LR model.

Chapter 3

Probabilistic Graphical Models

According to Koller and Friedman (2009), there are three important concepts in building useful probabilistic models. *Representation, inference* and *learning*. This chapter introduces the first two and *learning* is presented throughout Chapter 4.

It is often possible to use domain-knowledge to represent complex models in an understandable and tractable way. This is possible given that variables commonly interact directly only with few others.

Further, we are interested in a model representation that is useful to infer queries such as $Pr(y = 1|\mathbf{x})$, i.e. the default probability for a new loan given the evidence encoded in the customer's data. This kind of (posterior) inference is at the core of credit scoring.

Finally, we would like to learn from previous experiences and use a probabilistic model to reason about the outcome of a new loan and be able to estimate $Pr(y = 1|\mathbf{x})$. Hence, the probabilistic encoding must be such that it facilitates learning from data in an efficient and scalable way.

3.1 Conditional Probability and The Bayes' Theorem

Suppose a bank wants to understand the relationship between customers' *salary* s and the outcome of the loan y . To simplify the task, bank analysts

transform *salary* into the categories *low* and *high* salary. One way to understand this relationship is by obtaining past data and model this relationship somehow.

However, the bank is not interested in understanding the relationship between s and y in the past. They are interested in the case where the bank observes the salary of a new applicant and how that salary will influence the outcome of the loan. In other words, how should the bank update their beliefs (based on previous data) about the relationship between y and s in this case?

Conditional probability deals with these kind of problems. Let us assume that A defines the event $Salary = high$ and B the event $y = 1$. Hence, the conditional probability of a defaulted loan given an applicant with high salary is

$$Pr(B|A) = \frac{Pr(A, B)}{Pr(A)}, \quad (3.1)$$

where the sample space $\Omega = \bigcap_{i=1}^2 B_i$, $Pr(A, B) = Pr(A \cap B)$ is the probability of the intersection of A and B , and $Pr(A)$ is given by the law of total probability

$$Pr(A) = \sum_{i=1}^2 Pr(A|B_i)Pr(B_i). \quad (3.2)$$

Hence, the conditional probability is an *informed* measure after we have observed the applicant's salary.

Note that Equation 3.1 suggests $Pr(B|A)Pr(A) = Pr(A, B)$. Generally, given a set of n events $\{\bigcap_{i=1}^n E_i\} = \Omega$, the chain rule of conditional probabilities is

$$Pr(E_1, E_2, \dots, E_n) = Pr(E_1)Pr(E_2|E_1) \cdots Pr(E_n|E_{n-1}, \dots, E_1), \quad (3.3)$$

i.e. the joint probability of all possible events in Ω can be expressed in terms of conditional probabilities.

Putting together the definition of conditional probability and the chain rule of conditional probabilities, we obtain the seminal Bayes' Theorem

$$Pr(B|A) = \frac{Pr(A|B)Pr(B)}{Pr(A)}, \quad (3.4)$$

where $Pr(B|A)$ is the posterior probability, $Pr(A|B)$ is the data distribution, $Pr(B)$ is the prior probability of B , and $Pr(A)$ is the marginal distribution of A .

It is important to mention that Equation 3.4 can be expressed in terms of continuous random variables and not discrete events as we introduced it in this section. In the case of continuous variables, we replace $Pr(E)$ for $p(\mathbf{x})$, where $p(\cdot)$ is a probability density function and \mathbf{x} is a continuous random variable. Further, we can use the data distribution to update our knowledge not only about quantities of interest but also about unknown population parameters.

Hence, the Bayes' Theorem adopts the more general notation

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{p(\boldsymbol{\theta})p(\mathbf{x}|\boldsymbol{\theta})}{p(\mathbf{x})}, \quad (3.5)$$

where $\boldsymbol{\theta}$ are the unknown quantities or population parameters, $p(\boldsymbol{\theta}|\mathbf{x})$ is the posterior distribution, $p(\mathbf{x}|\boldsymbol{\theta})$ is the data distribution, $p(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$, and $p(\mathbf{x}) = \int p(\boldsymbol{\theta}, \mathbf{x})d\boldsymbol{\theta}$ is the marginal distribution of the data, which is also referred to as the *evidence*. Note that, the function $p(\mathbf{x}|\boldsymbol{\theta})$ also called the *likelihood function* whenever the data is regarded as fixed.

Conjugate models for exponential distributions

For some exponential family distributions the posterior distribution in the Bayes' Theorem has a closed form and it can be derived analytically. In this case, we say that the prior is conjugate to the likelihood. Let us assume that the feature vector \mathbf{x} with customer's characteristics is multivariate Gaussian distributed, i.e. $\mathcal{N} \sim (\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with known covariance matrix $\boldsymbol{\Sigma}$. Further, assume that the unknown parameter $\boldsymbol{\mu}$ is also Gaussian distributed, i.e. $\boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, where $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$ are hyperparameters. Using the Bayes' Theorem we can find the posterior distribution

$$p(\boldsymbol{\mu}|\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\mu})p(\boldsymbol{\mu})}{\int p(\mathbf{x}, \boldsymbol{\mu})d\boldsymbol{\mu}} \propto p(\mathbf{x}|\boldsymbol{\mu})p(\boldsymbol{\mu}), \quad (3.6)$$

where the marginal distribution is $p(\mathbf{x}) = \int p(\mathbf{x}, \boldsymbol{\mu})d\boldsymbol{\mu}$. Note that in the last expression in Equation 3.6 we express the posterior distribution $p(\boldsymbol{\mu}|\mathbf{x})$

proportional to the joint density $p(\mathbf{x}, \boldsymbol{\mu})$ since the evidence $p(\mathbf{x})$ does not depend on $\boldsymbol{\mu}$.

We can now compute the posterior distribution (up to a proportionality constant) by completing the square, collecting common terms, and pulling out constant factors, as follows

$$\begin{aligned}
p(\boldsymbol{\mu}|\mathbf{x}) &\propto \exp\left(-\frac{1}{2}\left[(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) + (\boldsymbol{\mu} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\mu} - \boldsymbol{\mu}_0)\right]\right) \\
&\propto \exp\left(-\frac{1}{2}\left[\boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} - 2\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu} - 2\boldsymbol{\mu}^T \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0\right]\right) \\
&= \exp\left(-\frac{1}{2}\left[\boldsymbol{\mu}^T (\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_0^{-1}) \boldsymbol{\mu} - 2\boldsymbol{\mu}^T (\boldsymbol{\Sigma}^{-1} \mathbf{x} + \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0)\right]\right) \\
&= \exp\left(-\frac{1}{2}\left[\boldsymbol{\mu}^T \mathbf{A} \mathbf{A}^{-1} \mathbf{A} \boldsymbol{\mu} - 2\boldsymbol{\mu}^T \mathbf{A} \mathbf{A}^{-1} \mathbf{B} - \mathbf{B}^T \mathbf{A}^{-1} \mathbf{B} + \mathbf{B}^T \mathbf{A}^{-1} \mathbf{B}\right]\right) \\
&\propto \exp\left(-\frac{1}{2}\left[(\mathbf{A} \boldsymbol{\mu} - \mathbf{B})^T \mathbf{A}^{-1} (\mathbf{A} \boldsymbol{\mu} - \mathbf{B})\right]\right), \tag{3.7}
\end{aligned}$$

where $\mathbf{A} = \boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_0^{-1}$ and $\mathbf{B} = \boldsymbol{\Sigma}^{-1} \mathbf{x} + \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0$. Note that Equation 3.7 is a second order polynomial in $\boldsymbol{\mu}$, so it is multivariate Gaussian distributed. The mode of the posterior distribution is given at $\frac{\partial \log p(\boldsymbol{\mu}|\mathbf{x})}{\partial \boldsymbol{\mu}} = 0$ and its variance is given by $-\frac{\partial^2 \log p(\boldsymbol{\mu}|\mathbf{x})}{\partial \boldsymbol{\mu}^2}^{-1}$. Therefore, we can derive an analytical expression for the posterior distribution as follows

$$\begin{aligned}
\frac{\partial \log p(\boldsymbol{\mu}|\mathbf{x})}{\partial \boldsymbol{\mu}} &= -\frac{1}{2}(2\mathbf{A} \boldsymbol{\mu} - 2\mathbf{B})^T \mathbf{A}^{-1} \mathbf{A} \\
&= -\mathbf{A} \boldsymbol{\mu} + \mathbf{B} = 0, \tag{3.8}
\end{aligned}$$

and

$$\frac{\partial^2 \log p(\boldsymbol{\mu}|\mathbf{x})}{\partial \boldsymbol{\mu}^2} = -\mathbf{A}, \tag{3.9}$$

so the posterior is also a multivariate Gaussian distribution

$$p(\boldsymbol{\mu}|\mathbf{x}) \sim \mathcal{N}(\mathbf{A}^{-1} \mathbf{B}, \mathbf{A}^{-1}). \tag{3.10}$$

3.2 Directed Graphical Models

Let us take a relative easy example in credit scoring to motivate directed graphical models, which are pictorial and compact representations for complex joint distributions where domain-knowledge is used to specify conditional independence. This is a key concept in deep generative models.

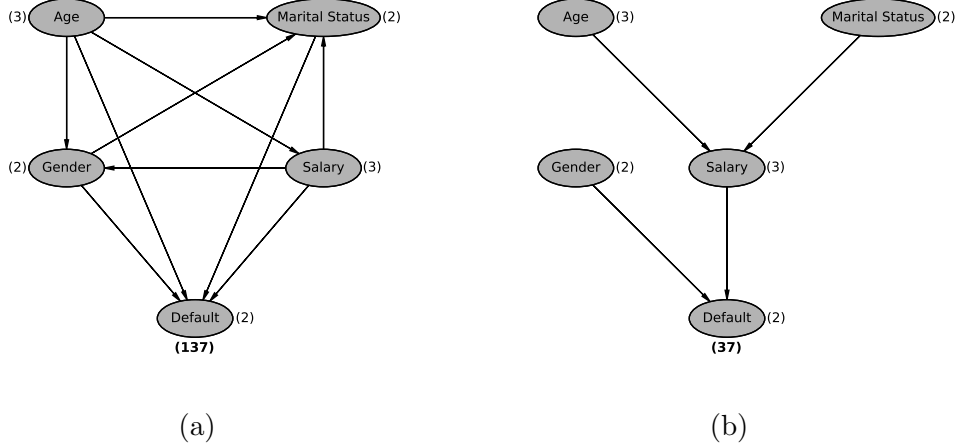


Figure 3.1: Probabilistic graphical models for the factorization in Equation 3.11 (left panel) and 3.12 (right panel). The numbers in parenthesis show the numbers of different states in the sample space for each variable, and the bold number shows the total number of nonredundant states.

Let us assume that a bank collects some features describing the payment behavior of their current customers. These variables are *age*, *salary*, *gender* and *marital status* (*ms*). For the sake of simplicity, they create 3 groups for both *age* and *salary*. In addition to these customers' features, the outcome of the loan *y* is also collected. Hence, $age \in \{a_1, a_2, a_3\}$, $salary \in \{s_1, s_2, s_3\}$, $gender \in \{g_1, g_2\}$, $ms \in \{ms_1, ms_2\}$, and $y \in \{0, 1\}$. The sample space in this simple scenario has 137 (nonredundant) possible outcomes to these 5 random variables and the joint density, using Equation 3.3, is

$$p(age, salary, gender, ms, y) = p(age)p(salary|age)p(gender|salary, age) \\ p(ms|gender, salary, age)p(y|ms, gender, salary, age) \quad (3.11)$$

A senior analyst suggests, based on previous experience and internal information, that *age*, *gender* and *ms* are independent variables. Further, the analyst suggests that the outcome of the loan only depends on *gender* and *salary*. Similarly, salary depends on both *age* and *ms*. We can express this knowledge mathematically using both conditional distributions and a picto-

rial representation of it. The joint distribution can then be expressed as

$$p(\textit{age}, \textit{salary}, \textit{gender}, \textit{ms}, y) = p(\textit{age})p(\textit{gender})p(\textit{ms}) \\ p(\textit{salary}|\textit{ms}, \textit{age})p(y|\textit{salary}, \textit{gender}). \quad (3.12)$$

Figure 3.1 (a) and (b) show the difference between Equation 3.11 and 3.12 using directed graphs, which are also referred to as Bayesian Networks (BN). Each variable in Equation 3.11 and 3.12 is represented by nodes, and conditional distributions are denoted by arrows. For example, for the conditional probability $p(\textit{salary}|\textit{ms}, \textit{age})$ we have two arrows from nodes *age* and *ms* to *salary*. Further, given that the arrows go from *age* and *ms* to *salary*, both *age* and *ms* are *parents* of node *salary*, and *salary* is their *child*.

Note that the information provided by the senior analyst has a major consequence for the modeling exercise. The number of states in the sample space is reduced from 137 to 37 as shown in Figure 3.1. Hence, incorporating domain-knowledge in the form of conditional independence introduces model parsimony. Furthermore, model training may become feasible if the sample space is reduced to a number of states where model training is doable.

3.3 Variational Inference

The example introduced in Section 3.2 is a simplified version of a credit scoring model. In reality, banks have access to more features about their current customers and applicants. In addition, some banks buy more information from credit bureaus since they have access to a wider audience. Hence, the available data is a high-dimensional vector $\mathbf{x} = (x_1, x_2, \dots, x_\ell)^T$.

Note that the feature vector \mathbf{x} reflects different information about the same person. Hence, it would make sense to assume that \mathbf{x} is generated (Murphy, 2012), or governed (Blei et al., 2017), by an unseen (latent) variable \mathbf{z} . In statistics, these sort of relations are described in Latent Variable Models (LVMs), which were introduced by Lazarsfeld (1950, 1954) in a study about the perception of American soldiers about foreign cultures during World War II.

In Section 3.1 we showed that for conjugate priors we can arrive to the posterior distribution analytically by completing the square and deriving the moments of a Gaussian distribution. However, for many interesting problems,

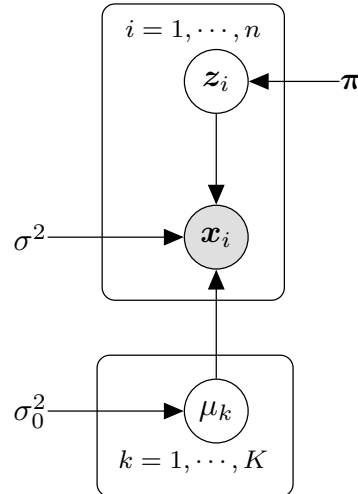


Figure 3.2: Plate notation for the Bayesian Gaussian Mixture Model (GMM) for univariate Gaussian variables presented in Section 3.3.

including LVMs, this is not the case because the integral in the marginal distribution $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$ is intractable. Hence, we need to rely on approximation methods.

To understand why we cannot arrive to analytical expressions for posterior distributions in LVMs let us take as example a Bayesian Gaussian Mixture Model (GMM) for univariate Gaussian variables. Suppose we have a data set of observations $X = \{x_1, x_2, \dots, x_n\}$ and a data set of latent variables $\mathbf{Z} = \{z_1, z_2, \dots, z_n\}$ where for each x_i there is a one-hot-encoded latent variable z_i indicating the k 'th component in the GMM to which x_i belongs to. Therefore, $z_i^T \boldsymbol{\mu}$ selects one μ_k from $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_K)^T$. Finally, for simplicity, we assume that μ_k is drawn independently from a common distribution with known parameters, and that the variance σ^2 in the Gaussian distribution for $x_i | z_i, \boldsymbol{\mu}$ is also known. Hence,

$$\begin{aligned} \mu_k &\sim \mathcal{N}(0, \sigma_0^2) & k = 1, \dots, K, \\ z_i &\sim \text{cat}(\boldsymbol{\pi}) & i = 1, \dots, n, \\ x_i | z_i, \boldsymbol{\mu} &\sim \mathcal{N}(z_i^T \boldsymbol{\mu}, \sigma^2) & i = 1, \dots, n, \end{aligned}$$

where $\mathcal{N}(\cdot)$ denotes the Gaussian distribution and $\text{cat}(\cdot)$ is a categorical

distribution with parameter vector $\boldsymbol{\pi}$. Figure 3.2 shows the GMM in plate notation.

The posterior distribution for this model is given by the joint density of all variables

$$p(\boldsymbol{\mu}, \mathbf{Z}, X) = p(\boldsymbol{\mu}) \prod_{i=1}^n p(z_i) p(x_i | z_i, \boldsymbol{\mu}), \quad (3.13)$$

the marginal distribution of the data

$$p(X) = \int \sum_{\mathbf{z}} p(X, \mathbf{Z}, \boldsymbol{\mu}) d\boldsymbol{\mu}, \quad (3.14)$$

and the Bayes' Theorem of Equation 3.5

$$p(\boldsymbol{\mu}, \mathbf{Z} | X) = \frac{p(\boldsymbol{\mu}) \prod_{i=1}^n p(z_i) p(x_i | z_i, \boldsymbol{\mu})}{\int \sum_{\mathbf{z}} p(X, \mathbf{Z}, \boldsymbol{\mu}) d\boldsymbol{\mu}}, \quad (3.15)$$

where we are able to move $p(\boldsymbol{\mu})$ outside the likelihood function given that it is assumed to be independent.

The problem of calculating this posterior distribution arises when we try to derive the marginal distribution $p(X) = \int \sum_{\mathbf{z}} p(X, \mathbf{Z}, \boldsymbol{\mu}) d\boldsymbol{\mu}$, which is given by the K -dimensional integral

$$\begin{aligned} p(X) &= \int \cdots \int p(\boldsymbol{\mu}) \prod_{i=1}^n \sum_{z_i} p(z_i) p(x_i | z_i, \boldsymbol{\mu}) d\mu_1 \cdots d\mu_K \\ &= \int \cdots \int p(\boldsymbol{\mu}) \prod_{i=1}^n p(x_i | \boldsymbol{\mu}) d\mu_1 \cdots d\mu_K \\ &= \int \cdots \int p(\boldsymbol{\mu}) [p(x_1 | \boldsymbol{\mu}) \times p(x_2 | \boldsymbol{\mu}) \times \cdots \times p(x_n | \boldsymbol{\mu})] d\mu_1 \cdots d\mu_K. \end{aligned} \quad (3.16)$$

Note that the K -dimensional integral in Equation 3.16 has $\mathcal{O}(K^n)$ complexity (Blei et al., 2017). Further, as pointed out by Blei et al. (2017), it is possible to express Equation 3.16 as

$$p(X) = \sum_{\mathbf{z}_i} p(\mathbf{z}_i) \int p(\boldsymbol{\mu}) \prod_{i=1}^n p(x_i | z_i, \boldsymbol{\mu}) d\boldsymbol{\mu},$$

which has the advantage that the integral has a closed form (just as in the example in Section 3.1). However, there are K^n of those integrals and the problem remains intractable.

Variational Inference (VI) (Hinton and Van Camp, 1993; Waterhouse et al., 1996; Jordan et al., 1999) deals with these kind of problems by approximating the posterior distribution with a *variational distribution*, which should be flexible enough to approximate the true posterior. That is, given a vector of observable variables $\mathbf{x} = (x_1, x_2, \dots, x_{\ell_x})^T$ and a vector of latent variable $\mathbf{z} = (z_1, z_2, \dots, z_{\ell_z})^T$, VI approximates the posterior distribution

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}, \quad (3.17)$$

where $p(\mathbf{z})$ is the prior distribution of the latent variable \mathbf{z} , $p(\mathbf{x}|\mathbf{z})$ is the likelihood function, and $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z})d\mathbf{z}$ is the marginal distribution of \mathbf{x} or the *evidence* that is assumed to be intractable, with a parametric variational distribution $q(\mathbf{z}|\mathbf{x}; \boldsymbol{\phi})$ where $\boldsymbol{\phi}$ is a vector of variational parameters. For simplicity and when it is possible, we replace the formal, but cluttered, notation $q(\mathbf{z}|\mathbf{x}; \boldsymbol{\phi})$ for the shorthand version $q(\mathbf{z}|\mathbf{x})$.

The common way to measure the quality of approximating the true posterior distribution $p(\mathbf{z}|\mathbf{x})$ with $q(\mathbf{z}|\mathbf{x})$ is using the Kullback-Leibler (KL) divergence

$$\begin{aligned} KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})] &= - \int q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\ &= \mathbb{E}_q[\log q(\mathbf{z}|\mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{z}|\mathbf{x})]. \end{aligned} \quad (3.18)$$

Note that the KL divergence is strictly non-negative given that

$$\begin{aligned} KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})] &= - \mathbb{E}_q \left[\log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} \right] \\ &\geq - \log \left[\mathbb{E}_q \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} \right] \\ &= - \log \left[\int q(\mathbf{z}|\mathbf{x}) \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z} \right] \\ &= 0, \end{aligned} \quad (3.19)$$

where the inequality is a result of the concavity of \log and Jensen's inequality. Therefore, the KL divergence term is often referred to as the *distance* between two densities and it is minimized when $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})$.

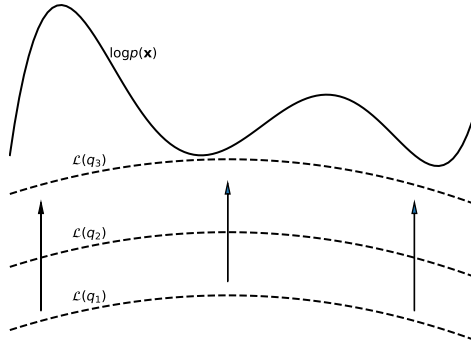


Figure 3.3: Graphical representation of the variational inference approximation. The lower bound $\mathcal{L}(q)$ on the intractable density $\log p(\mathbf{x})$ is optimized in such a way that the divergence $KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})]$ is minimized. This is achieved by tuning the variational parameters ϕ so that the gap between $\mathcal{L}(q)$ and $\log p(\mathbf{x})$, which is referred as the tightness of the lower bound, is minimized.

If it would be possible, we could find the *best* variational distribution by minimizing the KL divergence term in Equation 3.18. However, using the Bayes' Theorem in the second term of Equation 3.18 we see that

$$\begin{aligned} KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})] &= \mathbb{E}_q[\log q(\mathbf{z}|\mathbf{x})] - \mathbb{E}_q\left[\log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}\right] \\ &= \mathbb{E}_q[\log q(\mathbf{z}|\mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z})] + \log p(\mathbf{x}) \end{aligned} \quad (3.20)$$

contains the intractable evidence $p(\mathbf{x})$.

To circumvent this problem VI maximizes the *lower bound*

$$\begin{aligned} \log p(\mathbf{x}) &= \mathbb{E}_q\left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right] + KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})] \\ &\geq \mathbb{E}_q\left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right] \\ &\equiv \mathcal{L}(q), \end{aligned} \quad (3.21)$$

where the inequality is a result of the non-negative KL divergence (see Equation 3.19) and $\mathcal{L}(q)$ is the lower bound on $\log p(\mathbf{x})$. Note that maximizing

the lower bound in Equation 3.21 makes the approximation $q(\mathbf{z}|\mathbf{x})$ closer to the true posterior distribution $p(\mathbf{z}|\mathbf{x})$. Additionally, the gap between the lower bound and $\log p(\mathbf{x})$ decreases, which is referred to as the *tightness* of the lower bound. This is illustrated in Figure 3.3

Noting that variational densities are characterized by their variational parameters, Blei et al. (2017) claim that VI is an optimization problem over the family of variational densities $q(\mathbf{z}|\mathbf{x}; \phi) \in \mathcal{Q}$ and the variational parameters ϕ are tuned to minimize

$$q^*(\mathbf{z}|\mathbf{x}) = \arg \min_{q(\mathbf{z}|\mathbf{x}) \in \mathcal{Q}} KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})]. \quad (3.22)$$

It is also possible to use Jensen’s inequality to arrive at the lower bound in Equation 3.21 as follows

$$\begin{aligned} \log p(\mathbf{x}) &= \log \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\ &= \log \mathbb{E}_q \left[\frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right] \\ &\geq \mathbb{E}_q \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right], \end{aligned} \quad (3.23)$$

where the inequality is a result of the concavity of \log and Jensen’s inequality.

For completeness, in the following sections we review some of the common VI methods, e.g. mean field approximation, stochastic variational inference, black-box variational inference, and amortized inference. Table 3.1 shows a quick overview over the methods presented in the following sections. Note that we only cover methods where the variational distribution is fully factorized, i.e. $p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n) = p(\mathbf{x}_1)p(\mathbf{x}_2)\dots p(\mathbf{x}_n)$, and where the KL divergence is used as measure for the distance between two densities. The interested reader is referred to Wainwright et al. (2008), Blei et al. (2017) and Zhang et al. (2018a) for deeper reviews on VI methods.

3.3.1 Mean Field Approximation

Before we start explaining the mean field approximation, we introduce some standard notation. Given a set of observable variables $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ and a set of latent variables $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m\}$, we use the short notation

	Characteristics			
	Scalability	Conjugate	Non-conjugate	Amortized Inference
Mean Field VI :		✓		
Stochastic VI:	✓	✓		
Black-Box VI:			✓	
Amortized VI:	✓		✓	✓

Table 3.1: Methods for Variational Inference (VI)

\mathbb{E}_{q_j} to refer to the expectation $\mathbb{E}[q(\mathbf{z}_j)]$; hence, the subscript j refers to the j 'th variable and \mathbf{z}_{-j} refers to all variables in \mathbf{Z} but z_j . Similarly, \mathbb{E}_{q_j} is the expectation with respect to \mathbf{z}_j and $\mathbb{E}_{q_{-j}}$ is the expectation with respect to all variables but \mathbf{z}_j .

The mean field approximation assumes that \mathcal{Q} is in the class of fully factorized distributions, e.g.

$$q(\mathbf{Z}; \phi_i) = \prod_i^m q_i(\mathbf{z}_i; \phi_i), \quad (3.24)$$

where ϕ_i are the variational parameters for the i 'th latent variable. This sort of fully factorized distributions can be optimized efficiently, as we will show shortly, but in some cases it might be too restricted to approximate the true posterior distribution accurately. The main idea behind the mean field approximation is to optimize each \mathbf{z}_j one at a time while holding the others fixed. This iterative procedure is called coordinate ascent.

From Equation 3.20 and 3.21 we can express the lower bound over a data set as

$$\mathcal{L}(q) = \log p(\mathbf{X}) + \mathbb{E}_q[\log p(\mathbf{Z}|\mathbf{X})] - \mathbb{E}_q[\log q(\mathbf{Z})], \quad (3.25)$$

and to find the optimal q_j^* , while holding q_{-j} fixed, it is useful to use the chain rule of conditional probabilities (see Section 3.1) and the fact that

$\mathbb{E}_q[\log q] = \sum_{i=1}^m \mathbb{E}_{q_i}[\log q_i]$. Further, we can pull out terms that do not depend on q_j . Hence,

$$\mathcal{L}(q) = \sum_{i=1}^m \mathbb{E}_q[\log p(\mathbf{z}_i | \mathbf{z}_{-i}, \mathbf{X})] - \mathbb{E}_q[\log q] + \log p(\mathbf{X})$$

and

$$\begin{aligned} \mathcal{L}(q_j) &\propto \int \cdots \int q \log p(\mathbf{z}_j | \mathbf{z}_{-j}, \mathbf{X}) d\mathbf{z}_1 \cdots d\mathbf{z}_m - \mathbb{E}_{q_j}[\log q_j] \\ &= \int q_j \int \cdots \int q_{-j} \log p(\mathbf{z}_j | \mathbf{z}_{-j}, \mathbf{X}) d\mathbf{z}_{-j} d\mathbf{z}_j - \mathbb{E}_{q_j}[\log q_j] \\ &= \mathbb{E}_{q_j}[\mathbb{E}_{q_{-j}}[\log p(\mathbf{z}_j | \mathbf{z}_{-j}, \mathbf{X})]] - \mathbb{E}_{q_j}[\log q_j] \\ &= \mathbb{E}_{q_j}[\log(\exp\{\mathbb{E}_{q_{-j}}[\log p(\mathbf{z}_j | \mathbf{z}_{-j}, \mathbf{X})]\})] - \mathbb{E}_{q_j}[\log q_j] \\ &= -KL[q_j || \exp\{\mathbb{E}_{q_{-j}}[\log p(\mathbf{z}_j | \mathbf{z}_{-j}, \mathbf{X})]\}]. \end{aligned} \quad (3.26)$$

As pointed out in Section 3.3, the divergence $KL[q(\cdot) || p(\cdot)]$ is minimized when $q(\cdot) = p(\cdot)$. Hence, the mean field update is

$$q_j^* \propto \exp\{\mathbb{E}_{q_{-j}}[\log p(\mathbf{z}_j | \mathbf{z}_{-j}, \mathbf{x})]\}. \quad (3.27)$$

The updating equation in Equation 3.27 computes the expectation (hence the name *mean field*) of q_{-j} at each iteration, which is often referred to as coordinate ascent variational inference (CAVI) (Bishop, 2006). Note that we have only specified the factorization of the model but not the family of distributions and for some models the expectation $\mathbb{E}_{q_{-j}}$ may not be easy to compute.

3.3.2 Stochastic Variational Inference

The mean field approximation presented in the previous section has two major drawbacks. First, it scales poorly to large data sets since it requires to analyze the whole data set before updating the variational parameters. Second, it is restricted to models where the expectation in Equation 3.27 has a closed form.

Hoffman et al. (2013) develop a stochastic variational inference (SVI) framework to deal with large data sets in the context of VI. Their work builds

up on previous developments in stochastic optimization where the objective function is optimized using unbiased estimates of the gradient.

To present the stochastic optimization algorithm introduced by Hoffman et al. (2013) we assume a data set of n observable variables \mathbf{X} with n local hidden variables \mathbf{Z} . Each global parameter $\boldsymbol{\lambda}$ govern global variables $\boldsymbol{\beta}$, and local parameters ϕ_i governs the i 'th latent variable z_i . Assuming that the variational parameters belong to the exponential family we have

$$q(\boldsymbol{\beta}|\boldsymbol{\lambda}) = h(\boldsymbol{\beta}) \exp\{\boldsymbol{\lambda}^T t(\boldsymbol{\beta}) - a_g(\boldsymbol{\lambda})\} \quad (3.28)$$

and

$$q(z_{i,j}|\phi_{i,j}) = h(z_{i,j}) \exp\{\phi_{i,j}^T t(z_{i,j}) - a_\ell(\phi_{i,j})\}, \quad (3.29)$$

where $z_{i,j}$ is the j 'th variable in the i 'th observation, $h(\cdot)$, $a(\cdot)$, and $t(\cdot)$ are the the base measure, log-normalizer, and sufficient statistics functions respectively.

Their mean field updates are

$$\boldsymbol{\lambda}^* = \mathbb{E}_q[\eta_g(\mathbf{X}, \mathbf{Z}, \boldsymbol{\alpha})] \quad (3.30)$$

and

$$\phi_{i,j}^* = \mathbb{E}_q[\eta_\ell(x_n, z_{n,-j}, \boldsymbol{\beta})], \quad (3.31)$$

where η_g and η_ℓ are the natural parameter function of the global and local variational parameters respectively.

Equations 3.30 and 3.31 are the updates needed in the coordinate ascent algorithm of Section 3.3.1. However, CAVI is inefficient for large data sets since all local variational parameters must be optimized before updating global parameters. The method proposed by Hoffman et al. (2013) uses mini-batches \mathbf{X}^m of m randomly drawn data points from the full data set \mathbf{X} . Hence, the objective function is approximated based on mini-batches as follows

$$\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\phi}; \mathbf{X}) \approx \hat{\mathcal{L}}^m(\boldsymbol{\lambda}, \boldsymbol{\phi}; \mathbf{X}) = \frac{n}{m} \sum_{i=1}^m \hat{\mathcal{L}}(\boldsymbol{\lambda}, \boldsymbol{\phi}; \mathbf{x}_i). \quad (3.32)$$

Finally, the gradients of the mini-batch approximation $\nabla \hat{\mathcal{L}}^m(\boldsymbol{\lambda}, \boldsymbol{\phi}; \mathbf{X})$ are replaced by the *natural gradients* (Amari, 1998) $\tilde{\nabla} \hat{\mathcal{L}}^m(\boldsymbol{\lambda}, \boldsymbol{\phi}; \mathbf{X})$. Natural gradients consider the geometry of its parameter space, improving convergence compared to standard gradients (Hoffman et al., 2013; Blei et al., 2017).

3.3.3 Non-conjugate Variational Inference

SVI is an efficient method that handles large data sets as the ones used in Hoffman et al. (2013). However, it is restricted to conjugate models where the lower bound can be derived analytically. In this section we introduce black-box variational inference (BBVI) (Ranganath et al., 2014), which is a flexible method that allows to analyze a wider range of models.

The main idea behind BBVI is to find a representation for the gradient of the lower bound $\nabla \mathcal{L}(q)$ as an expectation of the gradient with respect to q , i.e. $\mathbb{E}_q[\nabla f(\mathcal{L}(q))]$. These sort of gradients are often referred as REINFORCE or score gradients.

To derive the REINFORCE gradients in Ranganath et al. (2014) we take the dominated convergence theorem

$$\nabla_{\phi} \int f(x; \phi) dx = \int \nabla_{\phi} f(x; \phi) dx \quad (3.33)$$

as granted (Çınlar, 2011). Further, we use the log-derivative identity

$$\nabla_{\phi} q(\mathbf{z}|\mathbf{x}) = \nabla_{\phi} [\log q(\mathbf{z}|\mathbf{x})] q(\mathbf{z}|\mathbf{x}), \quad (3.34)$$

and we define the function $g(\mathbf{z}, \mathbf{x}) \equiv \log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})$. Then, the gradient of the lower bound with respect to the variational parameters can be written as

$$\begin{aligned} \nabla_{\phi} \mathcal{L}(q) &= \nabla_{\phi} \int q(\mathbf{z}|\mathbf{x}) g(\mathbf{z}, \mathbf{x}) d\mathbf{z} \\ &= \int \nabla_{\phi} q(\mathbf{z}|\mathbf{x}) g(\mathbf{z}, \mathbf{x}) d\mathbf{z} \\ &= \int \nabla_{\phi} [q(\mathbf{z}|\mathbf{x})] g(\mathbf{z}, \mathbf{x}) d\mathbf{z} + \int \nabla_{\phi} [g(\mathbf{z}, \mathbf{x})] q(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\ &= \int \nabla_{\phi} [\log q(\mathbf{z}|\mathbf{x})] g(\mathbf{z}, \mathbf{x}) q(\mathbf{z}|\mathbf{x}) d\mathbf{z} + \int \nabla_{\phi} [\log p(\mathbf{x}, \mathbf{z}) \\ &\quad - \log q(\mathbf{z}|\mathbf{x})] q(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\ &= \mathbb{E}_q[\nabla_{\phi} [\log q(\mathbf{z}|\mathbf{x})] g(\mathbf{z}, \mathbf{x})] - \mathbb{E}_q[\nabla_{\phi} \log q(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_q[\nabla_{\phi} [\log q(\mathbf{z}|\mathbf{x})] (\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x}))], \end{aligned} \quad (3.35)$$

where $\nabla_\phi[\log q(\mathbf{z}|\mathbf{x})]$ is called the *score function* and its expectation is

$$\begin{aligned}\mathbb{E}_q[\nabla_\phi \log q(\mathbf{z}|\mathbf{x})] &= \int q(\mathbf{z}|\mathbf{x}) \frac{\nabla_\phi q(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\ &= \nabla_\phi \int q(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\ &= 0.\end{aligned}$$

Finally, Equation 3.35 is approximated with the Monte Carlo expectation

$$\nabla_\phi \mathcal{L}(q) \approx \frac{1}{L} \sum_{l=1}^L \nabla_\phi \log q(\mathbf{z}_l|\mathbf{X}) (\log p(\mathbf{X}, \mathbf{z}_l) - \log q(\mathbf{z}_l|\mathbf{X})), \quad (3.36)$$

where $\mathbf{z}_l \sim q(\mathbf{z}|\mathbf{x})$. The noise score gradient in Equation 3.36 is used for stochastic optimization. Unfortunately, this gradient suffers from high variance, which can lead to slow and poor convergence. Ranganath et al. (2014) suggest to use Rao-Blackwellization and Control Variates to reduce the variance of the approximation of the gradient in Equation 3.36.

The main advantage of BBVI is that the score function and the sampling scheme depends only on the variational distributions. Hence, it should be possible to pre-program a collection of variational distributions and their score functions and reuse them in a package where the only distribution that needs to be specified is $\log p(\mathbf{x}, \mathbf{z})$.

3.3.4 Amortized Inference

Another method that scales to large data sets was introduced by Kingma and Welling (2013) and Rezende et al. (2014), which developed an efficient framework for VI where variational and generative parameters are shared across n data points. This variational inference method is often referred to as *amortized inference*. Consider we have n observable and latent variables $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ and $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n\}$ respectively, and that the variational distribution takes the form of a fully factorized distribution

$$q(\mathbf{Z}|\mathbf{X}; \phi) = \prod_{i=1}^n q(\mathbf{z}_i|\mathbf{x}_i; \phi). \quad (3.37)$$

Note that ϕ is not dependent on the i 'th latent data point as in Equation 3.24 and that under the amortized inference approach the optimal \mathbf{z}_i is inferred

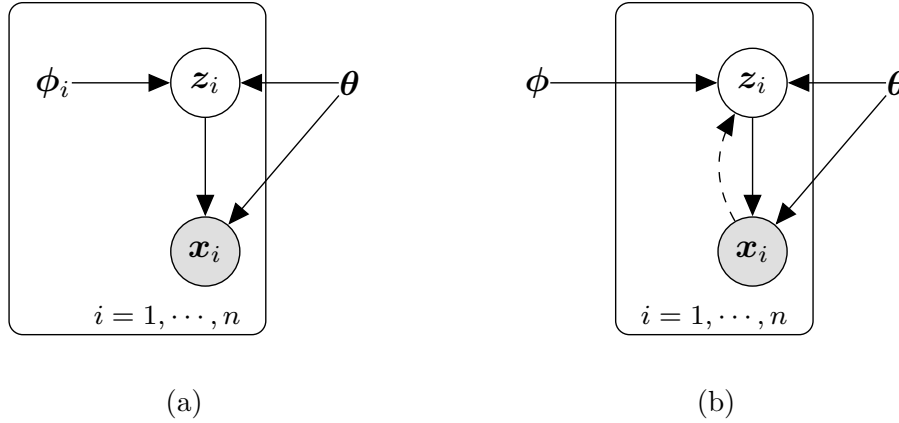


Figure 3.4: Plate notation of the mean field approximation method (left panel) and Variational Autoencoder (right panel). In the mean field approximation the variational parameter ϕ_i needs to be optimized for each data point \mathbf{x}_i , while in the amortized variational inference approach ϕ is shared across n data points.

based on \mathbf{x}_i as follows

$$z_i | \mathbf{x}_i \sim q(z_i | \mathbf{x}_i; \boldsymbol{\eta}_z = f_\phi(\mathbf{x}_i)), \quad (3.38)$$

where $\boldsymbol{\eta}_z$ are the parameters of the $q(\cdot)$ density function and $f(\cdot)$ is a neural network with learnable parameters ϕ . That is, a neural network is used as a powerful prediction function whose parameters ϕ are shared across n data points (Zhang et al., 2018a). This is the main idea behind amortized inference and it is shown in Figure 3.4.

Similarly, the generative process is

$$\mathbf{x}_i | z_i \sim p(\mathbf{x}_i | z_i; \boldsymbol{\eta}_x = f_\theta(z_i)), \quad (3.39)$$

where $f(\cdot)$ is a neural network with parameters θ and $\boldsymbol{\eta}_x$ are the parameters of the $p(\cdot)$ density function.

The idea of amortized inference has become very popular and coupled with variational inference it offers a powerful modeling framework. We will explain such a framework in more detail in the following chapter.

Chapter 4

Deep Generative Models

In the previous chapter we introduced probabilistic graphical models that use latent variables. Further, we presented the concept of variational inference together with different methods for variational inference, e.g. amortized inference. In this chapter, we formalize the notion of amortized inference by explaining a model called Variational Autoencoder (VAE) (Kingma and Welling, 2013). The VAE is a concrete example of a deep generative model, which uses latent variables, and it exemplifies the main aspects in more complex deep generative models. Further, we show that deep generative models meet the last criteria for being a useful probabilistic model according to Koller and Friedman (2009), which is *learning*.

Therefore, the focus of this chapter is only on directed graphical models where the distributions' parameters are parameterized by (deep) neural networks. For a review about other deep generative models, e.g. Boltzmann Machines, Deep Belief Networks, Generative Adversarial Networks etc. the reader is referred to Goodfellow et al. (2016).

4.1 Variational Autoencoder

VAEs assume that for each observation $\mathbf{x} = (x_1, x_2, \dots, x_{d_x}) \in \mathbb{R}^{d_x}$ there is a latent variable $\mathbf{z} = (z_1, z_2, \dots, z_{d_z}) \in \mathbb{R}^{d_z}$ that it is drawn from a prior distribution

$$p(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{1}). \quad (4.1)$$

Further, \mathbf{x} is assumed to be generated by the conditional distribution $p(\mathbf{x}|\mathbf{z})$, which we assume follows a Gaussian distribution

$$p(\mathbf{x}|\mathbf{z}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad (4.2)$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_{d_x})$ and $\boldsymbol{\Sigma}$ is a $d_x \times d_x$ diagonal covariance matrix with main diagonal $\boldsymbol{\sigma}^2 = (\sigma_1^2, \sigma_2^2, \dots, \sigma_{d_x}^2)$.

Therefore, the generative process in a VAE is $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ and, assuming the framework introduced in Section 3.3, it optimizes the variational lower bound in Equation 3.21

$$\mathbb{E}_q \left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right] = \mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z})] - KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})], \quad (4.3)$$

where $q(\mathbf{z}|\mathbf{x})$ is the variational distribution and we assume it is Gaussian distributed, i.e.

$$q(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad (4.4)$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_{d_z})$ and $\boldsymbol{\Sigma}$ is a $d_z \times d_z$ diagonal covariance matrix with main diagonal $\boldsymbol{\sigma}^2 = (\sigma_1^2, \sigma_2^2, \dots, \sigma_{d_z}^2)$.

As explained in Section 3.3.4, amortized inference uses neural networks as prediction functions to learn the density parameters in the generative and inference process. Specifically, we use multilayer perceptrons (MLPs)¹ to learn the density parameters in Equation 4.2 and 4.4, that is

$$p(\mathbf{x}|\mathbf{z}) \sim \mathcal{N}(\mathbf{x}|\mathbf{z}; \boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}} = f_{\boldsymbol{\theta}}(\mathbf{z}), \boldsymbol{\sigma}_{\mathbf{x}|\mathbf{z}}^2 = f_{\boldsymbol{\theta}}(\mathbf{z})), \quad (4.5)$$

and

$$q(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}(\mathbf{z}|\mathbf{x}; \boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}} = f_{\boldsymbol{\phi}}(\mathbf{x}), \boldsymbol{\sigma}_{\mathbf{z}|\mathbf{x}}^2 = f_{\boldsymbol{\phi}}(\mathbf{x})), \quad (4.6)$$

where $f_{\boldsymbol{\phi}}(\mathbf{x})$ and $f_{\boldsymbol{\theta}}(\mathbf{z})$ are MLPs with trainable parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, $\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}}$ and $\boldsymbol{\sigma}_{\mathbf{z}|\mathbf{x}}^2$ are defined as in Equation 4.2, and $\boldsymbol{\sigma}_{\mathbf{x}|\mathbf{z}}^2$ and $\boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}}$ are defined as in Equation 4.4.

Assuming that the MLPs parametrizing Equation 4.5 and 4.6 have 3 layers (input, hidden, and output layer) and that we have a data set $\mathbf{X} =$

¹Given that we use vectorized data in the research conducted in this thesis, the neural networks that we use are multilayer perceptrons. MLPs and the backpropagation algorithm are presented in Appendix A.

$\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, the reconstruction of the i 'th vector \mathbf{x}_i and its latent representation \mathbf{z}_i are given by

$$\begin{aligned} \mathbf{h} &= \mathbf{a}(\mathbf{W}_2 \mathbf{x}_i + \mathbf{b}_2), & \mathbf{h} &= \mathbf{a}(\mathbf{W}_2 \mathbf{z}_i + \mathbf{b}_2), \\ \boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}} &= \mathbf{w}_1^L \mathbf{h} + b_1, & \boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}} &= \mathbf{w}_1 \mathbf{h} + b_1, \\ \boldsymbol{\sigma}_{\mathbf{z}|\mathbf{x}}^2 &= \mathbf{w}_2^L \mathbf{h} + b_2, & \boldsymbol{\sigma}_{\mathbf{x}|\mathbf{z}}^2 &= \mathbf{w}_2 \mathbf{h} + b_2, \\ \mathbf{z}_i &= \boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}} + \boldsymbol{\sigma}_{\mathbf{z}|\mathbf{x}} \odot \boldsymbol{\epsilon}, & \hat{\mathbf{x}}_i &= \boldsymbol{\mu}_{\mathbf{x}|\mathbf{z}} + \boldsymbol{\sigma}_{\mathbf{x}|\mathbf{z}} \odot \boldsymbol{\epsilon}, \end{aligned} \quad (4.7)$$

where $\mathbf{w}_i^L = (w_{i,1}^L, \dots, w_{i,L-1}^L)^T$ is the weight vector of the i 'th neuron at the output layer $r = L$ for $i = 1, 2$, $\mathbf{W}_2 = \{\mathbf{w}_1^2, \mathbf{w}_2^2, \dots, \mathbf{w}_{k_r}^2\}$ is the set of k_r weight vectors in the $r = 2$ layer and $\mathbf{b}_2 = (b_1^2, \dots, b_{k_r}^2)^T$ is a vector of k_r bias terms for $r = 2$, \mathbf{a} is an activation function, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and \odot is the element-wise product. Therefore, $\boldsymbol{\theta} = \{\mathbf{W}_2, \mathbf{w}_1^L, \mathbf{w}_2^L, \mathbf{b}_2, b_1, b_2\}$ and $\boldsymbol{\phi} = \{\mathbf{W}_2, \mathbf{w}_1^L, \mathbf{w}_2^L, \mathbf{b}_2, b_1, b_2\}$ are the learnable parameters for the MLP parametrizing Equation 4.5 and 4.6 respectively. The last lines² in Equation 4.7 are often referred to as the *reparameterization trick*, which we explain in Section 4.3.

4.1.1 Connection with autoencoders

Note that Equation 4.4 takes the input data \mathbf{x} and generates a latent representation or a code \mathbf{z} . Then, Equation 4.2 takes the code \mathbf{z} and reconstructs the input data \mathbf{x} . Therefore $q(\mathbf{z}|\mathbf{x})$ is often referred to as a probabilistic encoder and $p(\mathbf{x}|\mathbf{z})$ as a probabilistic decoder. The mechanism encode-decode is the same as in autoencoders (AEs) neural networks. However, AEs are deterministic neural networks trained to reconstruct the input data as close as possible while VAEs assume probability density functions and are trained to maximize the variational lower bound. Figure 4.1 shows the architecture of AEs and VAEs to highlight their similarities.

4.1.2 Generative properties

VAEs, and DGMs in general, have a generative process $p(\mathbf{x}|\mathbf{z})$. During training, the term $\mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z})]$ in Equation 4.3 is maximized by putting emphasis on latent variables \mathbf{z} that help to reconstruct \mathbf{x} . After training, we can use the MLP parametrizing $p(\mathbf{x}|\mathbf{z})$ to generate \mathbf{x} using its latent

²In practice, it is common that the output layer of the MLP parametrizes $\log \boldsymbol{\sigma}^2$ to avoid negative variances. If this is case, then $\boldsymbol{\sigma} = \sqrt{\exp(\log \boldsymbol{\sigma}^2)}$.

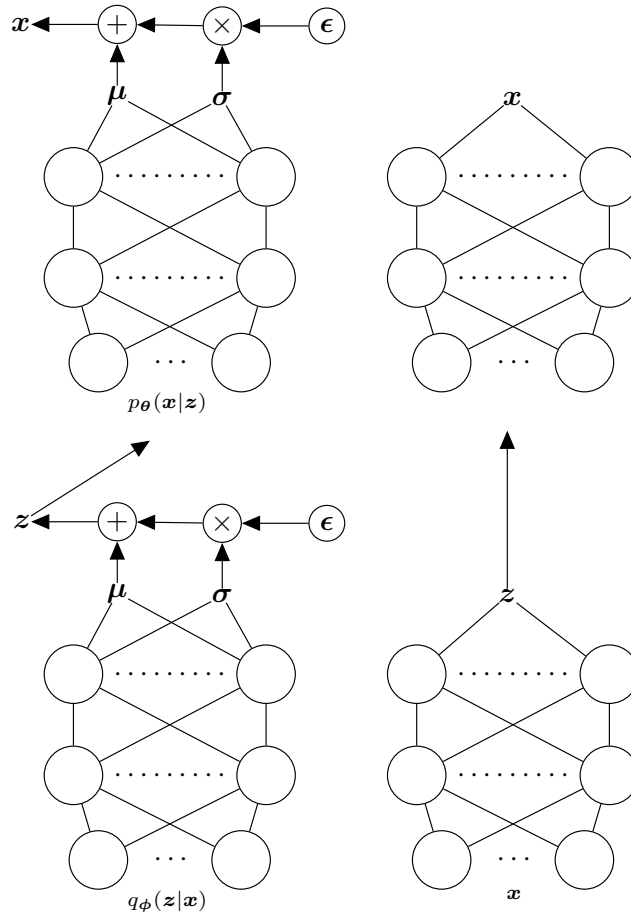


Figure 4.1: Graphical representation of VAE (left side) and AE (right side). Both VAE and AE, with two hidden layers, take as input a data vector \mathbf{x} and generates a code \mathbf{z} , which is the input to reconstruct the original input vector \mathbf{x} . The difference between VAE and AE is that AE uses deterministic neural networks, while VAE assumes probability density functions for \mathbf{x} and \mathbf{z} , which are parametrized by neural networks.

representation \mathbf{z} , see Equation 4.7. Note that it is not necessary to observe \mathbf{x} to draw \mathbf{z} from the posterior $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$. It is possible to directly draw samples $\tilde{\mathbf{z}}$ from the space \mathcal{Z} and use those to generate $\mathbf{x} \sim p(\mathbf{x}|\tilde{\mathbf{z}})$, or even do arithmetic operations in the space \mathcal{Z} to steer the generative process, e.g. Su et al. (2018); Hsu et al. (2017).

4.2 Deriving the Lower Bound

Expanding the objective function in Equation 4.3, we see that the expectation

$$\mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})] \quad (4.8)$$

is composed by the integrals

$$\int q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}|\mathbf{z}) d\mathbf{z} = \int \mathcal{N}(\mathbf{z}|\mathbf{x}) \log \mathcal{N}(\mathbf{x}|\mathbf{z}) d\mathbf{z}, \quad (4.9)$$

$$\int q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{z}) d\mathbf{z} = \int \mathcal{N}(\mathbf{z}|\mathbf{x}) \log \mathcal{N}(\mathbf{z}) d\mathbf{z}, \quad (4.10)$$

and

$$- \int q(\mathbf{z}|\mathbf{x}) \log q(\mathbf{z}|\mathbf{x}) d\mathbf{z} = - \int \mathcal{N}(\mathbf{z}|\mathbf{x}) \log \mathcal{N}(\mathbf{z}|\mathbf{x}) d\mathbf{z}. \quad (4.11)$$

The integrals in Equation 4.10 and 4.11 can be solved analytically using the following lemma. An incomplete proof of the lemma can be found in Kingma and Welling (2013) and a full proof is given by Zheng et al. (2016), but for the benefit of the reader we provide a different proof.

Lemma 1 *Given two multivariate Gaussian distribution $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $q(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, where $\mathbf{x} \in \mathbb{R}^{d_x}$, $\boldsymbol{\mu}_i = (\mu_{1,1}, \mu_{1,2}, \dots, \mu_{1,d_x})$ and $\boldsymbol{\Sigma}_i$ is a $d_x \times d_x$ diagonal covariance matrix with main diagonal $\boldsymbol{\sigma}_i^2 = (\sigma_{1,1}^2, \sigma_{1,2}^2, \dots, \sigma_{1,d_x}^2)$ for $i = 1, 2$, we have:*

$$\int q(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} = \sum_{j=1}^{d_x} -\frac{1}{2} \log(2\pi\sigma_{1,j}^2) - \frac{\sigma_{2,j}^2}{2\sigma_{1,j}^2} - \frac{(\mu_{2,j} - \mu_{1,j})^2}{2\sigma_{1,j}^2}, \quad (4.12)$$

where $\mu_{j,i}$ and $\sigma_{j,i}$ are the j 'th element of their respective $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i^2$ parameter for $i = 1, 2$.

Proof:

$$\begin{aligned}
\int q(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} &= \int q(\mathbf{x}) \log \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_1|^{1/2}} \\
&\quad \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1)\right) d\mathbf{x} \\
&= -\frac{1}{2} \log(2\pi\sigma_{1,i}^2) - \int q(\mathbf{x}) \frac{(x_i - \mu_{1,i})^2}{2\sigma_{1,i}^2} d\mathbf{x} - \dots \\
&\quad \dots - \frac{1}{2} \log(2\pi\sigma_{1,d}^2) - \int q(\mathbf{x}) \frac{(x_d - \mu_{1,d})^2}{2\sigma_{1,d}^2} d\mathbf{x} \\
&= -\frac{1}{2} \log(2\pi\sigma_{1,i}^2) - \frac{\mathbb{E}_q[x_i^2] - 2\mathbb{E}_q[x_i]\mu_{1,i} + \mu_{1,i}^2}{2\sigma_{1,i}^2} - \dots \\
&\quad \dots - \frac{1}{2} \log(2\pi\sigma_{1,d}^2) - \frac{\mathbb{E}_q[x_d^2] - 2\mathbb{E}_q[x_d]\mu_{1,d} + \mu_{1,d}^2}{2\sigma_{1,d}^2} \\
&= -\frac{1}{2} \log(2\pi\sigma_{1,i}^2) - \frac{\sigma_{2,i}^2 + \mu_{2,i}^2 - 2\mu_{2,i}\mu_{1,i} + \mu_{1,i}^2}{2\sigma_{1,i}^2} - \dots \\
&\quad \dots - \frac{1}{2} \log(2\pi\sigma_{1,d}^2) - \frac{\sigma_{2,d}^2 + \mu_{2,d}^2 - 2\mu_{2,d}\mu_{1,d} + \mu_{1,d}^2}{2\sigma_{1,d}^2} \\
&= -\frac{1}{2} \log(2\pi\sigma_{1,i}^2) - \frac{\sigma_{2,i}^2 + (\mu_{2,i} - \mu_{1,i})^2}{2\sigma_{1,i}^2} - \dots \\
&\quad \dots - \frac{1}{2} \log(2\pi\sigma_{1,d}^2) - \frac{\sigma_{2,d}^2 + (\mu_{2,d} - \mu_{1,d})^2}{2\sigma_{1,d}^2} \\
&= \sum_j^{d_x} -\frac{1}{2} \log(2\pi\sigma_{1,j}^2) - \frac{\sigma_{2,j}^2}{2\sigma_{1,j}^2} - \frac{(\mu_{2,j} - \mu_{1,j})^2}{2\sigma_{1,j}^2}. \tag{4.13}
\end{aligned}$$

Hence, Equation 4.10 is

$$\int q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{z}|\mathbf{x}) d\mathbf{z} = -\frac{d_z}{2} \log 2\pi - \frac{1}{2} \sum_{j=1}^{d_z} (\sigma_{j,q}^2 + \mu_{j,q}),$$

and Equation 4.11

$$\int q(\mathbf{z}|\mathbf{x}) \log q(\mathbf{z}|\mathbf{x}) d\mathbf{z} = -\frac{d_z}{2} \log 2\pi - \frac{1}{2} \sum_{j=1}^{d_z} (1 + \sigma_{j,q}^2).$$

Finally, taking the Monte Carlo expectation of Equation 4.9 by drawing L samples from $q(\mathbf{z}|\mathbf{x})$ the lower bound for the i 'th observation is

$$\mathcal{L}(\mathbf{x}_i) = \frac{1}{L} \sum_{l=1}^L \log \mathcal{N}(\mathbf{x}_i | \mathbf{z}_{i,l}) + \frac{1}{2} \sum_{j=1}^{d_z} (1 + \sigma_{j,q}^2 - \sigma_{j,q}^2 - \mu_{j,q}), \quad (4.14)$$

where d_z is the dimension of \mathbf{z} .

4.3 The Reparameterization Trick

The motivation for the reparameterization is twofold. First, we need a practical and efficient estimate for the gradient of the lower bound. Second, as shown in Section 4.2, it turns out that we need to draw samples from $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$ to approximate an expectation in the lower bound, see Equation 4.14. Hence, the reparameterization of \mathbf{z} as a deterministic function makes the randomness to *lie* outside the backpropagation path. As shown in the appendix, the backpropagation algorithm propagates backwards using the chain rule of differentiation. Therefore, the reparameterization trick converts the MLP into a deterministic function that incorporates a random component outside the iterative procedure of chain rule of differentiation.

4.3.1 Reparameterization Gradients

The *reparameterization trick* and the equality $q(\mathbf{z}|\mathbf{x})d\mathbf{z} = p(\boldsymbol{\epsilon})d\boldsymbol{\epsilon}$ used in Kingma and Welling (2013) is a consequence of the *change of variable* technique, which we introduce in this section. For the convenience of the reader we present a full proof here. Murphy (2012), for example, presents a different view of the proof.

Lemma 2 *Let $X \in \mathbb{R}^\ell$ be a continuous vector-valued random variable with probability density $f_X(x)$ and cumulative function $F_X(x)$. Further, let $Y = u(X)$ be an increasing and invertible function of X with inverse function $X = v(Y)$. Then, by the change of variable technique the probability density function of Y is*

$$\begin{aligned} f_Y(y) &= f_X(v(y)) \times \left| \frac{\partial v(y)}{\partial y} \right| \\ &= f_X(x) \times \left| \frac{\partial x}{\partial y} \right|, \end{aligned} \quad (4.15)$$

where $|\cdot|$ is the absolute value.

Proof:

$$\begin{aligned}
 F_Y(y) &= \Pr(Y \leq y) \\
 &= \Pr(u(X) \leq y) \\
 &= \Pr(X \leq v(y)) \\
 &= F_X(v(y)),
 \end{aligned} \tag{4.16}$$

so,

$$\begin{aligned}
 f_Y(y) &= \frac{\partial F_Y(y)}{\partial y} \\
 &= \frac{\partial F_X(v(y))}{\partial y} \\
 &= F'_X(v(y))v'(y) \\
 &= f_X(v(y))v'(y),
 \end{aligned} \tag{4.17}$$

where $v'(y)$ is the (square) Jacobian matrix

$$J(y) = \begin{bmatrix} \frac{\partial v(y_1)}{\partial y_1} & \cdots & \frac{\partial v(y_1)}{\partial y_\ell} \\ \vdots & \ddots & \vdots \\ \frac{\partial v(y_\ell)}{\partial y_1} & \cdots & \frac{\partial v(y_\ell)}{\partial y_\ell} \end{bmatrix}. \tag{4.18}$$

If $u(X)$ is strictly decreasing, Equation 4.16 is $F_Y(y) = 1 - F_X(v(y))$ and Equation 4.17 becomes $f_Y(y) = -f_X(v(y))v'(y)$. Hence, $|v'(v)|$ generalizes both cases.

Kingma and Welling (2013) use the invertible function

$$z = g(\epsilon), \tag{4.19}$$

where

$$g_\phi(\epsilon) = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4.20}$$

to re-express $q(\mathbf{z}|\mathbf{x})d\mathbf{z}$ as $p(\boldsymbol{\epsilon})d\boldsymbol{\epsilon}$, which is a result of Lemma 2. This implies that

$$\begin{aligned}\int q(\mathbf{z}|\mathbf{x})f(\mathbf{z})d\mathbf{z} &= \int p(\boldsymbol{\epsilon})f(\mathbf{z})d\boldsymbol{\epsilon} \\ &= \int p(\boldsymbol{\epsilon})f(g_\phi(\boldsymbol{\epsilon}))d\boldsymbol{\epsilon} \\ \mathbb{E}_q[f(\mathbf{z})] &= \mathbb{E}_p[f(g_\phi(\boldsymbol{\epsilon}))],\end{aligned}\tag{4.21}$$

hence we can use the right hand side of Equation 4.21 to obtain differentiable estimates of any function $f(\mathbf{z})$. Specifically, we use the Monte Carlo estimates

$$\mathbb{E}_p[f(g_\phi(\boldsymbol{\epsilon}))] \approx \frac{1}{L} \sum_{l=1}^L f(g_\phi(\boldsymbol{\epsilon}_l)).\tag{4.22}$$

An interesting property of reparametrization gradients is that they have relatively low variance compare to score gradients. To our knowledge, there is no formal proof about this property (Ruiz et al., 2016), and as suggested by Gal (2016) and Zhang et al. (2018a) it is not always the case. However, we show the low variance property of reparametrization gradients in the following simulation example.

Let us assume that $p(x) \sim \mathcal{N}(\theta, 1)$ and we want to minimize

$$\arg \min_{\theta} \mathbb{E}_p[x^2].\tag{4.23}$$

The score derivative is given by

$$\begin{aligned}\frac{\partial}{\partial \theta} \mathbb{E}_p[x^2] &= \int \frac{\partial}{\partial \theta} p(x) x^2 dx \\ &= \mathbb{E}_p\left[\frac{\partial}{\partial \theta} \log[p(x)] x^2\right] \\ &= \mathbb{E}_p[(x - \theta)x^2].\end{aligned}\tag{4.24}$$

Now, let us use the reparameterization $x = \theta + \epsilon$ where $q(\epsilon) \sim \mathcal{N}(0, 1)$. Therefore, $\mathbb{E}_p[x^2] = \mathbb{E}_q[(\theta + \epsilon)^2]$ and its derivative is

$$\begin{aligned}\frac{\partial}{\partial \theta} \mathbb{E}_q[(\theta + \epsilon)^2] &= \int \frac{\partial}{\partial \theta} q(\epsilon) (\theta + \epsilon)^2 d\epsilon \\ &= \mathbb{E}_q[2(\theta + \epsilon)].\end{aligned}\tag{4.25}$$

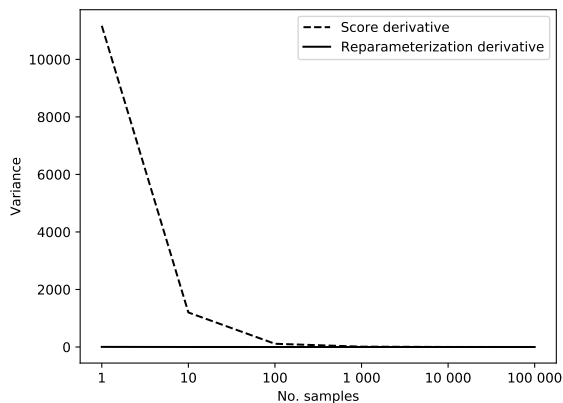


Figure 4.2: Variance for reparameterization gradients (solid line) and score gradients (dashed line) for $N = [1, 10, 100, 1000, 10000, 100000]$ samples from $p(x) \sim \mathcal{N}(\theta, 1)$, where $\theta = 10$, and $q(\epsilon) \sim \mathcal{N}(0, 1)$.

We simulate $N = [1, 10, 100, 1000, 10000, 100000]$ samples from $p(x) \sim \mathcal{N}(\theta, 1)$, where $\theta = 10$, and $q(\epsilon) \sim \mathcal{N}(0, 1)$ to estimate the variance of 100 Monte Carlo estimates of Equation 4.24 and 4.25. The results in Figure 4.2 clearly show that the variance of the reparameterization gradients is smaller than the score gradient for less than 1000 observations. Remember that we are interested in an efficient way to estimate the gradient and sampling 1000 observations is too costly.

4.3.2 Backpropagate gradients through a deterministic reparameterization

The second advantage of reparameterization gradients is that we can use gradient optimization on the neural network reparameterizing $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ in $q(\mathbf{z}|\mathbf{x})$. Note that we need to draw samples from $q(\mathbf{z}|\mathbf{x})$ to feed the decoder neural network $p(\mathbf{x}|\mathbf{z})$. At the same time, we need to backpropagate through the encoder neural network $q(\mathbf{z}|\mathbf{x})$.

Therefore, the reparameterization $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon}$ *excludes* the random component ($\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$) from the encoder MLP as shown in Figure 4.3. The left panel shows a neural where the latent variable \mathbf{z} is drawn at the output layer. It is not clear how we could backpropagate through such a network. On the other hand, the architecture on the right panel allows us to backprop-

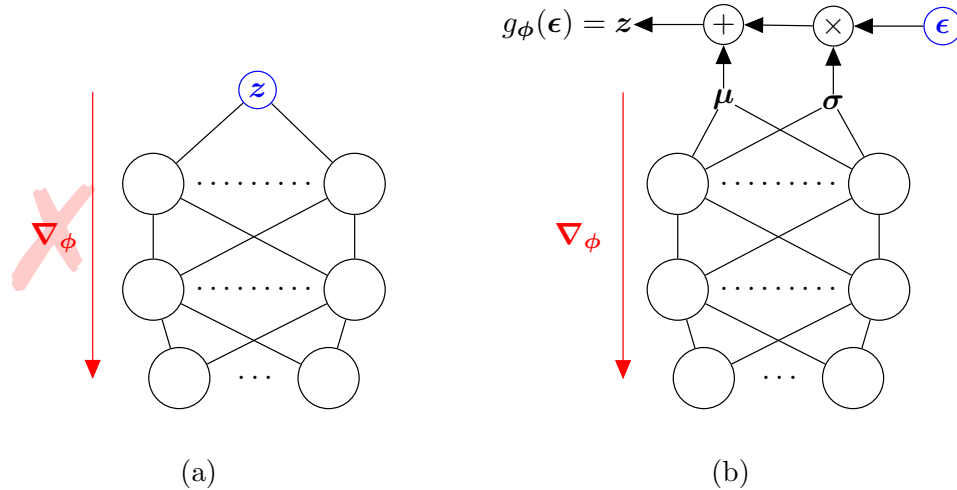


Figure 4.3: The panel (a) shows an MLP with a random component (depicted in blue) at the output layer. It is not clear how we can apply the chain rule of differentiation in such a case. However, the MLP in the (b) panel shows the reparametrization trick where the random component (depicted in blue) lies outside from the iterative procedure of the chain rule of differentiation used in the backpropagation algorithm. Details about the backpropagation algorithm can be found in Appendix A.

agate through the network and at the same time draw z -variables to feed the decoder neural network.

4.4 Improving DGMs

Despite the promising results achieved by VAEs in different research domains, there are well-known problems that limit their potential. This section reviews some of the limitations of VAEs, as well as the research focused on correcting and improving them. Note that the problems and solutions presented here also apply to any DGM.

4.4.1 Tightness of the ELBO

DGMs optimizes the evidence lower bound (ELBO), which is an optimization over the family of variational densities $q(\mathbf{z}|\mathbf{x}) \in \mathcal{Q}$. It is easy to show that

the log-likelihood in the VAE is equal to

$$\begin{aligned} \log p(\mathbf{x}) &= \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})] + KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})] \\ &\equiv \mathcal{L}(\mathbf{x}) + KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})]. \end{aligned} \quad (4.26)$$

Hence, the variational lower bound approaches $\log p(\mathbf{x})$, which is referred to as the tightness of the ELBO, as the approximation $q(\mathbf{z}|\mathbf{x})$ becomes closer to the true posterior distribution $p(\mathbf{z}|\mathbf{x})$.

Burda et al. (2016) show that

$$\log p(\mathbf{x}) \geq \mathcal{L}_{k+1}(\mathbf{x}) \geq \mathcal{L}_k(\mathbf{x}) \quad \forall k, \quad (4.27)$$

where

$$\mathcal{L}_k(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}_{1:k})} \left[\log \frac{1}{k} \sum_{i=1}^k \frac{p(\mathbf{x}, \mathbf{z}_i)}{q(\mathbf{z}_i|\mathbf{x})} \right], \quad (4.28)$$

and $q(\mathbf{z}_{1:k}) = q(\mathbf{z}_1)q(\mathbf{z}_2) \cdots q(\mathbf{z}_k)$. Note that $k = 1$ recovers the lower bound in the VAE and for $k > 1$ Equation 4.28 is a closer approximation to $\log p(\mathbf{x})$.

Domke and Sheldon (2018) note that $\mathbb{E}_q[p(\mathbf{x}, \mathbf{z})/q(\mathbf{z}|\mathbf{x})] = p(\mathbf{x})$ and, therefore, the more concentrated $p(\mathbf{x}, \mathbf{z})/q(\mathbf{z}|\mathbf{x})$ is around its mean $p(\mathbf{x})$, the tighter the lower bound is. Hence, the sample average

$$\frac{1}{k} \sum_{i=1}^k \frac{p(\mathbf{x}, \mathbf{z}_i)}{q(\mathbf{z}_i|\mathbf{x})} \quad (4.29)$$

should be more concentrated around $p(\mathbf{x})$. The bound proposed by Burda et al. (2016) is not only a closer approximation of $\log p(\mathbf{x})$, but its implicit variational distribution $q(\mathbf{z}|\mathbf{x})$ gets closer to the true posterior $p(\mathbf{z}|\mathbf{x})$ as $k \rightarrow \infty$ (Cremer et al., 2017). Hence, its latent space is more flexible compared to regular VAEs.

4.4.2 Beyond the mean-field assumption

It is common to assume a fully factorized variational distribution in the VAE, which is often referred to as mean-field variational distribution (Equation 3.37). The implication is that each latent variable is independent and with its own distribution. However, this choice may impact the quality of inference made by such a model.

Rezende and Mohamed (2016) acknowledge that one limitation of the mean-field variational approximation is that even in an asymptotic regime it would never recover the true posterior distribution. Therefore, they propose a family of flexible distributions $q(\mathbf{z}|\mathbf{x})$, which can contain the true posterior as one of their possible solutions, using normalizing flows. A normalizing flow transforms a probability density through an invertible mapping that applies the rule of change of variable (Lemma 2) and obtains a valid distribution. One limitation of the normalizing flows proposed by Rezende and Mohamed (2016) is that they do not scale to high-dimensional latent spaces (Kingma et al., 2016). To overcome such limitation, Kingma et al. (2016) introduce inverse autoregressive flows, which utilize Gaussian autoregressive autoencoders (Germain et al., 2015) to parametrize Gaussian parameters at each step in the flow. Other examples of normalizing flows are presented in (Dinh et al., 2014; Salimans et al., 2015; Dinh et al., 2016).

Using a different approach, Tran et al. (2016) introduce variational Gaussian processes (VGPs). VGPs are Bayesian nonparametric variational models able to draw inputs from simple distributions, which are then pass through nonlinear mappings to approximate any posterior distribution. Interestingly, the nonlinear outputs are used in a mean-field distribution. However, the VGP is able to capture correlation between latent variables since its independent Gaussian processes share the same latent input.

4.4.3 The Kullback-Leibler divergence is restrictive

Let us assume the factorization $q(\mathbf{x}, \mathbf{z}) = p(\mathbf{x})q(\mathbf{z}|\mathbf{x})$, where $p(\mathbf{x})$ is the true data density and it is approximated by the empirical data distribution $\hat{p}(\mathbf{x})$

given a data set. Note that the mutual information

$$\begin{aligned}
 I(\mathbf{x}, \mathbf{z}) &= \int q(\mathbf{x}, \mathbf{z}) \log \frac{q(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} d\mathbf{x}d\mathbf{z} \\
 &= \int q(\mathbf{x}, \mathbf{z}) [\log q(\mathbf{z}|\mathbf{x}) - \log q(\mathbf{z}) + \log p(\mathbf{z}) - \log p(\mathbf{z})] d\mathbf{x}d\mathbf{z} \\
 &= \int q(\mathbf{x}, \mathbf{z}) \left[\log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} - \log \frac{q(\mathbf{z})}{p(\mathbf{z})} \right] d\mathbf{x}d\mathbf{z} \\
 &= \int q(\mathbf{x}, \mathbf{z}) \left[\log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} \right] d\mathbf{x}d\mathbf{z} - KL[q(\mathbf{z})||p(\mathbf{z})] \\
 &\leq \int q(\mathbf{x}, \mathbf{z}) \left[\log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} \right] d\mathbf{x}d\mathbf{z} \\
 &= \mathbb{E}_{p(\mathbf{x})}[KL[q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]], \tag{4.30}
 \end{aligned}$$

where the inequality is a result of the non-negative KL divergence (see Equation 3.19). Hence, the mutual information is a lower bound of the average KL divergence over the data. The ELBO minimizes Equation 4.30, which leads to less informative latent representations. Alemi et al. (2018) derive the bounds of mutual information and show how they relate to the optimization of the ELBO.

In addition to the undesired upper bound on mutual information, the KL divergence tends to underestimate the variance of the posterior distribution of latent variables and can degenerate solutions that zero out the probability of some configurations of latent variables (Ranganath et al., 2018). Hence, Ranganath et al. (2018) introduce operator variational inference, which utilizes the Langevin-Stein objective function. Their method results in richer approximations that do not require analytically tractable densities.

Li and Turner (2016) use the Renyi’s α -divergence, which includes the KL divergence as a special case, to derive a new variational Renyi (VR) bound. Interestingly, the Monte Carlo approximation of the VR bound is a generalization of the lower bound introduced by Burda et al. (2016) (Equation 4.28).

4.4.4 Learning expressive latent representations

Posterior collapse occur when the variational distribution matches the prior distribution, implying that the latent variables do not depend on the data

(Lucas et al., 2019; Dieng et al., 2019). Further, the generative model tends to ignore the latent variables since they do not contain useful information about the data. The KL divergence term in the ELBO has been commonly blamed as the cause of the posterior collapse. However, Lucas et al. (2019) show that it might be spurious local maxima in the ELBO that ultimately can lead to the posterior collapse.

Dieng et al. (2019) propose to use neural networks with skip connections to parametrize the generative model in the VAE. In that way, the generative model forces the likelihood function to maintain a strong connection between the data and its latent representation. Other approaches to alleviate posterior collapse is to simply anneal the KL term (Bowman et al., 2015; Sønderby et al., 2016) as this prevents units in the inference network from being idle at the beginning of the optimization.

Higgins et al. (2016) introduce a constraint on the maximization to the generative model to derive their proposed β -VAE, which scales the KL divergence term by the β term. As Higgins et al. (2016) pointed out, the new parameter β modifies the learning pressure during training, encouraging the learning of different representations.

Part II

Summary of research

Chapter 5

Paper I - Learning Latent Representations of Bank Customers with the Variational Autoencoder

The Variational Autoencoder has shown encouraging results in different research fields. Inspired by those results and the lack of useful representation learning methods for credit scoring, we develop a novel representation learning technique for credit scoring. To that end, we introduce a supervision stage into the VAE framework where we form a specific grouping of the data using the Weight of Evidence transformation. Our method provides a data representation in the latent space of the VAE, which captures the customers' creditworthiness in a well-defined clustering structure.

The empirical results in Paper I show that our proposed method is able to capture the intrinsic clustering structure of the credit data. Other representation learning methods, such as ISO maps, kernel PCA, t-SNE, and PCA, fail to learn a representation with a well-defined clustering structure. Further, the learned data representation of our proposed method is also able to capture the spatial coherence of customers' creditworthiness, see Figure 5.1. The clusters identified in the latent space not only have significantly different default rates, but also rank the default probability across the dimensions of the latent space.

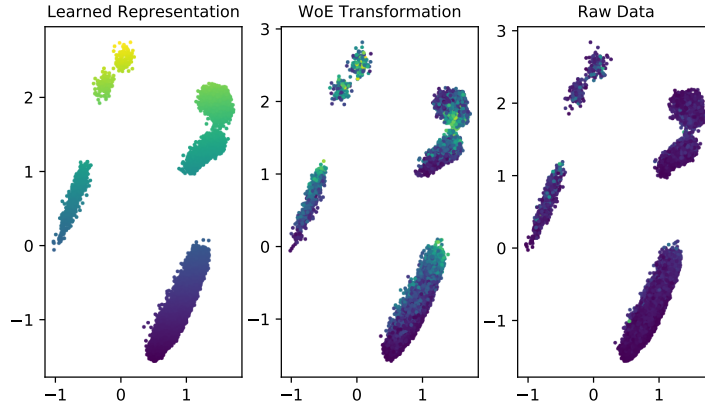


Figure 5.1: Best viewed in color. We estimate the default probability $Pr(y = 1|\mathbf{x})$ for different customers in the data set used in Paper I using three different input data \mathbf{x} : i) our proposed data representation, ii) WoE transformations, and iii) raw data. Further, we generate the latent space for those customers using the trained VAE in our proposed approach. Finally, we use the three estimated values for $Pr(y = 1|\mathbf{x})$ to create a colormap on the learned data representation with our proposed methodology. Note that the left panel shows a smooth color transition.

Other advantage of our proposed method is that the number of clusters is suggested by the learned representation itself, and we proposed an automated way to assign cluster labels during training. Furthermore, the VAE can generate the latent configuration of new customers and assign them to one of the existing clusters. Finally, we show that the data representations of bank customers can be used to obtain descriptive labels associated with each cluster, which are then used in marketing campaigns, or can be used to improve the credit scoring accuracy using a segment-based approach. See Appendix B for more details about the segment-based credit scoring approach using the data representations learned by our proposed method.

5.1 Contributions by the author

The idea was conceived by myself and further developed in cooperation with my supervisors that are co-authors in the paper. The data extraction for the Santander Bank data was done by bank’s analysts. My contributions are as

follows:

- I suggested to introduce a supervision stage in the VAE framework to learn a useful data representation for credit scoring data.
- I developed an automatic approach to assign cluster labels while the VAE is trained.
- I coded the model and experiments conducted in this paper and Michael Kampffmeyer helped me to debug some parts of the code.
- I wrote the manuscript draft of the paper, which was further improved in collaboration with the co-authors of the paper.

Chapter 6

Paper II - Deep Generative Models for Reject Inference in Credit Scoring

Commonly, credit scoring models are developed based on accepted applications, even though such a sample is biased since it excludes rejected applications systematically. This problem is known as selection bias. The main motivation in Paper II is to minimize the selection bias problem by adding the rejected applications into the model development and infer their unknown creditworthiness since we do not have class labels for rejected applications. To that end, we developed two new methods for reject inference in credit scoring combining auxiliary variables and Gaussian mixture models in a semi-supervised framework with generative models for the first time. Figure 6.1 shows the plate notation for our proposed models.

Our main motivation to include Gaussian mixture models is two fold. First, we hypothesized that the two class labels in credit scoring data are generated by two different process. Second, a Gaussian mixture model generates a flexible latent space, helping to improve the approximation of the inference process.

Auxiliary variables are used to improve the classification power of our proposed model. Therefore, the classifier takes the form $q(y|\mathbf{x}, \mathbf{a})$ where y is the class label for customers, \mathbf{x} is the feature vector and \mathbf{a} is a latent aux-

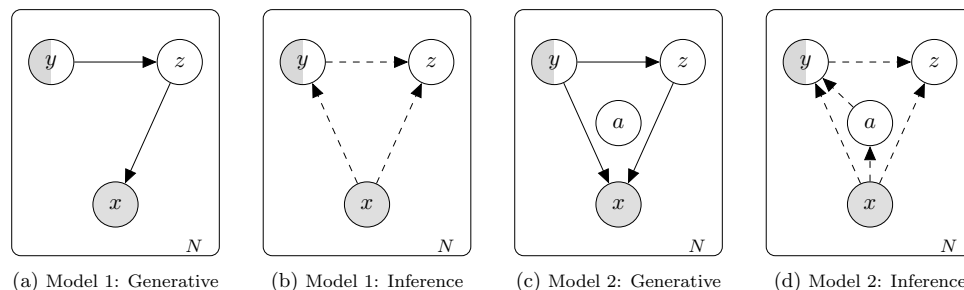


Figure 6.1: Plate notation for Model 1 and Model 2 in Paper II, where x is the observed feature vector, y is the outcome of the loan and it is only observed for the accepted applications, and z are latent variables. The generative process is specified by solid lines, while the inference process is shown with dotted lines.

iliary variable capturing high-level information about the customers' credit worthiness.

The empirical results in Paper II show that our proposed models achieve higher model performance compared to many models for reject inference in credit scoring. Further, model performance increases as we add more data for model training. Our experiments also show that our proposed methodology scales to large data sets, which is a major advantage compared to other methods for reject inference in credit scoring. Finally, the results in Paper II confirmed the powerful information embedded in the latent space of auxiliary variables when used for downstream classification tasks.

6.1 Contributions by the author

The original idea of Paper II was proposed by myself and improved in collaboration with the co-authors of the paper. Further, we use a real data set provided by bank's analysts at Santander Bank. My own contributions are:

- I developed the two semi-supervised models proposed in the paper.
- I fully derived the objective function in the models.
- I coded all models in the experimental setup with exception of the semi-supervised support vector machine that was made available by the

original author of the model. Michael Kampffmeyer helped to debug some parts of the code.

- I wrote the main draft of the paper and it was improved together with the co-authors.

Chapter 7

Paper III - Generating Customer's Credit Behavior with Deep Generative Models

Retail banks collect information during the application process as well as through the loan period, see Figure 7.1. Hence, banks possess multiple measurement modalities that provide complementary information about the customers' creditworthiness. Multi-modal learning designs models that utilize multiple modalities to learn a shared data representation that traditionally has been used in downstream classification.

In Paper III, we developed a novel multi-modal learning model for credit scoring that is able to learn a shared latent data representation that is not only useful for downstream classification, but it can also be used to generate future credit data. Further, our proposed model generates the future data conditioned on the information obtained during the application process, keeping the relationship between these two modalities. We also introduced a new lower bound that maximizes mutual information between view \mathbf{x}_2 and the shared latent representation \mathbf{z} .

The empirical results in Paper III show that our proposed model can generate future credit data more accurately than the state-of-the-art multi-modal learning models. Further, our proposed model achieves on a pair downstream classification results compared to the benchmark model. Finally, the lower

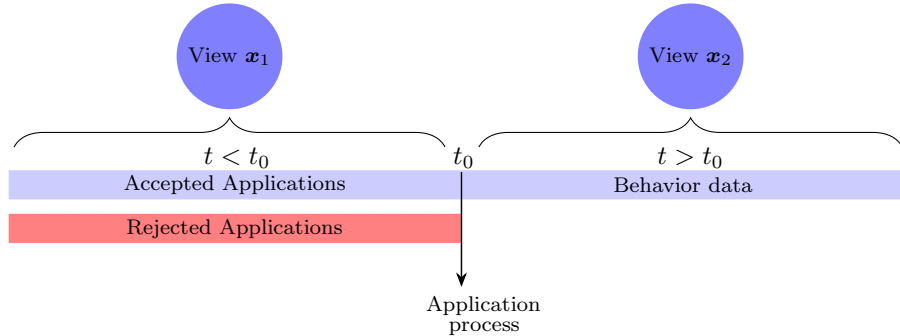


Figure 7.1: Multi-modal credit data. At the time of the applications process t_0 , only \mathbf{x}_1 is available. This data view, which commonly is composed of socio-demographic features, is generated during $t < t_0$ and is used in traditional credit scoring models. After the loan is granted, a new view of data \mathbf{x}_2 is generated and it provides complementary information about the customer. Commonly, the view \mathbf{x}_2 is used to develop behavior models.

bound introduced in our proposed methodology helps to learn an amortized inference distribution and achieves higher model performance compared to the classical lower bound in DGMs.

7.1 Contributions by the author

The original idea of Paper III was proposed by myself and improved in collaboration with the co-authors of the paper. My own contributions are:

- I developed the proposed model and its lower bound.
- I coded all models in the experimental setup with exception of the CCA-based models that were obtained by the authors of the models DCCA and DCCAE. Michael Kampffmeyer helped to debug some parts of the code.
- I designed the experimental setup with help from Kjersti Aas, Michael Kampffmeyer, and Robert Jensen.
- I wrote the main draft of the paper and it was improved together with the co-authors.

Chapter 8

Concluding remarks

The research in this thesis develops novel methods in credit scoring using DGMs that provide solutions to some problems in credit scoring. Narrowing the gap between research based on plain-vanilla neural networks and advanced machine learning models, such as DGMs. Concretely, we focused on developing a useful representation learning methodology, which is able to capture the customers' creditworthiness. Further, we developed novel models for reject inference in credit scoring that are able to infer the unknown creditworthiness of rejected applications. Finally, we addressed credit scoring from a multi-modality learning point of view to learn shared data representations that are useful to generate future credit data and for downstream classification.

We investigated different dimensionality reduction methods to deal with high-dimensional data in credit scoring, and developed a new methodology which is not only able to reduce the dimensionality of the input data, but it is also able to learn a useful data representation that captures the customers' creditworthiness. The learned data representation of our proposed method reflects the natural clustering structure of the credit data and encapsulates the creditworthiness for each group. Therefore, the clusters that are identified by our proposed methodology are well suited for a segment-based credit scoring approach, which achieves higher performance compared to the traditional credit scoring approach where only one classifier is fitted to the entire data set.

The selection bias problem is inherent in credit scoring modeling since banks only know the true label for accepted customers and not for applicants who were rejected. We studied this problem from a semi-supervised learning point of view. That is, including rejected applications with no class labels together with applications where the label is known. Our proposed method infers the unknown customers’ creditworthiness by exact enumeration of the two possible states for class labels. Further, classification accuracy increases with the amount of data used for training our proposed models. Finally, our empirical results confirm the powerful information embedded in latent representations for downstream classification with DGMs.

Finally, we addressed credit scoring from a multi-modal learning point of view. We developed a novel multi-modal learning model for credit scoring, which learns a shared data representation to generate future credit data \mathbf{x}_2 and for downstream classification. Further, we introduced a novel lower bound that optimizes mutual information between the shared latent representation \mathbf{z} and the view of data \mathbf{x}_2 . Our empirical results show that our proposed model performs best at reconstructing the future view \mathbf{x}_2 , and obtains on a par classification accuracy compared to the state-of-the-art multi-modal learning models. Finally, our proposed lower bound improves the generative process of our proposed model as well as its classification accuracy.

8.1 Weaknesses and future work

In this section we discuss some limitations that we have identified in the papers included in this thesis. Likewise, we suggest future research work.

Paper I: One of the key steps to learn a useful data representation in our proposed method, is the Weight of Evidence transformation. We showed that the fine classing WoE approach does not encapsulate the customers’ creditworthiness in the same way as the coarse classing approach. Therefore, the latent space in the VAE depends on the coarse classing WoE. It will be useful to add an automated coarse classing approach to our proposed method to developed an end-to-end representation learning and clustering algorithm. In this way, the user does not require previous knowledge about the coarse and fine classing WoE transformations.

We show that the clusters in the learned data representations are appropriate

for a segment-based credit scoring approach. It will be interesting to develop a model which uses the learned data representation in Paper I for classification in a unified framework, that is optimizing representation learning and classification accuracy. Further, it will be interesting to compare the predictive power in a segment-based classification task using the clusters identified with our propose methodology and using other clustering techniques.

Our proposed method is able to generate the latent representation of an unseen customer and assign it to an existing cluster. It would be interesting to analyze cluster assignment as a function of time using different snap-shots of customer's information.

Paper II: Bayesian inference arrives to posterior distributions of unknown parameters or density functions, which is a useful approach to answer relevant queries about a given problem. However, the price of Bayesian inference is the assumption of likelihood functions, or the generative process in DGMs. Assuming multivariate Gaussian distributions for credit data, as we did in Paper II, can always be argued. Similarly, our proposed models in Paper II assume diagonal covariance matrices. This assumption is made to keep models simple and we let for future research to analyze the impact of other type of covariance matrices.

One of the motivations for adding rejected applications into the modeling exercise in Paper II is to deal with the selection bias. However, given the nature of the data sets that we used in Paper II, where we do not have the true label for the rejected applications, we are not able to test whether our proposed models are able to correct the selection bias. In a future research, it will be interesting to design an experiment where we can test, to some extend, whether our proposed reject inference models can solve the selection bias problem in credit scoring.

The experimental setup in Paper II focused on evaluating the predictive power of the models. Using different performance metrics and under different scenarios with different number of labeled and unlabeled data points. However, we did not test the benefits of including a Gaussian Mixture Models. Similarly, we did not include an analysis of the latent space generated by such a Gaussian Mixture. Another interesting alternative for future research is to developed a model where the auxiliary variable, the one used for classification in addition to the input data, is pulled towards a Gaussian Mixture and evaluate the predictive power of such a model.

Paper III: CCA-based models optimize the canonical correlation between shared latent representations. This approach works well when only one view of data is available during test time, and when the views are uncorrelated. On the other hand, models maximizing a variational lower bound do not care about the correlation between shared data representations. Therefore, it would be interesting to add a term in the variational lower bound that maximizes canonical correlation. This choice can improve the performance of variational-based methods.

Our proposed model in Paper III uses maximum mean discrepancy and Gaussian kernels to maximize mutual information between the shared latent representation and the view \mathbf{x}_2 . We left for further research to investigate the impact of different divergence measures or different type of kernels. It would be interesting to try Stein variational gradient approaches and see if the reconstruction of view \mathbf{x}_2 can be further improved.

Once again, for simplicity, we assumed a diagonal covariance matrix for the generative model in our proposed model. The choice of diagonal covariance matrices may harm the reconstruction for some features, e.g. age and income. Therefore, it would be interesting to model a full covariance matrix in the generative process and see the impact on the reconstruction of features in view \mathbf{x}_2 .

Finally, in Paper III, we motivated the choice for a point estimate for features in view \mathbf{x}_2 based on a quadratic loss function for simplicity. It would be an interesting avenue for future research to incorporate Bayesian decision theory into the optimization of the lower bound.

Part III

Included papers

Chapter 9

Paper I



Learning latent representations of bank customers with the Variational Autoencoder

Rogelio A. Mancisidor^{a,b,d,*}, Michael Kampffmeyer^{a,c,d}, Kjersti Aas^c, Robert Jenssen^{a,c,d}

^a Department of Physics and Technology, Faculty of Science and Technology, UiT - The Arctic University of Norway, Hansine Hansens veg 18, Tromsø 9037, Norway

^b Credit Risk Models, Santander Consumer Bank AS, Strandveien 18, Lysaker 1325, Norway

^c Norwegian Computing Center, P.O. Box 114 Blindern, Oslo, Norway

^d RAM, MK, and RJ are all with the UiT Machine Learning Group: <http://machine-learning.uit.no>, Tromsø, Norway

ARTICLE INFO

Keywords:

Variational Autoencoder
Data representations
Clustering
Machine learning

ABSTRACT

Learning data representations that reflect the customers' creditworthiness can improve marketing campaigns, customer relationship management, data and process management or the credit risk assessment in retail banks. In this research, we show that it is possible to steer data representations in the latent space of the Variational Autoencoder (VAE) using a semi-supervised learning framework and a specific grouping of the input data called Weight of Evidence (WoE). Our proposed method learns a latent representation of the data showing a well-defined clustering structure. The clustering structure captures the customers' creditworthiness, which is unknown a priori and cannot be identified in the input space. The main advantages of our proposed method are that it captures the natural clustering of the data, suggests the number of clusters, captures the spatial coherence of customers' creditworthiness, generates data representations of unseen customers and assign them to one of the existing clusters. Our empirical results, based on real data sets reflecting different market and economic conditions, show that none of the well-known data representation models in the benchmark analysis are able to obtain well-defined clustering structures like our proposed method. Further, we show how banks can use our proposed methodology to improve marketing campaigns and credit risk assessment.

1. Introduction

Banks need to estimate the creditworthiness of both customers and applicants to improve marketing campaigns, customer relationship management, data and process management or the credit risk assessment (Anderson, 2007). Further, Anderson (2007) suggests that customer segmentation can improve the aforementioned bank activities. Therefore, it is important to learn a data representation of bank customers that has the ability to express the natural clustering of the data, and that can be used in marketing campaigns, product offering or in improving the credit risk assessment.

The Variational Autoencoder (VAE) (Kingma & Welling, 2013; Rezende et al., 2014) has shown promising results in different research domains. The powerful information embedded in its latent space has been documented e.g., in health analytics (Rampasek & Goldenberg, 2017; Titus et al., 2018; Way & Greene, 2017a, 2017b), in speech emotion recognition (SER) (Latif et al., 2017), and in natural language processing (NLP) (Bowman et al., 2015; Su et al., 2018), among others.

Additionally, research has been conducted where the VAE has been modified to improve its feature learning properties, e.g. Bouchacourt et al. (2018), Higgins et al. (2017), Hsu et al. (2017), Su et al. (2018). However, to the best of our knowledge there is no previous work on data representations of bank customers using a modified version of the VAE and therefore this is the first research to focus on the development of a data representation framework suitable for the bank industry.

Inspired by the previous results in other research fields and the lack of research on learning data representations of bank customers, we adopt the VAE and the Auto Encoding Variational Bayesian (AEVB) algorithm (Kingma & Welling, 2013) and propose a new framework that effectively learns a data representation that is useful to support the aforementioned banking activities. Our proposed method is able to steer the latent embeddings in the VAE by transforming the input data into a meaningful representation, and by creating a specific grouping of the data. Hence, the focus in this research is use the effective manifold

* Corresponding author at: Department of Physics and Technology, Faculty of Science and Technology, UiT - The Arctic University of Norway, Hansine Hansens veg 18, Tromsø 9037, Norway.

E-mail addresses: rogelio.a.mancisidor@uit.no (R.A. Mancisidor), michael.c.kampffmeyer@uit.no (M. Kampffmeyer), kjersti@nr.no (K. Aas), robert.jenssen@uit.no (R. Jenssen).

<https://doi.org/10.1016/j.eswa.2020.114020>

Received 10 April 2019; Received in revised form 20 August 2020; Accepted 14 September 2020

Available online 15 September 2020

0957-4174/© 2020 Elsevier Ltd. All rights reserved.

learning capabilities of the VAE (Goodfellow et al., 2016) and develop a new framework which is able to capture valuable information in the latent space for bank activities.

The main advantage of our method is that it learns a data representation in the low-dimensional latent space generated by the VAE, which can be visualized and suggests well-defined clustering structures. Therefore, our method reveals the number of groups in the natural clustering structure of the data. Further, these clusters are well suited for the bank industry given that they encapsulate different risk profiles, which are unknown a priori and cannot be identified in the input space that is high-dimensional and with complex relationships. In addition, the latent representations not only encapsulate creditworthiness, but also preserve its spatial coherence. Using the generative properties of the VAE, we can draw the latent space of unseen customers and map them into an existing cluster without the need of further supervision. Finally, our empirical results, based on real data sets reflecting different market and economic conditions, show that the data representations obtained with our proposed method are able to obtain well-defined clustering structures capturing the customers' creditworthiness, unlike some well-known data representation models, and we also show how banks can use our proposed methodology to improve marketing campaigns and credit risk assessment.

This paper is organized as follows. Section 2 reviews the related work where the VAE has been used to learn data representations in different research fields, while Section 3 introduces variational inference and the VAE. In Section 4 we explain the data transformation used to learn latent representations of bank customers and Section 5 presents our experiments and findings. Finally, Section 7 presents the main conclusions in this paper.

2. Related work

Methods to learn data representations from the input data can be divided into probabilistic graphical models (PGMs) and neural network-based models (Bengio et al., 2013). Data representations play an important role in the results we can achieve in detection or classification tasks (Bengio et al., 2013; LeCun et al., 2015; Zhong et al., 2016). The ability to express general-purpose priors, such as natural clustering or spatial coherence, among others, is what make data representations to be good (Bengio et al., 2013).

Further, PGMs aim to learn latent representations z , which are able to describe the input data x . This is done by modeling their joint distribution $p(z, x)$. Depending on how this joint distribution is constructed, PGMs can be divided in directed or undirected graphical models (Bengio et al., 2013).

The Variational Autoencoder (Kingma & Welling, 2013; Rezende et al., 2014) is an influential (unsupervised) directed probabilistic graphical model, which has been widely used to learn meaningful latent representations of the input data. For example, latent representations of gene expression data are used in Way and Greene (2017a) for cancer prediction. The results show that the VAE latent features are useful to predict cancer and its predictive power is similar to other data transformation methods, e.g. principal component analysis (PCA) (Pearson, 1901).

Latent representations in the VAE have also been used for predictions in a semi-supervised context. In Rampasek and Goldenberg (2017), latent representations for pre-treatment and post-treatment gene expression are used to predict drug response. Their proposed model achieves higher performance relative to Ridge logistic regression (Hoerl & Kennard, 1970) using the original input data. In addition, PCA transformations are used in three different classifiers to predict drug responses, but their performance, in most of the experiments, is not better relative to Ridge regression and the VAE model.

Classification of speech emotion is another example where latent representations of the input data have been successfully used for classification. Using Long Short Term Memory (LSTM) networks to classify

emotion, Latif et al. (2017) compare the predictive power of data transformations using the VAE and regular Auto Encoders. Speech emotion prediction is more accurate when the latent representations in the VAE are used as predictors. The classification results are further improved by using latent representations obtained with conditional VAE (Sohn et al., 2015).

In another classification study, Titus et al. (2018) train logistic regression models, on t-SNE (Hinton & Roweis, 2003) embeddings of high-dimensional VAE latent variables, to classify tumors. Their results show that the latent embeddings in the VAE learn a biological relevant information and successfully classify disease sub-types. Both works in Latif et al. (2017), Titus et al. (2018) build upon the Tybalt model (Way & Greene, 2017b). The Tybalt exploits the data transformation capabilities of the VAE to generate latent representations of gene expression data.

The VAE has also been used in the natural language processing field. Studying bilingual word embeddings, Su et al. (2018) use the VAE to generate latent representations, which explicitly induce the underlying semantics of bilingual text. Their model is able to learn a hidden representation of paired bag-of-words sentences. Furthermore, in Bowman et al. (2015) recurrent neural networks are combined with the VAE to model text data. The latent transformations are able to generate coherent sentences. In addition, the proposed model in this research is able to impute missing words in text corpus.

Research has also been conducted on modifying the original VAE aiming to improve the quality of the learned latent representations. In Higgins et al. (2017), for example, the authors add an hyperparameter β to the VAE, which limits the capacity of the latent information channel and impose an emphasis on learning statistically independent latent factors. Hence, the model is able to learn disentangled factors of variation.

In Bouchacourt et al. (2018) the concept of supervision in VAE is introduced. The authors group the input data, aiming to learn representations of the data that reflect the semantics behind a specific grouping of the data. In other words, the grouping makes it possible to learn a semantically useful data transformation. Similarly, Hsu et al. (2017), Su et al. (2018) use supervision but in the latent space. Both works Hsu et al. (2017), Su et al. (2018), manipulate the latent representations arithmetically to decompose the latent representation into different attributes.

There has been some work on segmentation in both the financial industry and in the marketing area. Hand et al. (2005) studies whether credit risk assessment can be improved by creating segments, which are created using a bipartite model. Bijak and Thomas (2012) use decision trees and chi-squared automation interaction detection trees for identifying customer segments. Further, they proposed a unified framework where segmentation and credit risk assessment is optimized simultaneously. Aurifeille (2000) uses a genetic algorithm and linear regressions to identify clusters in a marketing data set. More recently, Xiao et al. (2016) use an ensemble approach where k-means is used for segmentation. Similarly, Lim and Sohn (2007) use k-means to develop a dynamic model for credit risk assessment. However, the focus of the research in this paper is on representation learning of bank customers and not on coupling existing clustering and classification techniques.

In this research, as in Bouchacourt et al. (2018), Hsu et al. (2017) and Su et al. (2018), we introduce a supervision stage into the VAE. In this stage, we form groups that share a common factor of variation. The difference in our method is that the grouping is derived from the class label, see Section 4. This means that our proposed method is a semi-supervised representation learning model where we indirectly steer the data transformation using a specific grouping of the input data. Finally, we only focus on learning a data representation of bank customers' data that is able to capture the customers' creditworthiness in the latent space of the VAE, and not in the predictive power of such representations.

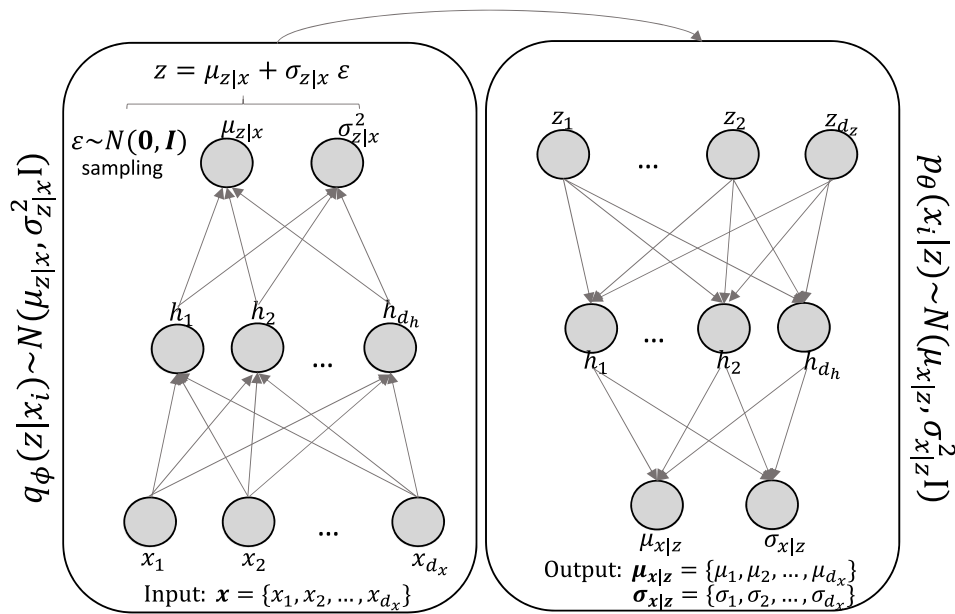


Fig. 1. Graphical representation of the VAE. The multilayer perceptron network to the left corresponds to the probabilistic encoder $q_{\phi}(z|x)$ where $x \in \mathbb{R}^{d_x}$ is the network input. The output of the network are the parameters in $q_{\phi}(z|x) \sim \mathcal{N}(\mu_{z|x}, \sigma_{z|x}^2 I)$. Note that $\epsilon \sim \mathcal{N}(0, I)$ is drawn outside the network in order to use gradient descent and backpropagation optimization techniques. Similarly, the feedforward network to the right corresponds to the probabilistic decoder $p_{\theta}(x|z)$. In this case, the input are the latent variables $z \in \mathbb{R}^{d_z}$ and the network output are the parameters in $p_{\theta}(x|z) \sim \mathcal{N}(\mu_{x|z}, \sigma_{x|z}^2 I)$. For readability purposes we do not specify the weights ϕ and θ in the decoder and encoder respectively. However, these parameters are represented by the lines joining the nodes in the networks plus a bias term attached to each node.

3. The variational autoencoder

3.1. Variational inference

In the rest of the paper we use the following notation. We consider i.i.d. data $\{x_i\}_{i=1}^n$ where $x_i \in \mathbb{R}^{d_x}$ is the customers data, e.g. income, age, marital status etc. Further, the latent variables $\{z_i\}_{i=1}^n$ where $z_i \in \mathbb{R}^{d_z}$ are the data transformation of x_i . The subscript i is dropped whenever the context allows for it.

The latent variable in the joint density $p(x, z)$ is drawn from a prior density $p(z)$ and then it is linked to the observed data through the likelihood $p(x|z)$. Inference amounts to conditioning on data and computing the posterior $p(z|x)$ (Blei et al., 2017).

The problem is that the posterior distribution $p(z|x)$ is intractable in most cases. Note that

$$p(z|x) = \frac{p(z, x)}{p(x)}, \tag{1}$$

involves the marginal distribution $p(x) = \int p(z, x) dz$. This integral, called the *evidence*, in some cases requires exponential time to be evaluated since it considers all configurations of latent variables. In other instances, it is unavailable in a closed form (Blei et al., 2017).

Variational Inference (VI) copes with this kind of problem by minimizing the Kullback–Leibler (KL) divergence¹ between the true posterior distribution $p(z|x)$ and a parametric function $q(z|x)$, which is chosen among a set of densities \mathfrak{F} (Blei et al., 2017). This set of densities is parameterized by *variational parameters* and they should be flexible enough to capture a density close to $p(z|x)$ and, in addition, be simple for efficient optimization. The parametric density which minimizes the KL divergence is

$$q^*(z|x) = \arg \min_{q(z|x) \in \mathfrak{F}} KL[q(z|x) \parallel p(z|x)]. \tag{2}$$

¹ The KL divergence $KL[q(\cdot) \parallel p(\cdot)]$ is a measure of the proximity between two densities and it is commonly measured in bits. It is non-negative and it is minimized when $q(\cdot) = p(\cdot)$.

Unfortunately, Eq. (2) cannot be optimized directly since it requires computing a function of $p(x)$. To see this, let us expand the KL divergence using the Bayes' theorem and noting that $p(x)$ does not depend on z

$$\begin{aligned} KL[q(z|x) \parallel p(z|x)] &= E_{z \sim q}[\log q(z|x) - \log p(z|x)] \\ &= E_{z \sim q}[\log q(z|x) - \log p(x, z)] + \log p(x). \end{aligned} \tag{3}$$

Given that Eq. (3) cannot be optimized directly, VI optimizes the alternative objective function

$$\begin{aligned} E_{z \sim q}[\log p(x, z) - \log q(z|x)] &= E_{z \sim q}[\log p(z) + \log p(x|z) - \log q(z|x)] \\ &= E_{z \sim q}[\log p(x|z)] - KL[q(z|x) \parallel p(z)] \\ &= ELBO. \end{aligned} \tag{4}$$

From Eqs. (3) and (4) we have that

$$\log p(x) = KL[q(z|x) \parallel p(z|x)] + ELBO. \tag{5}$$

Since the KL divergence is non-negative, the expression in Eq. (4) is called the *evidence lower bound* (ELBO). Noting that the ELBO is the negative KL divergence in Eq. (3) plus the constant term $\log p(x)$, it follows that maximizing the ELBO leads to minimizing Eq. (2).

It is worth mentioning that the term $KL[q(z|x) \parallel p(z)]$ makes the variational density to be close to the prior distribution, while the term $E_{z \sim q}[\log p(x|z)]$ encourages densities that place their mass on configurations of the latent variables that explain the observed data. The interested reader is referred to Blei et al. (2017), Doersch (2016) for further details.

3.2. The variational autoencoder and AEVB algorithm

The Variational Autoencoder, see Fig. 1, is a generative model, which aims to learn the distribution of the input data x . This means that the VAE can sample from a distribution that it is similar to the one that have generated x . In addition, the VAE assumes that latent variables $p(z) \sim \mathcal{N}(0, I)$ govern the distribution of x . In this research, the input data x represents the customer data, or a specific grouping of it, and the data transformation of such data is generated by $q(z|x)$. This

Table 1

Weight of Evidence transformation of the variable age. The top panel shows the fine classing approach, while the bottom panel shows the coarse approach where only three groups are created.

Fine classing approach								
Age	Count	Distribution all	Goods	Distribution goods	Bads	Distribution bads	Bad rate	WoE
Missing	1 000	2.5%	860	2.38%	140	3.65%	14.00%	-0.4272
18-22	4 000	10%	3 040	8.41%	960	25.00%	24.00%	-1.0898
23-26	6 000	15%	4 920	13.61%	1 080	28.13%	18.00%	-0.7261
27-29	9 000	22.5%	8 100	22.40%	900	23.44%	10.00%	-0.0453
30-35	10 000	25.0%	9 500	26.27%	500	13.02%	5.00%	0.7019
36-44	7 000	17.5%	6 800	18.81%	200	5.21%	2.86%	1.2839
44+	3 000	7.5%	2 940	8.13%	60	1.56%	2.00%	1.6493
Total	40 000	100%	36 160	100%	3 840	100%	9.60%	
Coarse classing approach								
Age	Count	Distribution all	Goods	Distribution goods	Bads	Distribution bads	Bad rate	WoE
Missing	1 000	2.5%	860	2.38%	140	3.65%	14.00%	-0.4272
18-29	19 000	47.5%	16 060	44.41%	2 940	76.56%	15.47%	-0.5445
30-44+	20 000	50%	19 240	53.20%	760	19.79%	3.80%	0.9889
Total	40 000	100%	36160	100%	3840	100%	9.60%	

data representation of the customer data should capture the customers' creditworthiness. In this section, we will show how the VAE approximates Eq. (5) by maximizing the ELBO. This is done using multilayer perceptron (MLP) networks and stochastic gradient optimization.

The MLPs, which optimize the ELBO, estimate the parameters μ and σ^2 in the density functions $p_\theta(x|z)$ and $q_\phi(z|x)$, i.e. $p_\theta(x|z) \sim \mathcal{N}(\mu_{x|z}, \sigma_{x|z}^2 \mathbf{I})$ and $q_\phi(z|x) \sim \mathcal{N}(\mu_{z|x}, \sigma_{z|x}^2 \mathbf{I})$. Note that given that the output of the MLPs are μ and σ^2 , the stochastic gradient optimization is on θ and ϕ , which are the weights in the MLPs. By doing this, the VAE learns the values of μ and σ^2 that maximize the ELBO.²

Specifically, assuming the set of i.i.d vectors $\{x_i, \dots, x_n\}$, the Auto Encoding Variational Bayesian (AEVB) algorithm (Kingma & Welling, 2013) learns the parameters θ, ϕ jointly using MLP networks, and by performing stochastic gradient descent on the

$$ELBO(\theta, \phi, x_i) = E_{z \sim q}[\log p_\theta(x_i|z)] - KL[q_\phi(z|x_i) \| p(z)] \quad (6)$$

for the i 'th customer. Therefore, the MLPs for $q_\phi(z|x)$ and $p_\theta(x|z)$ in Fig. 1 have the following form

$$\begin{aligned} h &= \tanh(W_1 x_i + b_1), & h &= \tanh(W_4 z_i + b_4), \\ \mu_{z|x} &= W_2 h + b_2, & \mu_{x|z} &= W_5 h + b_5, \\ \log \sigma_{z|x}^2 &= W_3 h + b_3, & \log \sigma_{x|z}^2 &= W_6 h + b_6, \\ z_i &= \mu_{z|x} + \sigma_{z|x} \odot \epsilon, & \hat{x}_i &= \mu_{x|z} + \sigma_{x|z} \odot \epsilon, \end{aligned} \quad (7)$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, \odot is the element-wise product, $\phi = \{W_1, W_2, W_3, b_1, b_2, b_3\}$ and $\theta = \{W_4, W_5, W_6, b_4, b_5, b_6\}$ are the unknown parameters in the MLPs for $q_\phi(z|x)$ and $p_\theta(x|z)$ respectively.

It is worth mentioning that the latent variable z has been reparametrized as a deterministic and differentiable system. The reason is that we need to backpropagate the term $E_{z \sim q}[\log p_\theta(x_i|z)]$ in Eq. (6). Without the reparametrization, z would be inside a sampling operation which cannot be propagated. This means that the AEVB algorithm actually takes the gradient of $E_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[\log p_\theta(x_i|z_i = \mu_i + \sigma_i \odot \epsilon)]$ (Kingma & Welling, 2013).

Note that $q_\phi(z|x)$ generates latent variables given x and $p_\theta(x|z)$ converts them into its original representation. Hence, the former is referred as probabilistic encoder and the latter as probabilistic decoder.

4. Learning latent representations

In this section we introduce the motivation for the specific grouping of data that we use to steer a data representation, which encapsulates the customers' creditworthiness in the latent space of the VAE. The

² It is possible to specify other distributions for $p(\cdot)$ and $q(\cdot)$. However, Gaussian distributions are appropriate for our data sets, and we assume a diagonal covariance matrix as in the original VAE.

presumption is that given that the AEVB algorithm has converged to the optimal variational density $q^*(z)$, the latent space should have learned a data representation, which encapsulates the customers' creditworthiness. Otherwise, the reconstruction would have failed, and the algorithm would not have converged to $q^*(z)$ in the first place.

To quantify creditworthiness, let us first define the ground truth class

$$y = \begin{cases} 1 & \text{if at least 90 days past due} \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

At least 90 days past due, or just 90+dpd, refers to the customers' payment status, which is known after the performance period is over.³ This definition is aligned with the Basel II regulatory framework (Anderson, 2007).

Let $C_j = \{c_{j,1}, c_{j,2}, \dots, c_{j,n_j}\}$ be the j 'th set of customers with class labels $Y_j = \{y_{j,1}, y_{j,2}, \dots, y_{j,n_j}\}$. Hence,

$$dr_{C_j} = \frac{\sum_i^{n_j} [y_{j,i} = 1]}{n_j}, \quad (9)$$

where $[\cdot]$ is the Iverson bracket, is the default rate of the j 'th group of customers.

Given that $dr_{C_j} > dr_{C_l}$, we say that the group C_j has lower creditworthiness compared to group C_l . In other words, customers in C_j have, on average, higher probability of default. Therefore, in order to identify L segments with a different propensity to fall into financial distress, we need to find segments where the average probability of default is different from the rest of the groups. Mathematically, we want to learn a data representation that satisfies

$$dr_{C_j} \neq dr_{C_l}, \quad \text{for } j, l = 1, 2, \dots, L \text{ and } j \neq l. \quad (10)$$

Now it should be clear that the data transformation $f(\cdot)$ that we are looking for, needs to incorporate the class label y . In this way, the latent space in the VAE should generate codes that also contain information about y . Otherwise, those codes will fail to reproduce $f(x|y)$.

One such transformation is the Weight of Evidence⁴ (WoE) (Anderson, 2007; Siddiqi, 2012), which is defined as

$$\log \frac{Pr(x|y=0)}{Pr(x|y=1)}. \quad (11)$$

³ The performance period is the time interval in which if customers are at any moment 90+dpd, then their ground truth class is $y = 1$. Frequently, 12 and 24 months are time intervals used for the performance period. Further, the performance period starts at the moment an applicant signs the loan contract.

⁴ Originally, the WoE was introduced by Irving John Good in 1950 in his book *Probability and the Weighing of Evidence* and it has been used in the logistic regression and Naïve Bayes literature, among others.

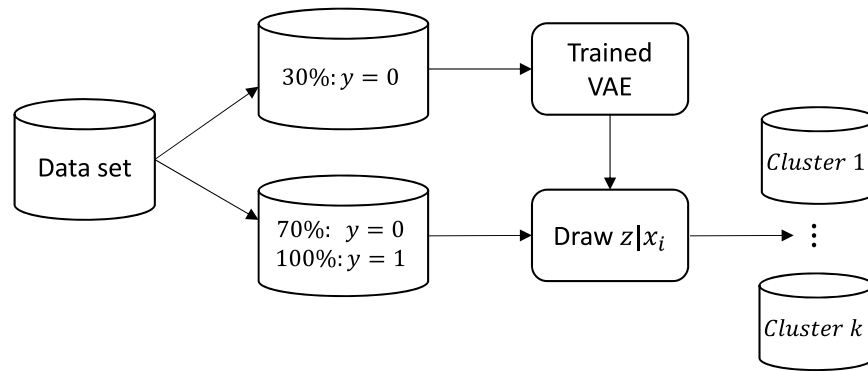


Fig. 2. Development methodology: We use 30% of the majority class data for training the VAE. Once it is trained, we generate the latent variables for the remaining data.

4.1. The weight of evidence

The WoE transformation has been used in credit scoring for a long time (Abdou, 2009), and it has become the standard in credit scoring models. The way to estimate it, given that the m 'th feature x_m is continuous, is by dividing its values into K bins B_1, B_2, \dots, B_K . In the case of categorical variables, the different categories are already these bins. Hence, the WoE for the k 'th bin of the m 'th feature is

$$\begin{aligned} \text{WoE}_{k,m} &= \log \frac{\Pr(x_m \in B_k | y = 0)}{\Pr(x_m \in B_k | y = 1)} \\ &= \log \frac{\frac{1}{n} \sum_{i=1}^n [x_{i,m} \in B_{k,m} \text{ and } y_i = 0]}{\frac{1}{n} \sum_{i=1}^n [x_{i,m} \in B_{k,m} \text{ and } y_i = 1]}, \end{aligned} \quad (12)$$

where n is the total number of observations. Note that the number of bins can vary for different features. See chapter 16.2 in Anderson (2007) or chapter 6 in Siddiqi (2012) for more details. Table 1 shows the difference between fine and coarse classing. In the fine classing approach, we create K bins, which provide the finest granularity. Then, fine bins with similar risk are binned into smaller groups resulting in the coarse classing. See Anderson (2007) for more details.

We use the coarse classing WoE transformation⁵ of the input data x to tilt the latent space in the VAE towards configurations which encapsulate the propensity to fall into financial distress.

5. Experiments and results

Our goals in this section are: (i) to show how can we reveal the natural clustering structure of financial data, which is unknown a priori, using our proposed framework, (ii) to analyze some of the properties of the learned data representations with our method, (iii) to benchmark our proposed methodology with some well-known representation learning methodologies, such as PCA (Pearson, 1901), kernel PCA (Schölkopf et al., 1998), isomaps (Tenenbaum et al., 2000), and t-SNE (Hinton & Roweis, 2003), and (iv) to show that our proposed supervision stage is able to steer representation learning for bank data.

5.1. Data description

We use real data sets from three different geographical regions, reflecting different market and economic conditions, allowing us to test the generalization properties of our proposed methodology for data representations of bank customers. The data set used in this section are a Norwegian and a Finnish car loan data set provided by Santander Consumer Bank Nordics and a public data set used in the Kaggle competition *Give me some credit*.⁶ These data sets show applicants'

status, financial and demographic factors at the time of application as well as the class label. The performance period for the real data sets is 12 months, while for the public data set it is 24 months. More details about the data sets can be found in Tables A.1, A.3 and A.4.

5.2. Training the VAE and generating latent representations

We train the VAE using the WoE as the input data, and using only the majority class ($y = 0$) data. The reason is because we want to have a robust estimate for the default rate in the data representation that we are learning. In addition, using observations from the minority class ($y = 1$) did not change the data representation in the latent space in our experiments, which is probably explained by the strong class imbalance in the three data sets.

Hence, we use 30% of the majority class to train the VAE. During training, we generate the latent space for the remaining 70% of the majority class and 100% of the minority class data, see Fig. 2. Based on the optimization of the ELBO, together with a heuristic visual comparison of the latent space in the training and test data sets, we select the optimal network architecture as well as the stopping criteria. It is worth mentioning that we observe that the shapes and proportions of the clusters in the training data are similar to the ones in the test data. This is a good indicator that our proposed model learns data representations that generalizes to unseen customers.

The VAE architectures that we tested are shown in Table A.2, and the final architecture IDs that we use are arch4, arch4 and arch1 for the Norwegian, Finnish and Kaggle data sets respectively. In addition, we use tanh activations in all hidden layers, linear and sigmoid activations in the μ output layer for the encoder and decoder respectively, and linear activations in all $\log \sigma^2$ layers.⁷ The MLP models are trained with the adagrad optimizer (Duchi et al., 2011) using constant 0.01 learning rate and 0.001 momentum.

Finally, we use the expectation over the latent space for the i 'th customer

$$E[z|x_i] = \int_{-\infty}^{\infty} z q_{\phi}(z|x_i) dz = \mu_{z|x} \quad (13)$$

as the data representation of bank customers in the latent space of the VAE. Note that it is simply the output in the encoder MLP network, see Eq. (7). We also tried the Monte Carlo version of Eq. (13) using 100 samples of z_i and Eq. (7) and the results do not change. At this point it is worth mentioning that our proposed methodology only utilizes the class label y during the supervision stage. After our model has been trained, we are able to transform the input data for new customers into the WoE and then transform the WoE into a data representation that lies in one of the clusters in the latent space.

⁵ We will simply call it as WoE in the remaining of the paper for brevity.

⁶ Website <https://www.kaggle.com/c/GiveMeSomeCredit>.

⁷ We need to use different activation functions depending on the kind of variable that the MLP is handling. See chapter 6 in Goodfellow et al. (2016) for more details.

In what follows, we provide some practical considerations to train a VAE with our proposed methodology that help to reveal useful data representations for bank customers:

- Create business intuitive and monotonic coarse classing WoE, and make sure that the final WoE groups are not small.
- Tune the hyperparameters of the model by grid search.
- Choose simple network architectures for the encoder and decoder.
- During training, plot the latent space often, e.g. every fifth epoch, for both the training and testing data set.
- Based on the optimization of the lower bound and the data representation for the test data set decide a stopping criteria.

To further analyze the learned data representation of our method, we assign labels to the structure in the two-dimensional latent space. This task can be done manually using a set of *if/else* rules given the well-defined clustering structure in the latent space. However, we propose an automated version, which is presented in Algorithm 1. The idea is to use the hierarchical clustering algorithm iteratively, generating only two clusters in each iteration. Always preserving the clustering structure in the learned data representation. For this purpose, Algorithm 1 specifies the minimum number of observations in each cluster, denoted by n_{min} . Similarly, the minimum Euclidean distance between the centroids in the two clusters needs to be specified, and it is denoted by ρ . These two parameters are data dependent and should be selected in such a way that Algorithm 1 assigns cluster labels preserving the clustering structure in the latent space. The advantage of this approach is that the labeling happens automatically while we train the VAE. Finally, the results of our proposed data representations are shown in Fig. 3 and Table 2.

5.3. Properties of the learned data representations

The first important result to highlight is that using the WoE transformation we learned a data representation with well-defined clusters in the latent space for all three data sets. Analyzing the Norwegian car loan data, we see that about 82% of all customers are in cluster 3, which is the cluster with the smallest default rate. This makes sense since the data set contains only 2 557 customers from the minority class. On the other hand, about 18% of the customers are in clusters with relatively high default rate. Finally, we check whether the default rates are significantly different using the 99% confidence interval for a binomial variable, i.e. $\hat{p} \pm 2.57 \sqrt{\hat{p}(1 - \hat{p})/n}$, where \hat{p} is the default rate in each cluster, 2.57 is the corresponding critical value and n is the total number of observations in the cluster. With the exception of clusters 1 and 5, the default rate for the other clusters are statistically different. See Table 2.

For the Finnish and Kaggle data sets we observe the same pattern. The majority of the customers are in the cluster with the smallest default rate. However, about 10% of the customers in the Kaggle data are in three clusters with very high default rates. Note that all default rates in the Kaggle data are significantly different. On the other hand, the confidence intervals for the default rates in cluster 1 and 2 for the Finnish data set overlap each other. This is driven by the relatively small number of defaults in the cluster 1, which increases the variance of their default rate estimate.

We estimate the default probability for customers in the Norwegian data set using three different input data: (i) the learned data representation of the VAE, (ii) the WoE transformation, and (iii) the raw data. We use 70% of the data to estimate the default probability of the remaining 30% of the data. Further, we use the trained VAE from Section 5.2 to generate the latent space of the customers for whom we estimated the default probability. In Fig. 4, we show the learned data representation for these customers and we use the three estimated values for the default probability to create a colormap. It is interesting to see that the default probability estimated with the learned data representation reveals a smooth color transition. On the other hand,

Algorithm 1: Labeling the latent data representation of bank customers.

```

Input :  $z, n_{min}, \rho$ 
Output: cluster labels
1 pending_data = {z};
2 labels = ones(length(z));
3 while EOF(pending_data) == FALSE do
4   for item in pending_data do
5     labels = HierarchicalAlgorithm(pending_data[item], k =
6       2);
7     get centroids c1 and c2;
8     split pending_data[item] into C1 and C2 using labels;
9     if  $n1 > n_{min}$  AND  $n2 > n_{min}$  AND  $\|c1 - c2\| > \rho$  then
10      update labels;
11      pending_data.append = {C1,C2}
12    end
13  end
14 return labels

```

when the default probability is estimated with the WoE or with the raw data, the colormaps show a relatively random pattern. This result shows that our proposed method is not only able to learn a data representation of customers data, which shows a well-defined clustering structure and captures the customers' creditworthiness, but which also ranks the default probability across the two dimensions of the latent space.

The well-defined clustering structure of the data representation in the latent space and its ability to capture customers' creditworthiness, allows our proposed method to generate good representations. These representations express general priors that are particularly useful in the bank industry. Specifically, our proposed data representation is able to learn the natural clustering and spatial coherence of creditworthiness in the customers data.

5.4. Representation learning benchmark

We use the k-means (Lloyd, 1982), affinity propagation (Frey & Dueck, 2007), hierarchical (Ward Jr, 1963), birch (Zhang et al., 1996) and GMM algorithms to cluster the WoE transformations for the Norwegian data set in the original input space. Note that for all of these algorithms, the number of cluster must be specified. Hence, we use as input parameter the number of cluster suggested by our proposed methods, which is 5 for the Norwegian data set, to make results comparable.

After we cluster the Norwegian WoE data using the original input space, we apply different dimensionality reduction techniques to visualize the clusters that we obtained in a two dimensional space. Specifically, we use PCA, kernel PCA, isomaps, and t-SNE for dimensionality reduction. Fig. 5 shows both the clusters obtained in the first step, represented by different colors, and the two dimensional data representation that we obtained. As can be seen from the figure, none of the existing state-of-the-art data representation methods that we benchmark are able to generate a well-defined clustering structure.

Note that we could use the representation learning algorithms in the original input space and then cluster the data representation obtained. However, this ordering does not change the learned data representation of the input data, it will only change the clustering results in the two dimensional space. Given that none of the representation learning methods that we benchmark are able to generate a well-defined structure, the clustering algorithms will never find such non-overlapping clusters as the ones that we obtain with our proposed method.

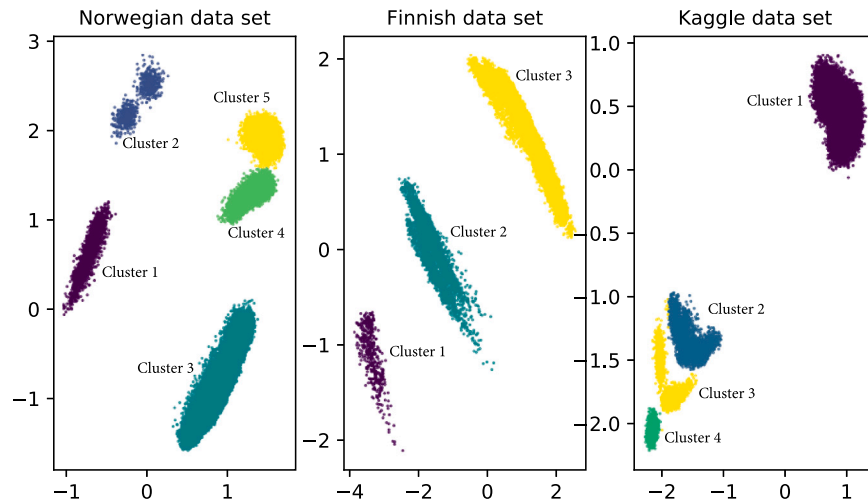


Fig. 3. Latent representation of bank customers.

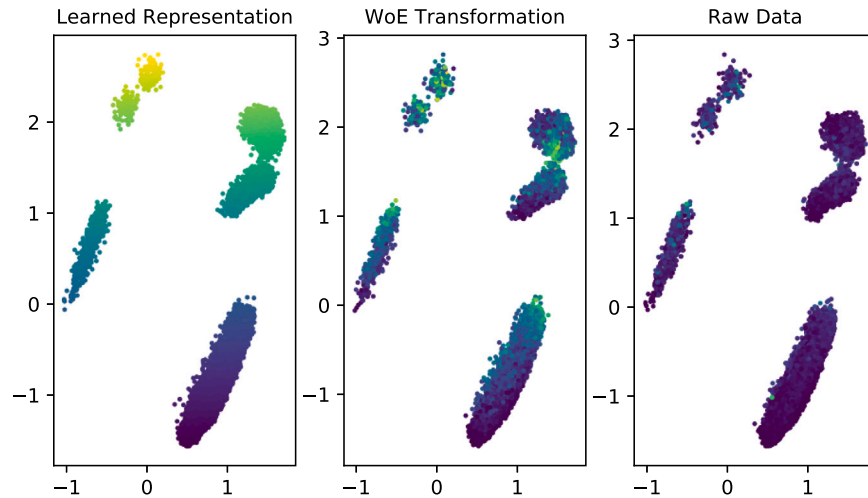


Fig. 4. Best viewed in color. We estimate the default probability for 30% of customers in the Norwegian data using the learned representation, WoE, and raw data. Further, we generate the latent space for these customers using the trained VAE. Finally, we use the three estimate values for default probability to create a colormap. Note that the left panel shows a smooth color transition.

Table 2

Default rates dr_{C_j} for the different clusters C_j in the data representation of bank customers are given in the first column. The 99% confidence intervals (CI) are shown in the second column for each data set, where non-overlapping lower bounds are denoted outside parenthesis and non-overlapping upper bounds are within parenthesis. Note that for the cluster with the lowest (highest) default rate we do not need to verify the lower (upper) bound. Finally, the total number of customers and the number of default customers are shown in the third and last column respectively.

Cluster	Norwegian data set				Finnish data set				Kaggle data set			
	dr	CI	Obs	$y = 1$	dr	CI	Obs	$y = 1$	dr	CI	Obs	$y = 1$
1	5.30%		2 206	117	5.93%		438	26	5.47%	(***)	97 434	5 327
2	11.24%	***	774	87	3.76%	***	6 067	228	33.13%	***(***)	6 121	2 028
3	1.39%	(***)	109 969	1539	0.91%	(***)	75 536	685	55.68%	***(***)	2 026	1 128
4	2.89%	***(***)	11 450	331					63.58%	***	2 427	1 543
5	5.30%		9 106	483								

5.5. Grouping of the input data

Now we want to show that the supervision step in our proposed methodology has valuable information about the input data, which is captured in well-defined clustering structures. Hence, we use different data transformations and, for each of these transformations, we train a new VAE, i.e. for each data transformation we learn a data representation using the same network architectures as the ones used to learn the data representations in Fig. 3. Specifically, we generate the latent space for the following data transformations:

1. PCA: The input data is transformed using principal component analysis with all the principal components, i.e. there is no dimensionality reduction.
2. Standardization: The input data is standardized by removing the mean and scaling to unit variance.
3. Fine classing WoE: The input data is transformed into WoE by creating bins with an approximately equal number of customers, i.e. no coarse classing is done.
4. Input data: Raw data without any transformation.

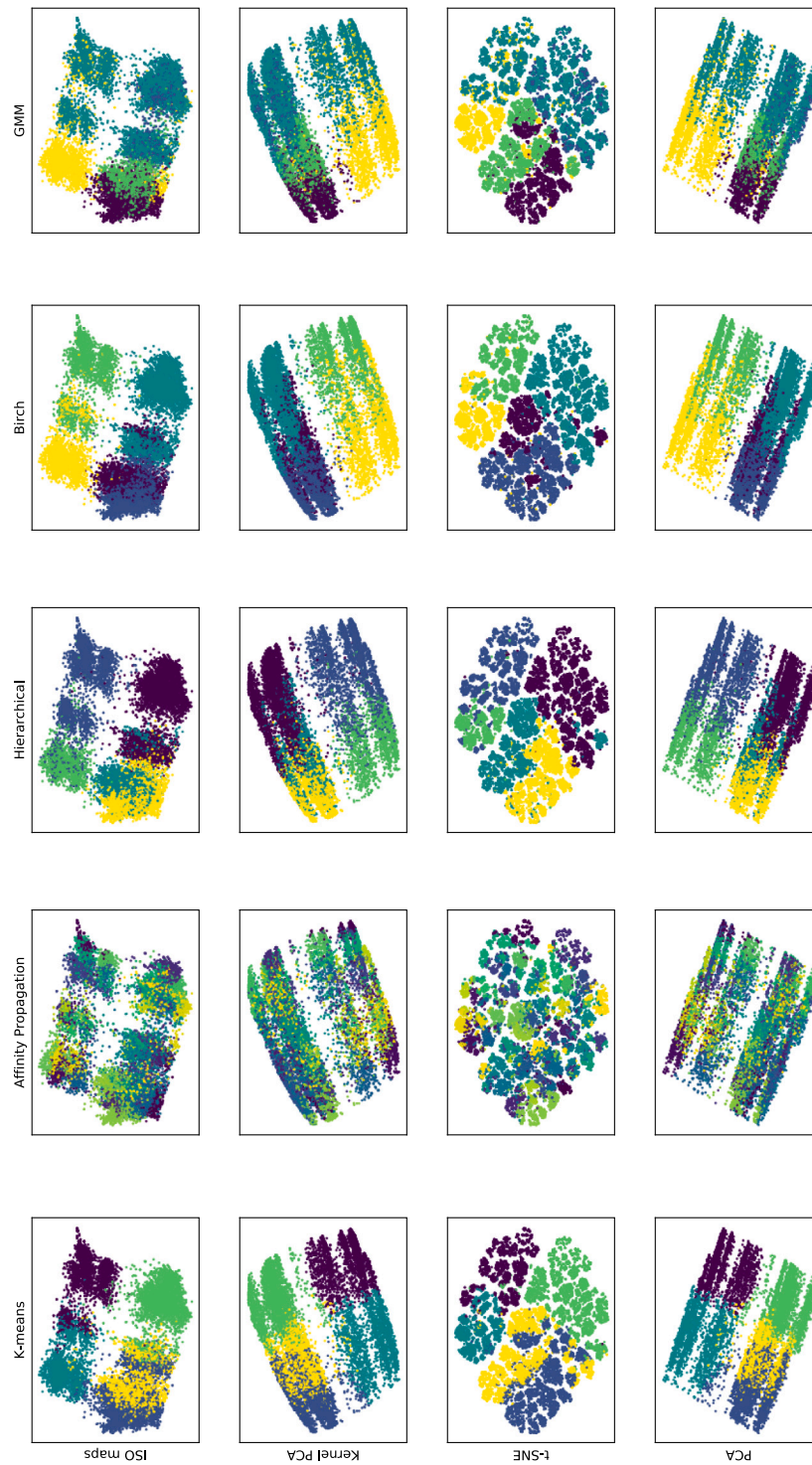


Fig. 5. Best viewed in color. We use the k-means, affinity propagation, hierarchical, birch, and GMM algorithms to cluster the WoE transformations for the Norwegian data, specifying five clusters. Then, we reduce the original dimensional space for the WoE to two dimensions using isomaps, kernel PCA, t-SNE and PCA. Cluster labels are given by the colors.

Fig. 6 shows the resulting latent spaces for the data transformations explained above. Interestingly, three of these transformations do not show any clustering structure at all. For the standardized transformation, the clusters have practically the same default rate. Hence, by identifying appealing data transformations and a useful grouping of the input data, it is possible to steer data representations in the latent space of the VAE. In this particular case, these representations are well-defined clusters with considerably different risk profiles.

Note that the right-most scatter in Fig. 6 represents the latent space for a traditional training approach where the input data is fed into the encoder network without any manipulation. It should be clear now that the supervision stage in our proposed methodology, together with the WoE transformation that encapsulates customers' creditworthiness, makes it possible to steer data representations in the latent space of the VAE. As a result, we obtain clusters with different risk profiles, which are unknown a priori and cannot be identified in the input space.

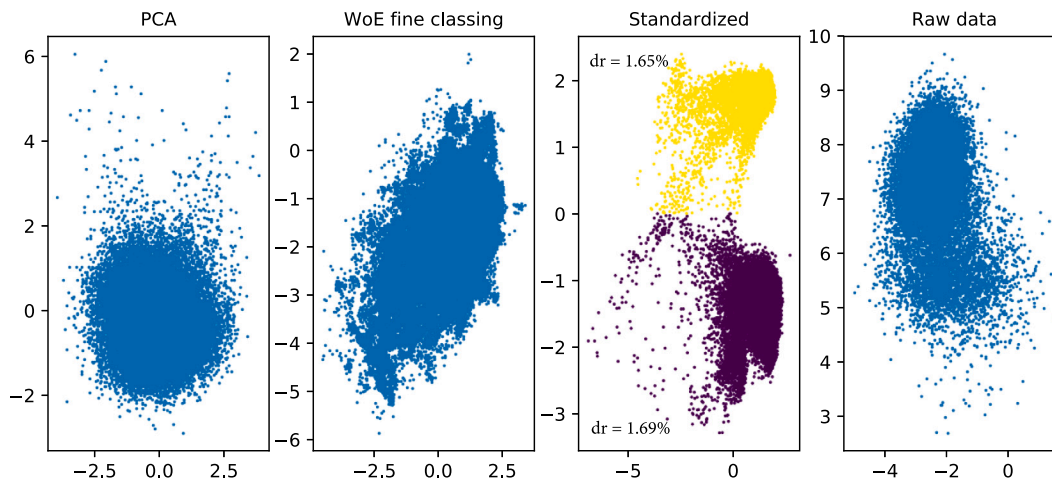


Fig. 6. Latent space for four different data transformation for the Norwegian data set. The left panel shows a PCA transformation (preserving the original data dimensionality). The second panel shows the latent space for the fine classing WoE transformation. The third panel shows the latent space for the standardized data, and finally, the right panel shows the latent space for the raw data. Standardizing the data reveals two clusters in the latent space. However, these clusters have practically the same default rate (dr). The other three transformations do not show any clustering structure.

6. Business application

In this section, we present two business applications of our proposed methodology for learning data representations of bank customers. First, we identify the salient dimensions in the clustering structure for the Norwegian data set and use those salient dimensions to obtain descriptive labels associated with each cluster. These labels can be used to find out which customers should be the target of a marketing campaign to sell a new product, for example. Second, we show how banks can improve the assessment of customers' creditworthiness using the clusters identified by our proposed method for the Norwegian data set.

6.1. Customers profiles

After the VAE model has been trained with our proposed methodology, we are able to map the input data for new customers into one of the clusters in the latent space, which have different risk profiles (see Table 2). Further, we assign descriptive labels to the clusters in the latent space so they can be used for marketing or product offering purposes. To that end, we adopt the salient dimension methodology presented in Azcarraga et al. (2005) and explained in Appendix B. This approach identifies features whose values are statistically significant in different clusters, and are called salient dimensions. In what follows, we analyze the salient dimensions for the Norwegian data set, and salient dimensions for the Kaggle and Finnish data sets can be found in Table A.5.

The first interesting result in Fig. 3 is the pattern of the latent variables for clusters 1 and 5 (both clusters have default rate = 5.30%), which are located on opposite sides of the two-dimensional space. The salient dimension *MaxBucket12* in cluster 1 shows that about 70% of the customers were between 30 and 60 days past due at the moment they applied for a new loan, i.e. they are existing customers applying for a new loan. Actually, all customers in cluster 1 are existing customers who are at least 30 days past due. On the other hand, about 51% of the customers in cluster 5 are new applicants willing to buy middle-age cars. Cluster 2 is actually composed only by existing customers, i.e. new applicants lie on the right side of Fig. 3, while existing customers on the left hand side. Therefore, we can label cluster 1 as *existing customers in arrears applying for a car loan* and cluster 5 as *new applicants and existing customers in arrears applying for a loan to by a middle-age car*.

Now let us see what characterizes cluster 3, which is the cluster with the lowest default rate. Looking at one of its salient dimension namely *DownPayment%*, we can see that the average down payment

in this cluster is about 20%. On the other hand, the average down payment for the rest of the clusters is less than 12%. In the bank industry, high down payments are linked to low default rates. Further, the salient dimension *AgeObject* shows that about 35% of the customers in cluster 3 are applying to buy relatively new cars. In contrast, the average percentage of customers applying to buy new cars, in the other clusters, is only 23%. Buyers of new cars are also associated with low default rates by bank risk analysts. Hence, we could label cluster 3 as *new applicants willing to buy new cars with high down payment amount*.

Cluster 2 has the highest default rate and can be explained by its salient dimension *MaxBucket12*. About 93% of customers in this cluster are between 1 and 90 days past due, while the percentage of customers in the other clusters in the same interval is only 15%. This cluster could have the label *existing customers in high arrears level*.

6.2. Improving customers' creditworthiness

To show whether the clustering structure revealed by our proposed methodology can improve the assessment of customers' creditworthiness, we train one multilayer perceptron for each of the 5 clusters found in the Norwegian data set (see Fig. 3). We use the WoE as input features and we divide the data set for each cluster in 70% for training and 30% testing. Further, we compare the classification performance of the segment-based strategy with the traditional credit scoring approach where only one classifier is trained for the whole data set. To train the classical credit scoring model, we also use the WoE as input features and for the training data we aggregate all training data sets for the 5 clusters in the segment-based approach. Similarly, we test model performance of the classical credit scoring model on each of the 5 test data sets for the segment-based approach. Table 3 shows the values of 4 different performance metrics, which are commonly used in the financial industry to measure the discriminative power of credit scoring models, obtained for our approach and the standard credit scoring approach. By using our proposed methodology to identify clusters that encapsulated customers' creditworthiness and developing segment-based classifiers, banks can improve the assessment of creditworthiness.

7. Conclusion

In this paper, we show that it is possible to steer data representations in the latent space of the Variational Autoencoder using a semi-supervised learning framework and a specific grouping of the

Table 3

Model performance for the segment-based and the classical credit scoring approach. Performance values are the average of a 10-cross-validation.

Norwegian data set			
Performance metric	Cluster	Segment-based	Portfolio-based
Kolmogorov–Smirnov	1	0.4648	0.4098
	2	0.3441	0.3280
	3	0.4318	0.4199
	4	0.3821	0.3489
	5	0.3410	0.3299
Gini coefficient	1	0.5377	0.4860
	2	0.3582	0.3402
	3	0.5511	0.5412
	4	0.4790	0.4377
	5	0.4043	0.3846
H-measure	1	0.2774	0.2310
	2	0.1665	0.1453
	3	0.2174	0.2076
	4	0.1760	0.1471
	5	0.1302	0.1193
AUC	1	0.7688	0.7430
	2	0.6791	0.6701
	3	0.7756	0.7706
	4	0.7395	0.7188
	5	0.7021	0.6923

input data. We show that the Weight of Evidence transformation encapsulates the propensity for financial distress and by training a VAE with our proposed methodology we can learn a latent data representation that captures the natural clustering of the data and encapsulates the customers' creditworthiness.

The data representations generated with our proposed methodology have certain features that are particularly useful in the bank industry. The representations are not only able to learn the natural clustering of the data, which is unknown a priori and cannot be identified in the input space, but also the spatial coherence of customers' creditworthiness.

The main advantages of our proposed method are that it captures the natural clustering of the data, suggests the number of clusters, captures the spatial coherence of customers' creditworthiness, generates data representations of unseen customers, and assigns them to one of the existing clusters. Finally, our empirical results, based on real data sets reflecting different market and economic conditions, show that none of the well-known data representation models in the benchmark analysis are able to obtain well-defined clustering structures. Further, we show how banks can use our proposed methodology to improve marketing campaigns and credit risk assessment.

CRedit authorship contribution statement

Rogelio A. Mancisidor: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Michael Kampffmeyer:** Conception and design of study, Writing - original draft, Writing - review & editing. **Kjersti Aas:** Conception and design of study, Writing - original draft, Writing - review & editing. **Robert Jenssen:** Conception and design of study, Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank Santander Consumer Bank for financial support and the real data sets used in this research. This work was also supported by the Research Council of Norway [grant number 260205] and SkatteFUNN, Norway [grant number 276428]. All authors approved the final version of the manuscript.

Appendix A. Figures and tables

Table A.1

Summary of the three data sets used in the different experiments in this paper. Default rate for the j 'th set of customers is defined as $dr_C = \frac{\sum_j |y_j|=1}{n_j}$, where n_j is the total number of customers and y_j is the class label.

Name	Cases	Features	Default rate
Norwegian data set	187 069	20	0.0137
Finnish data set	115 899	12	0.0081
Give me some credit	150 000	10	0.0668

Table A.2

Different architectures tested to train the VAE for the three different data sets. More complex architectures, with more hidden layers and different dimension in the latent spaces, were also tested. However, for the data sets under analysis relative complex architectures do not add any significant value.

Architecture ID	z dimension	Hidden layers	Neurons	Learning rate	Epochs
arch1	2	1	5	0.01	50
arch2	2	1	10	0.01	50
arch3	2	1	20	0.01	50
arch4	2	1	30	0.01	50
arch5	2	1	40	0.01	50
arch6	2	1	50	0.01	50
arch7	2	1	60	0.01	50
arch8	2	1	70	0.01	50
arch9	2	1	30	0.007	50
arch10	2	1	30	0.008	50
arch11	2	1	30	0.009	50
arch12	2	1	30	0.011	50
arch13	2	1	30	0.012	50
arch14	2	1	30	0.013	50
arch15	5	1	30	0.01	50
arch16	10	1	30	0.01	50
arch17	15	1	30	0.01	50
arch18	2	2	30	0.01	50
arch19	2	3	30	0.01	50
arch20	2	4	30	0.01	50
arch21	2	5	30	0.01	50

Table A.3

Variable name and description for all features in the Norwegian car loan data set.

Norwegian data set	
Variable name	Description
BureauScoreAge	Matrix with bureau scores and applicants age
NetincomeStability	Net income stability index
RiskBucketHistory	Delinquency history
NumApps6M	Number of applications last 6 months
ObjectGroupCarMake	Car brand in the application
DownPaymentAgeObject	Matrix with down payment and car model year
CarPrice	Car price
NetIncomet0t1	Change in applicant's net income
MaxBucketSnapshot	Delinquency at the time of application
MaxMoB12	Months on books at the time of application
NetIncomeTaxt0	Ratio between net income and taxes
AgeObject	Car model year
AgePrimary	Age of primary applicant
BureauScoreUnsec	Bureau score unsecured
DownPayment	Own capital
MaxBucket12	Maximum delinquency in the past 12 months
TaxAmountt0	Tax amount paid
BureauScore	Bureau score generic
Taxt0t1	Change in applicant's taxes
Netincomet0	Net income at the time of application

Table A.4
Variable name and description of all features in the Kaggle and Finnish data set data sets.

Kaggle	
Variable name	Description
RevolvingUtilizationOfUnsecuredLines	Total balance on credit lines
AgePrimary	Age of primary applicant
NumberOfTime3059DPD	Number of times borrower has been 30–59 dpd
Monthly debt payments divided by monthly gross income	Marital Status
Income	Monthly Income
NumberOfOpenCreditLines	Number of loans or credit cards)
NumberOfTimesDaysLate	Number of times borrower has been 90 dpd
NumberRealEstateLoansOrLines	Number of mortgage loans
NumberOfTime6089DPD	Number of times borrower has been 60–89 dpd
NumberOfDependents	Number of dependents in family
Finnish data set	
AgePrimary	Age of primary applicant
AgeObjectContractTerm	Matrix with car model year and number of terms
DownPayment	Own capital
Marital Status	Debt Ratio
MaxBucket24	Maximum delinquency in the past 24 months
MonthsAtAddress	Number of months living at current address
Number2Rem	Number of 2nd reminders last year
NumberRejectedApps	Number of rejected applications
ObjectPrice	Car price
ResidentialStatus	Whether the applicant owns a house
ObjectMakeUsedNew	Matrix with car make and whether it is new or used
EquityRatio	Debt to equity

Table A.5
Statistically significant salient dimensions for the Norwegian, Kaggle and Finnish data set. We use $s.d. = 1$ to define salient dimensions.

Norwegian data set		Kaggle		Finnish data set	
Cluster	Salient Dimension	Cluster	Salient Dimension	Cluster	Salient Dimension
1	MaxBucket12	1	NumberOfTime3059DPD	1	AgePrimary
2	NetIncomet0t1	1	NumberOfTimesDaysLate	1	Number2Rem
2	MaxBucket12	1	NumberRealEstateLoansOrLines	1	NumberRejectedApps
3	AgeObject	1	NumberOfTime6089DPD	2	Number2Rem
3	NetIncomet0t1	2	RevolvingUtilizationOfUnsecuredLines	2	NumberRejectedApps
3	Taxt0t1	2	DebtRatio	3	DownPayment
3	DownPayment	3	NumberOfTime3059DPD	3	ResidentialStatus
4	NumApps6M	3	NumberOfTime6089DPD		
4	AgeObject	3	NumberOfDependents		
4	NetIncomet0t1	4	NumberOfTime3059DPD		
4	Taxt0t1	4	NumberOfTimesDaysLate		
4	DownPayment	4	NumberOfTime6089DPD		
5	AgeObject				
5	NetIncomet0t1				
5	DownPayment				

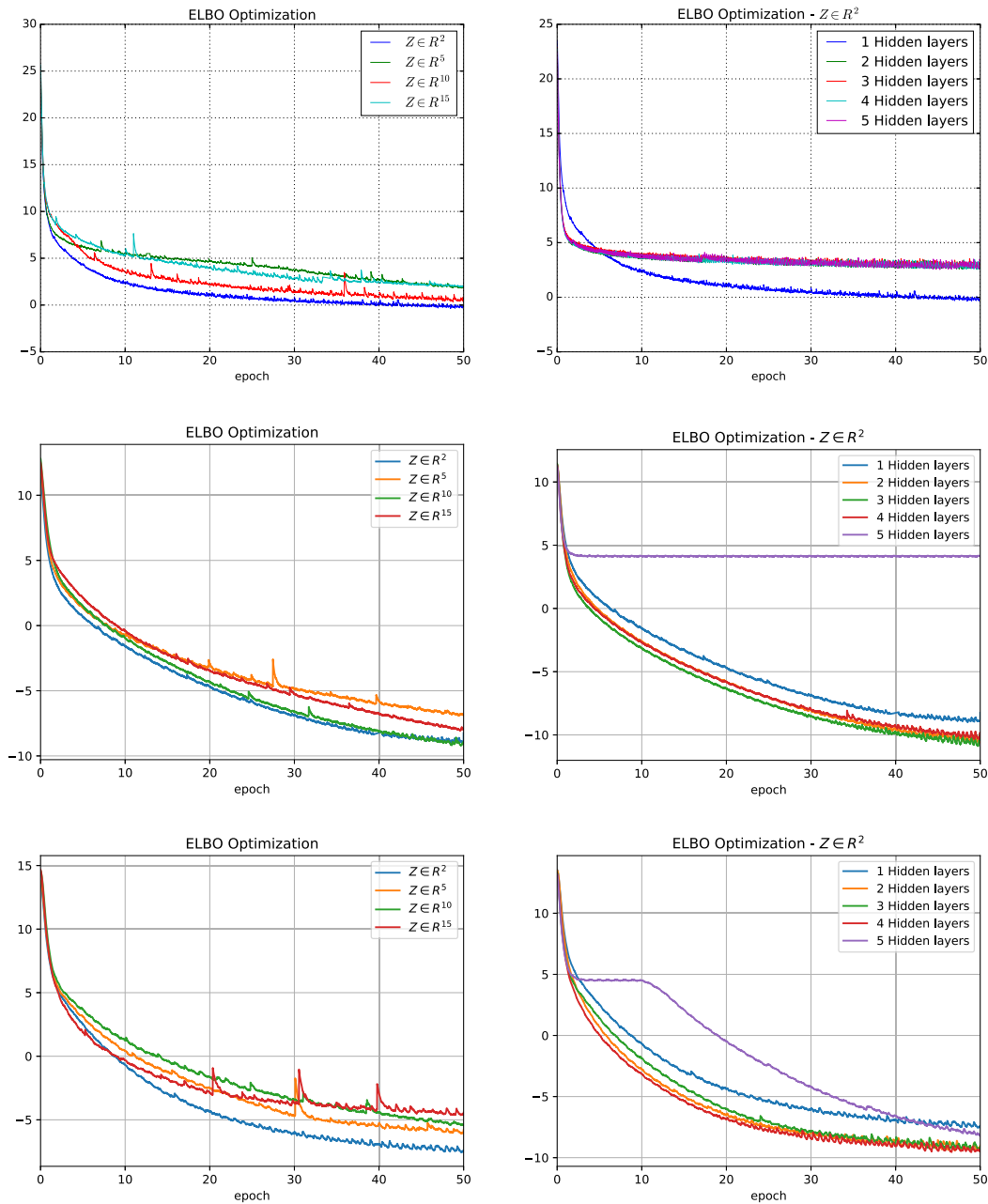


Fig. A.1. Panels to the left show the optimization of the negative ELBO for different dimensionalities in the latent space. For $z \in \mathbb{R}^2$, the AEVB algorithm converges faster to the optimal variational density $q^*(z)$ for all data sets (Norwegian data set top-left panel, Kaggle data set middle-left and Finnish data set bottom-left panel). Further, panels to the right also show the optimization of the ELBO but for $z \in \mathbb{R}^2$ and for a different number of hidden layers. The VAE for the Norwegian data set (top-right panel) with 1 hidden layer converges faster to $q^*(z)$. For the Kaggle data set (middle-right panel), 2–4 hidden layers converge faster to the optimal variational density. However, the resulting clustering structure in the latent space contains only two clusters. Similarly, for the Finnish data set (bottom-right panel) 2–4 hidden layers makes the algorithm converge faster. However, the resulting clustering structure contains four clusters. For this data set, it is not optimal to have four clusters.

Appendix B. Salient dimensions

Let v be the v 'th dimension of the i 'th vector $x_{i,v}$, where $x \in \mathbb{R}^\ell$. Further let $\Phi_{in}(k)$ be the set of in-patterns (within cluster k) and $\Phi_{out}(k)$ be the set of out-patterns (not within cluster k). Then compute the mean input values

$$\mu_{in}(k, v) = \frac{\sum_{x_i \in \Phi_{in}(k)} x_{i,v}}{|\Phi_{in}(k)|}, \quad (14)$$

$$\mu_{out}(k, v) = \frac{\sum_{x_i \in \Phi_{out}(k)} x_{i,v}}{|\Phi_{out}(k)|}, \quad (15)$$

where $|\{\cdot\}|$ returns the cardinality of $\{\cdot\}$. Further, compute the difference factors

$$df(k, v) = \frac{\mu_{in}(k, v) - \mu_{out}(k, v)}{\mu_{out}(k, v)}, \quad (16)$$

and their mean and standard deviations

$$\mu_{df}(k) = \frac{1}{\ell} \sum_v df(k, v), \quad (17)$$

$$\sigma_{df}(k) = \sqrt{\sum_v (df(k, v) - \mu_{df}(k))^2 / \ell}. \quad (18)$$

Finally, we say that the v 'th feature in cluster k is a salient dimension if

$$df(k, v) \leq \mu_{df}(k) - s.d. \cdot \sigma_{df}(k), \quad (19)$$

or

$$df(k, v) \geq \mu_{df}(k) + s.d. \cdot \sigma_{df}(k), \quad (20)$$

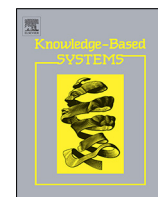
where $s.d.$ is the number of standard deviations to be used. The value for $s.d.$ is defined based on the data set. We use $s.d. = 1$ for all three data sets under analysis.

References

- Abdou, H. A. (2009). Genetic programming for credit scoring: The case of Egyptian public sector banks. *Expert Systems with Applications*, 36(9), 11402–11417.
- Anderson, R. (2007). *The credit scoring toolkit*. Oxford University Press.
- Aurifeille, J.-M. (2000). A bio-mimetic approach to marketing segmentation: Principles and comparative analysis. *European Journal of Economic and Social Systems*, 14(1), 93–108.
- Azcarraga, A. P., Hsieh, M.-H., Pan, S. L., & Setiono, R. (2005). Extracting salient dimensions for automatic SOM labeling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4), 595–600.
- Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798–1828.
- Bijak, K., & Thomas, L. C. (2012). Does segmentation always improve model performance in credit scoring?. *Expert Systems with Applications*, 39(3), 2433–2442.
- Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518), 859–877.
- Bouchacourt, D., Tomioka, R., & Nowozin, S. (2018). Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In Thirty-second AAAI conference on artificial intelligence.
- Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space. arXiv preprint arXiv:1511.06349.
- Doersch, C. (2016). Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908.
- Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)*, 12(Jul), 2121–2159.
- Frey, B. J., & Dueck, D. (2007). Clustering by passing messages between data points. *Science*, 315(5814), 972–976.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press, <http://www.deeplearningbook.org>.
- Hand, D. J., Sohn, S. Y., & Kim, Y. (2005). Optimal bipartite scorecards. *Expert Systems with Applications*, 29(3), 684–690.
- Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., & Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In International conference on learning representations.
- Hinton, G. E., & Roweis, S. T. (2003). Stochastic neighbor embedding. In *Advances in neural information processing systems* (pp. 857–864).
- Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1), 55–67.
- Hsu, W.-N., Zhang, Y., & Glass, J. (2017). Learning latent representations for speech generation and transformation. arXiv preprint arXiv:1704.04222.
- Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
- Latif, S., Rana, R., Qadir, J., & Epps, J. (2017). Variational autoencoders for learning latent representations of speech emotion. arXiv preprint arXiv:1712.08708.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436.
- Lim, M. K., & Sohn, S. Y. (2007). Cluster-based dynamic scoring model. *Expert Systems with Applications*, 32(2), 427–431.
- Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 129–137.
- Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11), 559–572.
- Rampasek, L., & Goldenberg, A. (2017). Dr. VAE: Drug response variational autoencoder. arXiv preprint arXiv:1706.08203.
- Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. arXiv preprint arXiv:1401.4082.
- Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5), 1299–1319.
- Siddiqi, N. (2012). *Credit risk scorecards: Developing and implementing intelligent credit scoring*. Vol. 3. John Wiley & Sons.
- Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems* (pp. 3483–3491).
- Su, J., Wu, S., Zhang, B., Wu, C., Qin, Y., & Xiong, D. (2018). A neural generative autoencoder for bilingual word embeddings. *Information Sciences*, 424, 287–300.
- Tenenbaum, J. B., De Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500), 2319–2323.
- Titus, A. J., Bobak, C. A., & Christensen, B. C. (2018). A new dimension of breast cancer epigenetics - applications of variational autoencoders with DNA methylation. In *Proceedings of the 11th international joint conference on biomedical engineering systems and technologies - Volume 4: BIOINFORMATICS* (pp. 140–145). SciTePress, INSTICC, ISBN: 978-989-758-280-6, <http://dx.doi.org/10.5220/0006636401400145>.
- Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301), 236–244.
- Way, G. P., & Greene, C. S. (2017a). Evaluating deep variational autoencoders trained on pan-cancer gene expression. arXiv preprint arXiv:1711.04828.
- Way, G. P., & Greene, C. S. (2017b). Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. *BioRxiv*.
- Xiao, H., Xiao, Z., & Wang, Y. (2016). Ensemble classification based on supervised clustering for credit scoring. *Applied Soft Computing*, 43, 73–86.
- Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. In *ACM sigmod record. Vol. 25* (pp. 103–114). ACM.
- Zhong, G., Wang, L.-N., Ling, X., & Dong, J. (2016). An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, 2(4), 265–278.

Chapter 10

Paper II



Deep generative models for reject inference in credit scoring[☆]

Rogelio A. Mancisidor^{a,b,d,*}, Michael Kampffmeyer^{a,d}, Kjersti Aas^c, Robert Jenssen^{a,d}

^a Department of Physics and Technology, Faculty of Science and Technology, UiT The Arctic University of Norway, Hansine Hansens veg 18, Tromsø 9037, Norway

^b Credit Risk Models, Santander Consumer Bank AS, Strandveien 18, Lysaker 1325, Norway

^c Statistical Analysis, Machine Learning and Image Analysis, Norwegian Computing Center, Gaustadalleen 23a, Oslo 0373, Norway

^d RAM, MK, and RJ are all with the UiT Machine Learning Group: <http://machine-learning.uit.no>, Tromsø, Norway



ARTICLE INFO

Article history:

Received 23 April 2019

Received in revised form 4 February 2020

Accepted 8 March 2020

Available online 12 March 2020

Keywords:

Reject inference

Deep generative models

Credit scoring

Semi-supervised learning

ABSTRACT

Credit scoring models based on accepted applications may be biased and their consequences can have a statistical and economic impact. Reject inference is the process of attempting to infer the creditworthiness status of the rejected applications. Inspired by the promising results of semi-supervised deep generative models, this research develops two novel Bayesian models for reject inference in credit scoring combining Gaussian mixtures and auxiliary variables in a semi-supervised framework with generative models. To the best of our knowledge this is the first study coupling these concepts together. The goal is to improve the classification accuracy in credit scoring models by adding reject applications. Further, our proposed models infer the unknown creditworthiness of the rejected applications by exact enumeration of the two possible outcomes of the loan (default or non-default). The efficient stochastic gradient optimization technique used in deep generative models makes our models suitable for large data sets. Finally, the experiments in this research show that our proposed models perform better than classical and alternative machine learning models for reject inference in credit scoring, and that model performance increases with the amount of data used for model training.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

Credit scoring uses statistical models to transform the customers' data into a measure of the borrowers' ability to repay the loan [1]. These models are developed, commonly, based on accepted applications because the bank knows whether the customer repaid the loan. The problem is that this data sample is biased since it excludes the rejected applications systematically. This is called selection bias.

Using a biased sample to estimate any model has several problems. The straightforward consequence is that the model parameters are biased [2], which has a statistical and economic impact [3,4]. Another consequence is that the default probability can be underestimated, affecting the risk premium and the profitability of the bank [5]. Hence, reject inference, which is the

process of attempting to infer the true creditworthiness status of the rejected applications [6], has created a great deal of interest.

There is a vast literature on reject inference using classical statistical methods. However, there has been little research using machine learning techniques (see Table 1). Semi-supervised learning approaches design and train models using labeled (accepted applications) and unlabeled data (rejected applications), and aim to utilize the information embedded in both data to improve the classification of unseen observations. There are several fields where semi-supervised deep generative models have achieved state-of-the-art results, e.g. in semi-supervised image classification [7,8], in semi-supervised sentiment analysis [9,10], and in unsupervised clustering [11]. Additionally, the useful information embedded in their latent space is well documented [12–15]. Inspired by the modeling framework introduced by [7], this research develops two novel models for reject inference models in credit scoring combining, for the first time, auxiliary variables [8] and Gaussian mixtures parameterized by neural networks in a semi-supervised framework.

Our proposed models have a flexible latent space, induced by the Gaussian mixtures, to improve the variational approximation and the reconstruction of the input data [8,31]. In addition, one of our models not only uses the input data to classify new loan applications, but also a latent representation of it. This makes the classifier more expressive [8,31]. We compare the

[☆] No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to <https://doi.org/10.1016/j.knosys.2020.105758>.

* Corresponding author at: Department of Physics and Technology, Faculty of Science and Technology, UiT The Arctic University of Norway, Hansine Hansens veg 18, Tromsø9037, Norway.

E-mail addresses: rogelio.a.mancisidor@uit.no (R.A. Mancisidor), michael.c.kampffmeyer@uit.no (M. Kampffmeyer), kjersti@nr.no (K. Aas), robert.jenssen@uit.no (R. Jenssen).

Table 1

Up to date research overview on reject inference. The scope of the research marked with * differs from ours, hence they are included in Section 2.

(Year) Author	Data type	Status of rejects	No. of accepts	No. of rejects	Reject Inference approach	Classification method
(1993) Joanes [16]	Artificial	Unknown	75	12	Reclassification	Logistic
(2000) Feelders [17]	Artificial	Unknown	Varying	Varying	EM	QDA, Logistic
(2001) Chen and Astebro [4]	Corporate	Known	298	599	Heckman's model	Probit, Bivariate probit
(2003) Banasik et al. [18]	Consumer	Known	8 168	4 040	Augmentation	Logistic, Probit
(2004) Crook and Banasik [19]	Consumer	Known	8 168	4 040	Augmentation, Extrapolation	Logistic
(2004) Verstraeten and Van den Poel [20]	Consumer	Partially known	38 048	6 306	Augmentation	Logistic
(2005) Banasik and Crook [21]	Consumer	Known	8 168	4 040	Augmentation	Logistic
(2006) Sohn and Shin [22]*	Consumer	Unknown	759	10	Reclassification	Survival analysis
(2007) Banasik and Crook [23]	Consumer	Known	8 168	4 040	Augmentation and Heckman's model	Logistic, Bivariate probit
(2007) Kim and Sohn [24]	Corporate	Known	4 298	689	Heckman's model	Bivariate probit
(2007) Wu and Hand [25]	Artificial	Known	Varying	Varying	Heckman's model	OLS, Bivariate Probit
(2010) Banasik and Crook [26]*	Consumer	Known	147 179	Varying	Augmentation	Survival analysis
(2010) Marshall et al. [5]	Consumer	Known	40 700	2 934	Heckman's model	Probit, Bivariate probit
(2010) Maldonado and Paredes [27]	Consumer	Known	800	200	Extrapolation	SVM
(2012) Chen and Astebro [28]	Corporate	Known	4 589	Varying	Bound and Collapse	Bayesian
(2013) B�ucker et al. [2]	Consumer	Unknown	3 984	5 667	Augmentation	Logistic
(2013) Anderson and Hardin [29]	Consumer	Unknown	3 000	1 500	Augmentation, EM	Logistic
(2016) Nguyen [3]	Consumer	Unknown	56 016	142 571	Augmentation, Extrapolation	Logistic
(2017) Li et al. [30]	Consumer	Unknown	56 626	563 215	Extrapolation	Semi-supervised SVM

performance of the semi-supervised generative models with a range of techniques representing the state-of-the-art in reject inference for credit scoring, including three classical reject inference techniques (reclassification, fuzzy parceling¹ and augmentation [32]), and three semi-supervised machine learning approaches (self-learning [33] MLP, self-learning SVM, and semi-supervised SVM [34]). Additionally, we include two supervised machine learning models (multilayer perceptron (MLP) [35] and support vector machine (SVM) [36]) to measure the marginal gain of reject inference.

To summarize, the main contributions of this paper are as follows:

1. We develop two novel reject inference models for credit scoring combining auxiliary variables and Gaussian mixtures in a semi-supervised framework with generative models for the first time.
2. We derive the objective functions for our proposed models and show how they can be parameterized by MLPs and optimized with stochastic gradient descent.
3. We parametrize the Gaussian mixtures using an MLP and we show how to train them with semi-supervised data.
4. Our empirical results show that our proposed models achieve higher performance compared to the state-of-art methods in credit scoring. Additionally, the model performance for our proposed models increases with the amount of data used for training.

The rest of the paper is organized as follows. Section 2 reviews the related work on reject inference in credit risk, then Section 3 presents an overview of semi-supervised deep generative models and introduces the proposed models. Section 4 explains the data, methodology and main results. Finally, Section 5 presents the main conclusion of this research.

2. Related work

Banks decide whether to grant credit to new applications as well as how to deal with existing customers, e.g. deciding

whether credit limits should be increased and determining which marketing campaign is most appropriate. The tools that help banks with the first problem are called credit scoring models, while behavioral scoring models are used to handle exiting customers [37]. Both type of models estimate the ability that a borrower will be unable to meet its debt obligations, which is referred to as default probability. This research focuses on reject inference to improve the classification accuracy of credit scoring models by utilizing the rejected applications. In Table 1, we present an updated research overview on reject inference in credit scoring extending the one presented in [30].

There are two broad approaches to estimate the default probability; the function estimation model (e.g. logistic regression) and the density estimation approach (e.g. linear discriminant analysis). The latter is more susceptible to provide biased parameter estimates when the rejected applications are ignored [6,17].

According to [6], reject inference represents several challenges. First of all, when attempting to correct the selection bias, the customer characteristics used to develop the current credit scoring model must be available. Otherwise, including the rejected applications in the new model might be insufficient to correct the selection bias. Some techniques, such as mixture decomposition, require assumptions about the default and non-default distributions. In general, these distributions are unknown. Finally, the methods based on supplementary credit information about the reject applications, which might be bought at credit bureaus, can be unrealistic for some financial institutions. Either they cannot afford to pay for it or the data may not be available.

A simple approach for reject inference is augmentation [32]. In this approach, the accepted applications are re-weighted to represent the entire population. The common way to find these weights is using the accept/reject probability. For example if a given application has a probability of being rejected of 0.80, then all similar applications would be weighted up $1/(1 - 0.8) = 5$ times [1]. None of the empirical research using augmentation shows significant improvements in either correcting the selection bias or improving model performance, see [1,2,18–21,23]. The augmentation technique assumes that the default probability is independent of whether the loan is accepted or rejected [38]. However, [24] shows empirically that this assumption is wrong.

¹ For a review of the reclassification and fuzzy parceling approaches see [1,3].

Heckman's bivariate two-stage model [39,40] has been used in different reject inference studies.² This approach simultaneously models the accept/reject and default/non-default mechanisms. Assuming that the error terms in these processes are bivariate normally distributed with unit variance and correlation coefficient ρ , the selection bias arises when $\rho \neq 0$ and it is corrected using the inverse of the Mills ratio.

Despite the popularity of Heckman's model, it is unclear whether this model can correct the selection bias or improve model performance. Some studies claim either higher model performance or different model parameters after using Heckman's model [5,18,23,24,42]. These results, as explained by [4], depend upon whether the selection and default equations are correlated. On the other hand, [25,28,43] state that the model parameters are inefficient, and the main criticism is that the Heckman's model fails to correct the selection bias when it is strong. This happens either when the correlation between the error terms in the selection and outcome equations is high or the data has high degree of censoring [43].

A comparison of different reject inference methods, e.g. augmentation, parceling, fuzzy parceling and the Heckman's model, is presented in [3]. The parceling and fuzzy parceling methods are very similar. They first fit a logistic regression model using the accepted applications. Then they use this model to estimate the default probability for all rejected applications. The difference is that the parceling method chooses a threshold on the default probability to assign the unknown outcome y to the rejected applications. On the other hand, the fuzzy parceling method assumes that each reject application has both outcomes $y = 1$ and $y = 0$, with weights given by the fitted model using only the accepted applications. Finally, the parceling (fuzzy parceling) method fits a new (weighted) logistic regression using both accepted and rejected applications. The results in [3] do not show higher model performance using the reject inference methods. However, the parameter estimates are different when applying the augmentation and parceling approaches. Hence, reject inference has a statistical and economic impact on the final model in this case.

Support vector machines are used in [27] to extend the self-training (SL) algorithm, by adding the hypothesis that the rejected applications are riskier.³ Specifically, their approach iteratively adds rejected applications with higher confidence, i.e. vectors far from the decision-hyperplane, to retrain a SVM (just as in the SL algorithm). However, vectors close to the hyperplane are penalized since the uncertainty about their true label is higher. Their proposed iterative approach shows superior performance compared to other reject inference configurations using SVMs, including semi-supervised support vector machines (S3VM). In addition to higher performance, the iterative procedure in [27] is faster than the S3VM.

The S3VM model is used in [30] for reject inference in credit scoring⁴ using the accepted and rejected applications to fit an optimal hyperplane with maximum margin. The hyperplane traverses through non-density regions of rejected applications and,

at the same time, separates the accepted applications. Their results show higher performance compared to the logit and supervised support vector machine models. In Section 4, we show that S3VM does not scale to large credit scoring data sets and that our proposed models are able to use, at least, 16 times more data compared to S3VM.

In [17] Gaussian mixture models (GMM) are used for density estimation of the default probability. The idea is that each component in the mixture density models a class-conditional distribution. Then, the model parameters are estimated using the expectation-maximization (EM) algorithm, which can estimate the parameters even when the class labels for the rejected applications are missing. The EM algorithm is also used for reject inference in [29]. Both papers report high model performance. However, the results in [17] are based on artificial data and [29] only judge performance based on the Confusion matrix. Finally, the major limitation of the EM algorithm is that we need to be able to estimate the expectation over the latent variables. We show in Section 3 that deep generative models circumvent this restriction by approximation.

A Bayesian approach for reject inference is presented in [28]. In this method the default probability is inferred from the missing data mechanism. The authors use the bound-collapse approach⁵ to estimate the posterior distribution over the score and class label, which is assumed to have a Dirichlet distribution as well as the marginal distribution of the missing class label. The reason for using the bound-collapse method is to avoid exhaustive numerical procedures, like the Gibbs Sampling, to estimate the posterior distributions in this model. Their results show that the Bayesian bound-collapse method perform better than the augmentation and Heckman's model.

In this research we propose a novel Bayesian inference approach for reject inference in credit scoring, which uses Gaussian mixture models and differs from [17,28] in that our models are based on variational inference, neural networks, and stochastic gradient optimization. The main advantages of our proposed method are that (i) inference of the rejected applications is based on an approximation of the posterior distribution and on the exact enumeration of the two possible outcomes that the rejected applications could have taken, (ii) the models use a latent representation of the customers' data, which contain powerful information, and (iii) deep generative models scale to large data sets.

3. Deep generative models

The principles of variational inference with deep neural networks are given in [45,46]. Building upon this work, [7] proposed a generalized probabilistic approach for semi-supervised learning. This approach will be explained in Section 3.1 before we introduce two novel models for reject inference in credit scoring in Sections 3.2 and 3.3.

3.1. Semi-supervised deep generative models for reject inference

In reject inference, the data set $D = \{D_{\text{accept}}, D_{\text{reject}}\}$ is composed of n (labeled) accepted applications $D_{\text{accept}} = \{(\mathbf{x}, y)_1, \dots, (\mathbf{x}, y)_n\}$ and m (unlabeled) rejected applications $D_{\text{reject}} = \{\mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+m}\}$, where $\mathbf{x} \in \mathbb{R}^{\ell_x}$ is the feature vector and $y_i \in \{0, 1\}$ is the class label or the outcome of the loan, $y = 0$ if the customer repaid the loan, otherwise $y = 1$. Additionally, generative models assume that latent variable $\mathbf{z} \in \mathbb{R}^{\ell_z}$ governs the distribution of \mathbf{x} .

² The Heckman's model, named after Nobel Laureate James Joseph Heckman, has been extended or modified in different directions. See [4] for a chronological overview of the model evolution and its early applications. It was in [41] where the Heckman's approach was first applied to credit scoring where the outcome is discrete.

³ The self-training algorithm is an iterative approach where highly confident predictions about the unlabeled data are added to retrain the model. This procedure is repeated as many times as the user specify it. The main criticism of this method is that it can strengthen poor predictions [7].

⁴ The model used in [30], originally developed by [44], uses a branch-and-bound approach to solve the mixed integer constrained quadratic programming problem faced in semi-supervised SVMs. This approach reduces the training time making it suitable for large-sized problems.

⁵ This model is originally presented in Sebastiani and Ramoni (2000) "Bayesian inference with missing data using bound and collapse".

The goal of the generative model is to obtain the joint distribution $p(\mathbf{x}, y)$ of the data used for credit scoring and the outcome of the loan. However, this distribution is intractable since it requires integration over the whole latent space, i.e. $\int p(\mathbf{x}, y, \mathbf{z})d\mathbf{z}$. Further, the intractability of $p(\mathbf{x}, y)$ translates into an intractable posterior distribution of \mathbf{z} through the relationship

$$p(\mathbf{z}|\mathbf{x}, y) = \frac{p(\mathbf{x}, y, \mathbf{z})}{\int p(\mathbf{x}, y, \mathbf{z})d\mathbf{z}}. \quad (1)$$

Hence, we approximate the true posterior $p(\mathbf{z}|\mathbf{x}, y)$ with the inference model $q(\mathbf{z}|\mathbf{x}, y)$ and minimize the Kullback–Leibler (KL) divergence⁶ $KL[q(\mathbf{z}|\mathbf{x}, y)||p(\mathbf{z}|\mathbf{x}, y)]$ to make the approximation as close as possible to the true density.

The $KL[q(\mathbf{z}|\mathbf{x}, y)||p(\mathbf{z}|\mathbf{x}, y)]$ term, the objective function \mathcal{L}_{accept} , and the density $p(\mathbf{x}, y)$ are related by the following expression

$$\begin{aligned} \log p(\mathbf{x}, y) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, y)}[\log p(\mathbf{x}, y)] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, y)}\left[\log \frac{p(\mathbf{x}, y, \mathbf{z})}{p(\mathbf{z}|\mathbf{x}, y)} \frac{q(\mathbf{z}|\mathbf{x}, y)}{q(\mathbf{z}|\mathbf{x}, y)}\right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, y)}\left[\log \frac{p(\mathbf{x}, y, \mathbf{z})}{q(\mathbf{z}|\mathbf{x}, y)}\right] + \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, y)}\left[\log \frac{q(\mathbf{z}|\mathbf{x}, y)}{p(\mathbf{z}|\mathbf{x}, y)}\right] \\ &:= -\mathcal{L}_{accept}(\mathbf{x}, y) + KL[q(\mathbf{z}|\mathbf{x}, y)||p(\mathbf{z}|\mathbf{x}, y)]. \end{aligned} \quad (2)$$

Given that the KL divergence in Eq. (2) is strictly positive, the term $-\mathcal{L}_{accept}(\mathbf{x}, y)$ is a lower bound on $\log p(\mathbf{x}, y)$, i.e. $\log p(\mathbf{x}, y) \geq -\mathcal{L}_{accept}(\mathbf{x}, y)$. Hence, since we cannot evaluate $p(\mathbf{z}|\mathbf{x}, y)$, we maximize $\log p(\mathbf{x}, y)$ by maximizing the negative lower bound.

Note that in Eq. (2) we assume that the outcome y of the loan is known. However, this is not the case for the rejected applications D_{reject} . In this case, generative models treat y as a latent variable and approximate the true posterior distribution $p(y|\mathbf{x})$ with the parametric function $q(y|\mathbf{x})$. Assuming the factorization $q(\mathbf{z}, y|\mathbf{x}) = q(y|\mathbf{x})q(\mathbf{z}|\mathbf{x}, y)$ and a simple form for $q(y|\mathbf{x})$, we can take the explicit expectation over the class label y , i.e. we handle the uncertainty about the outcome of the loan by summing over the two possible outcomes that it might have taken. Mathematically,

$$\begin{aligned} \mathbb{E}_{q(\mathbf{z}, y|\mathbf{x})}\left[\log \frac{p(\mathbf{x}, y, \mathbf{z})}{q(\mathbf{z}, y|\mathbf{x})}\right] &= \mathbb{E}_{q(y|\mathbf{x})}\mathbb{E}_{q(\mathbf{z}|\mathbf{x}, y)}\left[\log \frac{p(\mathbf{x}, y, \mathbf{z})}{q(\mathbf{z}, y|\mathbf{x})}\right] \\ &= \mathbb{E}_{q(y|\mathbf{x})}[-\mathcal{L}_{accept}(\mathbf{x}, y) - \log q(y|\mathbf{x})] \\ &= \sum_y q(y|\mathbf{x})[-\mathcal{L}_{accept}(\mathbf{x}, y) - \log q(y|\mathbf{x})] \\ &:= -\mathcal{L}_{reject}(\mathbf{x}). \end{aligned} \quad (3)$$

Therefore, the objective function in semi-supervised deep generative models is the sum of the supervised lower bound for the accepted applications and the unsupervised lower bound for the rejected applications

$$\mathcal{L} = \mathcal{L}_{accept}(\mathbf{x}, y) + \mathcal{L}_{reject}(\mathbf{x}). \quad (4)$$

Furthermore, deep generative models parametrize the parameters of the density functions in Eqs. (2) and (3) by multilayer perceptron (MLP) networks. For example, if $\mathbf{z}|\mathbf{x}, y$ is multivariate Gaussian distributed with diagonal covariance matrix, we use the notation

$$p(\mathbf{z}|\mathbf{x}, y) \sim \mathcal{N}(\mathbf{z}|\mathbf{x}, y; \boldsymbol{\mu} = f_{\theta}(\mathbf{x}, y), \boldsymbol{\sigma}^2 \mathbf{I} = f_{\sigma}(\mathbf{x}, y)), \quad (5)$$

where $\boldsymbol{\mu} \in \mathbb{R}^{\ell_z}$ and $\boldsymbol{\sigma}^2 \in \mathbb{R}^{\ell_z}$, to specify that the parameters of the Gaussian distribution are parameterized by an MLP network

⁶ The KL divergence is a measure of the proximity between two densities, e.g. $KL[q(\cdot)||p(\cdot)]$, and it is commonly measured in bits. It is non-negative and it is minimized when $q(\cdot) = p(\cdot)$.

denoted by $f(\mathbf{x}, y)$ with input data \mathbf{x}, y and weights $\boldsymbol{\theta}$.⁷ Hence, the optimization of the objective function is with respect to the weights in the MLP. An alternative notation is to simply use the subscript $\boldsymbol{\theta}$ in the corresponding distribution, i.e. $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}, y)$.

Finally, note that the EM algorithm used in [17,29] cannot be used in this context since it requires to compute the expectation of $p(\mathbf{z}|\mathbf{x}, y)$, which it is intractable. Other variational inference techniques, like mean-field or stochastic variational inference, determine different values of $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i^2$ for each data point \mathbf{x}_i , which is computationally expensive. Similarly, traditional EM algorithms need to compute an expectation w.r.t the whole data set before updating the parameters. Therefore, deep generative models use complex functions of the data \mathbf{x} (MLP networks) to estimate the best possible values for the latent variables \mathbf{z} . This allows replacing the optimization of point-specific parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i^2$, with a more efficient optimization of the MLP weights $\boldsymbol{\theta}$. The latter is denoted amortized inference [48].

3.2. Model 1: generative and inference process

In this section we build upon the work done in [7,11] to develop a new semi-supervised model with a Gaussian mixture parameterized with MLPs. The Gaussian mixture induces a flexible latent space that improves the approximation of the lower bound [8,31]. Hence, Model 1 assumes a generative process $p_{\boldsymbol{\theta}}(\mathbf{x}, y, \mathbf{z}) = p(y)p_{\boldsymbol{\theta}}(\mathbf{z}|y)p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, where $\mathbf{x} \perp y|\mathbf{z}$, with the following probability density functions

$$\begin{aligned} p(y) &\sim \text{Bernoulli}(y; \pi), \\ p(\mathbf{z}|y) &\sim \mathcal{N}(\mathbf{z}|y = k; \boldsymbol{\mu}_{z_k} = f_{\theta}(y), \boldsymbol{\sigma}_{z_k}^2 \mathbf{I} = f_{\sigma}(y)) \text{ for } k = 0, 1, \\ p(\mathbf{x}|\mathbf{z}) &\sim \mathcal{N}(\mathbf{x}|\mathbf{z}; \boldsymbol{\mu}_{\mathbf{x}} = f_{\theta}(\mathbf{z}), \boldsymbol{\sigma}_{\mathbf{x}}^2 \mathbf{I} = f_{\sigma}(\mathbf{z})). \end{aligned} \quad (6)$$

Here \mathcal{N} denotes the Gaussian distributions and $f(\cdot)$ is a multilayer perceptron model with weights denoted by $\boldsymbol{\theta}$. Furthermore, we assume that the inference process is factorized as $q(\mathbf{z}, y|\mathbf{x}) = q(y|\mathbf{x})q(\mathbf{z}|\mathbf{x}, y)$, with the following probability densities

$$\begin{aligned} q(y|\mathbf{x}) &\sim \text{Bernoulli}(y; \pi_{y|\mathbf{x}} = f_{\phi}(\mathbf{x})), \\ q(\mathbf{z}|\mathbf{x}, y) &\sim \mathcal{N}(\mathbf{z}|\mathbf{x}, y; \boldsymbol{\mu}_{\mathbf{z}} = f_{\phi}(\mathbf{x}, y), \boldsymbol{\sigma}_{\mathbf{z}}^2 \mathbf{I} = f_{\phi}(\mathbf{x}, y)). \end{aligned} \quad (7)$$

Again \mathcal{N} is the Gaussian distribution and $f(\cdot)$ is a multilayer perceptron model with weights denoted by $\boldsymbol{\phi}$. Note that the marginal distribution $p(\mathbf{z})$ in the generative process is a GMM, i.e.

$$\begin{aligned} p(\mathbf{z}) &= \sum_y p(y)p(\mathbf{z}|y) \\ &= \pi \mathcal{N}(\boldsymbol{\mu}_{z_0}, \boldsymbol{\sigma}_{z_0}^2 \mathbf{I}) + (1 - \pi) \mathcal{N}(\boldsymbol{\mu}_{z_1}, \boldsymbol{\sigma}_{z_1}^2 \mathbf{I}), \end{aligned}$$

where $(1 - \pi)$ represents the prior for the default probability. The generative and inference processes are shown in Fig. 1.

In the following sections, we use $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ to distinguish the expectation and variance terms in the generative process from the ones in the inference process as well as to differentiate the MLP's weights in the generative process from the ones in the inference process. Further, we derive the lower bound for the supervised and unsupervised data under our novel approach for reject inference in credit scoring.

⁷ Deep generative models can also be developed with convolutional neural networks (CNNs). However, CNNs require structured data like videos, images, or time-series data. The data sets in this research are feature vectors with customer's characteristics at the application time. This kind of data does not have the grid-like structure required for training CNNs. For an application of CNNs in credit scoring the reader is referred to [47].

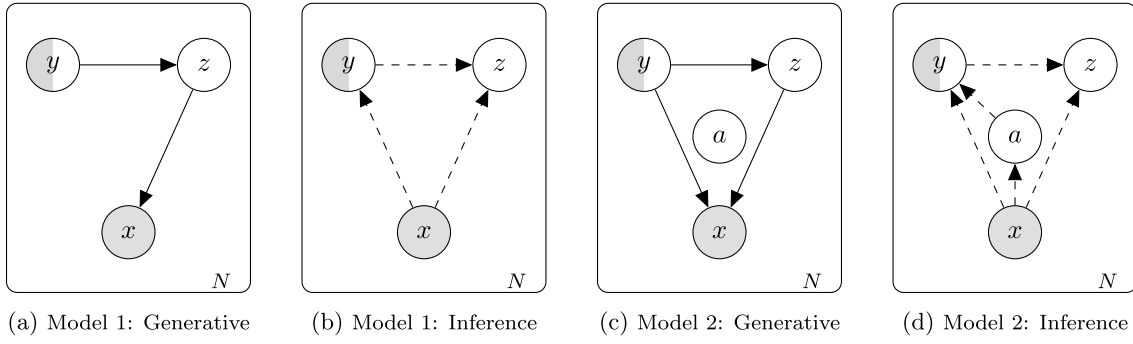


Fig. 1. Plate notation for Model 1 and Model 2 where \mathbf{x} is the observed feature vector, y is the outcome of the loan and it is only observed for the accepted applications, and z and a are latent variables. The generative process is specified by solid lines, while the inference process is shown with dotted lines. Note that the MLP weights θ and ϕ lie outside the plates and we omit them to do not clutter the diagrams.

Labeled data: Deriving the objective function $\mathcal{L}_{\text{accept}}$

We use Eq. (2) and the factorization of the generative process in Eq. (6) to derive the lower bound for the accepted data set D_{accept} . Hence, expanding the terms in the lower bound we obtain

$$\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x},y)} \left[\log \frac{p_{\theta}(\mathbf{x}, y, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x}, y)} \right] = \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x},y)} [\log p(y) + \log p_{\theta}(\mathbf{z}|y) + \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x}, y)], \quad (8)$$

and taking the expectations, see Appendix B.2 in the Appendix, we find the negative lower bound for a single (supervised) data point, which is

$$\begin{aligned} & -\mathcal{L}_{\text{accept}}(\{\mathbf{x}, y\}_i; \theta, \phi) \\ &= \frac{1}{2} \left[\sum_{j=1}^{\ell_z} (1 + \log \sigma_{\phi_j}^2) - \sum_{j=1}^{\ell_z} \left(\log \sigma_{\theta_{j,y}}^2 + \frac{\sigma_{\phi_j}^2}{\sigma_{\theta_{j,y}}^2} + \frac{(\mu_{\phi_j} - \mu_{\theta_{j,y}})^2}{\sigma_{\theta_{j,y}}^2} \right) \right] + \log \pi_i \\ &+ \frac{1}{L} \sum_{l=1}^L \log \mathcal{N}(x_i | z_{i,l}). \end{aligned} \quad (9)$$

Here ℓ_z is the dimension of \mathbf{z} , $\sigma_{\phi_j}^2$ and μ_{ϕ_j} are the j 'th element of σ^2 and μ , respectively, π_i is the prior distribution over the class label y_i , and L is the number of $\mathbf{z}_{i,l}$ samples drawn from $q_{\phi}(\mathbf{z}|\mathbf{x}, y)$. We use the *reparametrization trick* $\mathbf{z}_{i,l} = \mu_{i_{\phi}} + \sigma_{i_{\phi}} \odot \epsilon_l$, where $\epsilon_l \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and \odot denotes an element-wise multiplication, to backpropagate through σ^2 and μ . Hence, the last term in Eq. (9) is $\mathcal{N}(x_i | z_{i,l} = \mu_{i_{\phi}} + \sigma_{i_{\phi}} \odot \epsilon_l)$ and we use $q_{\phi}(\mathbf{z}|\mathbf{x}, y)$ to sample $\mu_{i_{\phi}}$ and $\sigma_{i_{\phi}}$. Note that since y is known in this case, we only need to backpropagate through its corresponding Gaussian component in the MLP parameterizing the GMM. In other words, if $y_i = 0$ the stochastic gradient optimization only updates all weights in μ_{θ_y} and $\sigma_{\theta_y}^2$ for the first component in Fig. 2. This is specified by the subscript y in Eq. (9).

Unlabeled data: Deriving the objective function $\mathcal{L}_{\text{reject}}$

In this case, we treat the unknown labels y as latent variables and we approximate the true posterior distribution with $q(y|\mathbf{x})$. Given that $q(y|\mathbf{x}) \sim \text{Bernoulli}(\cdot)$ is a relatively easy distribution, we take the explicit expectation in the unsupervised lower bound. Following the steps in Eq. (3) together with the factorization in Eqs. (6) and (7), we obtain

$$\begin{aligned} & \mathbb{E}_{q_{\phi}(\mathbf{z}, y|\mathbf{x})} \left[\log \frac{p_{\theta}(\mathbf{x}, y, \mathbf{z})}{q_{\phi}(\mathbf{z}, y|\mathbf{x})} \right] \\ &= \mathbb{E}_{q_{\phi}(\mathbf{z}, y|\mathbf{x})} [\log p(y) + \log p_{\theta}(\mathbf{z}|y) + \log p_{\theta}(\mathbf{x}|\mathbf{z}) - \log q_{\phi}(y|\mathbf{x})] \end{aligned}$$

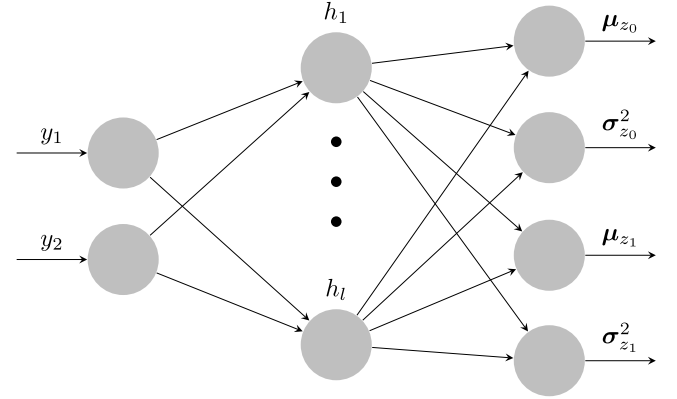


Fig. 2. Gaussian mixture components parameterized by a multilayer perceptron model, where y is the one-hot-encoding for the input data ($[y_1 \ y_2] = [0 \ 1]$ and $[y_1 \ y_2] = [1 \ 0]$ are the one-hot-encoding for $y = 1$ and $y = 0$ respectively), h_l is the l 'th neuron in the hidden layer, and μ_{z_i} and $\sigma_{z_i}^2$ are density moments for the i 'th component in the GMM. For the accepted applications, we backpropagate through its corresponding component, while for the rejected applications we backpropagate through both components.

$$\begin{aligned} & -\log q_{\phi}(\mathbf{z}|\mathbf{x}, y) \\ &= \mathbb{E}_{q_{\phi}(y|\mathbf{x})} [-\mathcal{L}_{\text{accept}}(\mathbf{x}; \theta, \phi) - \log q_{\phi}(y|\mathbf{x})] \\ &= \sum_y q_{\phi}(y|\mathbf{x}) [-\mathcal{L}_{\text{accept}}(\mathbf{x}; \theta, \phi) - \log q_{\phi}(y|\mathbf{x})], \end{aligned} \quad (10)$$

which is, by definition, the unsupervised negative lower bound $-\mathcal{L}_{\text{reject}}(\mathbf{x}; \theta, \phi)$. Furthermore, taking the expectations, see Appendix B.3 in the Appendix, we can obtain the negative lower bound for a single data point, which is

$$\begin{aligned} & -\mathcal{L}_{\text{reject}}(\mathbf{x}_i; \theta, \phi) = \frac{1}{2} \sum_{y=0}^1 \pi_{y|\mathbf{x}_i} \left[\sum_{j=1}^{\ell_z} (1 + \log \sigma_{\phi_j}^2) \right. \\ & \quad \left. - \sum_{j=1}^{\ell_z} \left(\log \sigma_{\theta_{j,y}}^2 + \frac{\sigma_{\phi_j}^2}{\sigma_{\theta_{j,y}}^2} + \frac{(\mu_{\phi_j} - \mu_{\theta_{j,y}})^2}{\sigma_{\theta_{j,y}}^2} \right) \right] \\ & + \sum_{y=0}^1 \pi_{y|\mathbf{x}_i} \log \frac{\pi}{\pi_{y|\mathbf{x}_i}} + \frac{1}{L} \sum_{l=1}^L \log \mathcal{N}(x_i | z_{i,l}), \end{aligned} \quad (11)$$

where $\pi_{y|\mathbf{x}}$ is the y 'th element of the posterior probability over the class labels $\pi_{y|\mathbf{x}} = [\pi_{y=0|\mathbf{x}} \ (1 - \pi_{y=0|\mathbf{x}})]$. The rest of the parameters have the same interpretation as in the supervised negative lower bound. Note that in this case we take the expectation over the latent variable y by enumerating the two possible values ($y = 0$ and $y = 1$) of the posterior parameter $\pi_{y|\mathbf{x}}$, which also implies

that we need to backpropagate through the two components, one at a time, in $\sigma_{\theta_y}^2$ and μ_{θ_y} , see Fig. 2.

We train Model 1 alternating the objective function

$$\mathcal{L} = \sum_i^n \mathcal{L}_{\text{accept}}((\mathbf{x}, y)_i; \boldsymbol{\theta}, \boldsymbol{\phi}) - \alpha \cdot \log \mathbb{E}_{\mathbb{P}_{\hat{p}(\mathbf{x}, y)}}[q_{\phi}(y_i | \mathbf{x}_i)] + \sum_j^{n+m} \mathcal{L}_{\text{reject}}(\mathbf{x}_j; \boldsymbol{\theta}, \boldsymbol{\phi}), \quad (12)$$

where $\mathbb{E}_{\mathbb{P}_{\hat{p}(\mathbf{x}, y)}}$ is the empirical distribution.

Note that we introduce the term $\log \mathbb{E}_{\mathbb{P}_{\hat{p}(\mathbf{x}, y)}}[q_{\phi}(y_i | \mathbf{x}_i)]$, which is actually the classifier in Model 1, into the supervised lower bound to take advantage of the accepted applications and train the best possible classifier. The term $\alpha = \beta \cdot \frac{m+n}{n}$ controls the importance of the classification in the supervised loss function, where m and n are the number of rejected and accepted observations respectively, and β is just a scaling factor.

3.2.1. Reject inference in credit scoring with model 1

Model 1 does not just learn the distribution $p(\mathbf{x}|\mathbf{z})$ of the customers' data used in credit scoring, but it also learns a latent representation $p(\mathbf{z}|\mathbf{x}, y)$ of it. This latent representation reflects an intrinsic structure or the semantics of the customers' data. Additionally, Model 1 approximates the posterior class label distribution $q(y|\mathbf{x})$, which we use to estimate the default probability for new applications. This probability is given by the mutually exclusive outcomes in the posterior parameter $\pi_{y|\mathbf{x}}$, which is parameterized by an MLP with softmax activation function in the output layer.

The most important characteristic of Model 1 for reject inference in credit scoring is that the unknown creditworthiness is evaluated by considering the two possible states $y = 1$ and $y = 0$ that the loan might have taken in case that the credit had been granted (Eq. (10)). This means that this method clearly differs from all extrapolation approaches for reject inference. Further, it is not as restrictive as the expectation-maximization algorithm since it relies on the approximation of the posterior distributions.

It can be shown that Eq. (12) includes the term $KL[q_{\phi}(\mathbf{z}|\mathbf{x}, y) || p_{\theta}(\mathbf{z}|y)]$. Then, the optimization of the objective function forces $q_{\phi}(\mathbf{z}|\mathbf{x}, y)$ to be as close as possible to $p_{\theta}(\mathbf{z}|y)$, which we have modeled as a mixture of Gaussian distributions. The first motivation for this is that the data for the accepted and rejected applications are generated by two different process, just as in [17]. Second, this mixture model generates a flexible latent space, which helps to improve the approximation of the inference process in Model 1.

Finally, the objective function in Eq. (12) includes the MLP weights $\boldsymbol{\theta}$ for the densities $p(\mathbf{z}|y)$ and $p(\mathbf{x}|\mathbf{z})$, and $\boldsymbol{\phi}$ for the densities $q(y|\mathbf{x})$ and $q(\mathbf{z}|\mathbf{x}, y)$. These are all the weights in Model 1 and are present in both the supervised and unsupervised loss. Hence, the stochastic gradient optimization updates these weights jointly and estimates the different parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}^2$, and $\boldsymbol{\pi}$ in Eqs. (6) and (7). In practice, when a labeled (accepted) observation is presented to the algorithm, the loss function in the backpropagation algorithm is $\mathcal{L}_{\text{accept}}((\mathbf{x}, y)_i; \boldsymbol{\theta}, \boldsymbol{\phi})$. Similarly, when handling unlabeled (rejected) observations the loss function is $\mathcal{L}_{\text{reject}}(\mathbf{x}_j; \boldsymbol{\theta}, \boldsymbol{\phi})$. In any case, all the MLP weights $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are updated at each iteration since the same MLP handles both accepted and rejected applications.

3.3. Model 2: generative and inference processes

Inspired by the work by [8,31], we develop an extension of Model 1 introducing auxiliary variables. Auxiliary variables improve the variational approximation and introduce a layer of

latent variables to the model's classifier. Hence, our proposed Model 2 combines a Gaussian mixture with auxiliary variables in a semi-supervised framework for the first time in the literature.

Specifically, we assume the generative process $p(\mathbf{x}, y, \mathbf{z}, \mathbf{a}) = p(\mathbf{a})p(y)p(\mathbf{z}|y)p(\mathbf{x}|\mathbf{z}, y)$ with the following distributions

$$\begin{aligned} p(y) &\sim \text{Bernoulli}(y; \pi), \\ p(\mathbf{a}) &\sim \mathcal{N}(\mathbf{a}; \mathbf{0}, \mathbf{1}), \\ p(\mathbf{z}|y) &\sim \mathcal{N}(\mathbf{z}|y = k; \boldsymbol{\mu}_{z_k} = f_{\theta}(y), \boldsymbol{\sigma}_{z_k}^2 \mathbf{I} = f_{\theta}(y)) \text{ for } k = 0, 1, \\ p(\mathbf{x}|\mathbf{z}, y) &\sim \mathcal{N}(\mathbf{x}|\mathbf{z}, y; \boldsymbol{\mu}_x = f_{\theta}(\mathbf{z}, y), \boldsymbol{\sigma}_x^2 \mathbf{I} = f_{\theta}(\mathbf{z}, y)). \end{aligned} \quad (13)$$

Here \mathcal{N} is the Gaussian distribution and $f(\cdot)$ is a multilayer perceptron model with weights denoted by $\boldsymbol{\theta}$. The inference process factorizes as $q(\mathbf{z}, \mathbf{a}, y|\mathbf{x}) = q(\mathbf{a}|\mathbf{x})q(y|\mathbf{x}, \mathbf{a})q(\mathbf{z}|\mathbf{x}, y)$. The distributions for this process are

$$\begin{aligned} q(\mathbf{a}|\mathbf{x}) &\sim \mathcal{N}(\mathbf{a}|\mathbf{x}; \boldsymbol{\mu}_a = f_{\phi}(\mathbf{x}), \boldsymbol{\sigma}_a^2 \mathbf{I} = f_{\phi}(\mathbf{x})), \\ q(y|\mathbf{x}, \mathbf{a}) &\sim \text{Bernoulli}(y|\mathbf{x}, \mathbf{a}; \pi_{y|\mathbf{x}, \mathbf{a}} = f_{\phi}(\mathbf{x}, \mathbf{a})), \\ q(\mathbf{z}|\mathbf{x}, y) &\sim \mathcal{N}(\mathbf{z}|\mathbf{x}, y; \boldsymbol{\mu}_z = f_{\phi}(\mathbf{x}, y), \boldsymbol{\sigma}_z^2 \mathbf{I} = f_{\phi}(\mathbf{x}, y)). \end{aligned} \quad (14)$$

Again \mathcal{N} is the Gaussian distribution and $f(\cdot)$ is a multilayer perceptron model with weights denoted by $\boldsymbol{\phi}$.

Labeled data: Deriving the objective function $\mathcal{L}_{\text{accept}}$

Following the steps in Section 3.1, it is straightforward to show that the supervised negative lower bound is

$$\begin{aligned} -\mathcal{L}(\mathbf{x}, y; \boldsymbol{\theta}, \boldsymbol{\phi})_{\text{accept}} &= \mathbb{E}_{q_{\phi}(\mathbf{z}, \mathbf{a}|\mathbf{x}, y)} \left[\log \frac{p_{\theta}(\mathbf{x}, y, \mathbf{z}, \mathbf{a})}{q_{\phi}(\mathbf{z}, \mathbf{a}|\mathbf{x}, y)} \right] \\ &= \mathbb{E}_{q_{\phi}(\mathbf{z}, \mathbf{a}|\mathbf{x}, y)} [\log p(\mathbf{a}) + \log p(y) \\ &\quad + \log p_{\theta}(\mathbf{z}|y) + \log p_{\theta}(\mathbf{x}|\mathbf{z}, y) \\ &\quad - \log q_{\phi}(\mathbf{a}|\mathbf{x}) - \log q_{\phi}(\mathbf{z}|\mathbf{x}, y)]. \end{aligned} \quad (15)$$

Using Eqs. (13) and (14) and taking the corresponding expectations, see Appendix B.4 in the Appendix, we obtain the lower bound for the i 'th data point, as follows⁸

$$\begin{aligned} -\mathcal{L}_{\text{accept}}((\mathbf{x}, y)_i; \boldsymbol{\theta}, \boldsymbol{\phi}) &= \frac{1}{2} \left[\sum_{j=1}^{\ell_z} (1 + \log \sigma_{\phi_{z_j}}^2) \right. \\ &\quad \left. - \sum_{j=1}^{\ell_z} \left(\log \sigma_{\theta_{j,y}}^2 + \frac{\sigma_{\phi_{z_j}}^2}{\sigma_{\theta_{j,y}}^2} + \frac{(\mu_{\phi_{z_j}} - \mu_{\theta_{j,y}})^2}{\sigma_{\theta_{j,y}}^2} \right) \right] + \log \pi_i \\ &\quad + \frac{1}{2} \sum_{c=1}^{\ell_a} (\sigma_{\phi_{a_c}}^2 + \mu_{\phi_{a_c}}^2 - (1 + \log \sigma_{\phi_{a_c}}^2)) \\ &\quad + \frac{1}{L_z} \sum_{l=1}^{L_z} \log \mathcal{N}(x_i | z_{i,l}, y). \end{aligned} \quad (16)$$

Here ℓ_z and ℓ_a are the dimensions of \mathbf{z} and \mathbf{a} respectively, σ_j^2 and μ_j are the j 'th element of $\boldsymbol{\sigma}^2$ and $\boldsymbol{\mu}$, respectively, and they refer to the variance or expectation of either \mathbf{z} or \mathbf{a} , π_i is the prior distribution over the class label y_i , and L_z is the number of $\mathbf{z}_{i,l}$ samples drawn from $q_{\phi}(\mathbf{z}|\mathbf{x}, y)$. Note that y is known in this case, hence we only backpropagate through its corresponding Gaussian component, just as in Model 1. This is specified by the subscript y in Eq. (16).

⁸ We clutter the notation by adding the subscript \mathbf{a} and \mathbf{z} in the distribution parameters. This helps to differentiate the parameters of the density $q_{\phi}(\mathbf{a}|\mathbf{x})$ from the ones in $q_{\phi}(\mathbf{z}|\mathbf{x}, y)$.

Unlabeled data: Deriving the objective function \mathcal{L}_{reject}

Using the factorization in Eqs. (13) and (14), the unsupervised negative lower bound in Model 2 has the form

$$\begin{aligned} -\mathcal{L}_{reject}(\mathbf{x}; \boldsymbol{\theta}, \boldsymbol{\phi}) &= \mathbb{E}_{q_{\phi}(\mathbf{z}, \mathbf{a}, \mathbf{y}|\mathbf{x})} \left[\log \frac{p_{\theta}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{a})}{q_{\phi}(\mathbf{z}, \mathbf{a}, \mathbf{y}|\mathbf{x})} \right] \\ &= \mathbb{E}_{q_{\phi}(\mathbf{z}, \mathbf{a}, \mathbf{y}|\mathbf{x})} [\log p(\mathbf{a}) + \log p(\mathbf{y}) + \log p_{\theta}(\mathbf{z}|\mathbf{y}) + \log p_{\theta}(\mathbf{x}|\mathbf{z}, \mathbf{y}) \\ &\quad - \log q_{\phi}(\mathbf{a}|\mathbf{x}) - \log q_{\phi}(\mathbf{z}|\mathbf{x}, \mathbf{y}) - \log q_{\phi}(\mathbf{y}|\mathbf{x}, \mathbf{a})]. \end{aligned} \quad (17)$$

For the i 'th observation, Eq. (17) takes the following form, see Appendix B.5 in the Appendix,

$$\begin{aligned} -\mathcal{L}_{reject}(\mathbf{x}_i; \boldsymbol{\theta}, \boldsymbol{\phi}) &= \frac{1}{2} \frac{1}{L_a} \frac{1}{L_z} \sum_{l_a=1}^{L_a} \sum_{y=0}^1 \pi_{y|\mathbf{x}_i, \mathbf{a}_{i, l_a}} \left[\sum_{j=1}^{\ell_z} (1 + \log \sigma_{\phi_{z_j}}^2) \right. \\ &\quad - \sum_{j=1}^{\ell_z} \left(\log \sigma_{\theta_{j, y}}^2 + \frac{\sigma_{\phi_{z_j}}^2}{\sigma_{\theta_{j, y}}^2} \right. \\ &\quad \left. \left. + \frac{(\mu_{\phi_{z_j}} - \mu_{\theta_{j, y}})^2}{\sigma_{\theta_{j, y}}^2} \right) + \frac{1}{L_z} \sum_{l_z=1}^{L_z} \log \mathcal{N}(\mathbf{x}_i | \mathbf{z}_{i, l_z}, y_{l_a}) \right] \\ &\quad + \frac{1}{2} \sum_{c=1}^{\ell_a} \left(\sigma_{\phi_{a_c}}^2 + \mu_{\phi_{a_c}}^2 \right. \\ &\quad \left. - (1 + \log \sigma_{\phi_{a_c}}^2) \right) + \frac{1}{L_a} \sum_{l_a=1}^{L_a} \sum_{y=0}^1 \pi_{y|\mathbf{x}_i, \mathbf{a}_{i, l_a}} (-\log q(y|\mathbf{x}_i, \mathbf{a}_{i, l_a})) \\ &\quad + \log \pi_i. \end{aligned} \quad (18)$$

Here all parameters are just as in $-\mathcal{L}_{accept}(\mathbf{x}, \mathbf{y}; \boldsymbol{\theta}, \boldsymbol{\phi})$. It is important to note that the posterior probability over the class labels $\pi_{y|\mathbf{x}, \mathbf{a}} = [\pi_{y=0|\mathbf{x}, \mathbf{a}} (1 - \pi_{y=0|\mathbf{x}, \mathbf{a}})]$ depends on the sampled auxiliary variables. We denote this dependency explicitly using the subscript \mathbf{a} .

Finally, just as we did in Model 1, we include the term $\log q_{\phi}(y|\mathbf{x}, \mathbf{a})$ in the unsupervised objective function to take advantage of the accepted applications. Therefore, the final objective function for Model 2 is

$$\begin{aligned} \mathcal{L} &= \sum_i^m \mathcal{L}_{accept}((\mathbf{x}, \mathbf{y})_i; \boldsymbol{\theta}, \boldsymbol{\phi}) - \alpha \cdot \log \mathbb{E}_{\tilde{p}(\mathbf{x}, \mathbf{y}, \mathbf{a})} [q_{\phi}(y_i|\mathbf{x}_i, \mathbf{a}_i)] \\ &\quad + \sum_j^n \mathcal{L}_{reject}(\mathbf{x}_j; \boldsymbol{\theta}, \boldsymbol{\phi}). \end{aligned} \quad (19)$$

3.3.1. Reject inference in credit scoring with model 2

Model 2 has almost the same characteristics as Model 1, but there are two new items. First, Model 2 approximates two layers of latent representations $q(\mathbf{a}|\mathbf{x})$ and $q(\mathbf{z}|\mathbf{x}, \mathbf{y})$. The posterior distribution $q(\mathbf{a}|\mathbf{x})$, together with the customers' data \mathbf{x} , is used to estimate the default probability (Eq. (14)). By doing so, Model 2 has a relatively more expressive estimation of creditworthiness. The presumption is that the latent representation \mathbf{a} captures the intrinsic structure of the data and that it therefore provides relevant features for enhancing the performance of the classifier $q(y|\mathbf{x}, \mathbf{a})$. Finally, note that $q(\mathbf{a}|\mathbf{x})$ is assumed to be multivariate Gaussian distributed, hence we use the reparametrization trick (see Section 3.2) to sample from this distribution, i.e. $\mathbf{a} = \boldsymbol{\mu}_a + \boldsymbol{\sigma}_a \odot \boldsymbol{\epsilon}$ where $\boldsymbol{\mu}_a$ and $\boldsymbol{\sigma}_a$ are the outputs in the MLP for the density $q(\mathbf{a}|\mathbf{x})$.

The second difference from Model 1 is that the data generating process $p(\mathbf{x}|\mathbf{z}, \mathbf{y})$ is conditioned on the latent variable \mathbf{z} and class label y . This is simply done to achieve better training stability. See Section 4.3 for more details about model training.

4. Experiments and results

The goal with the experiments is twofold. First, we compare the performance of our proposed models with a range of techniques representing the state-of-the-art in reject inference for credit scoring, including three classical reject inference techniques (reclassification, fuzzy parceling and augmentation [32]) and three semi-supervised machine learning approaches (self-learning [33] MLP, self-learning SVM, and semi-supervised SVM [34]) under a realistic scenario preserving the original acceptance rates in two real data sets. Second, to have a better understanding of the behavior of reject inference models for credit scoring, we test the model performance in different scenarios varying the number of accepted and rejected observations. In both cases, we include two supervised machine learning models (multilayer perceptron (MLP) [35] and support vector machine (SVM) [36]) to measure the marginal gain of reject inference.

4.1. Data description

We use two real data sets containing both rejected and accepted applications. The first data set is public⁹ and consists of personal loan applications through Lending Club, which is the world's largest peer-to-peer lending company. We replicate the data sample used in [30], which includes applications from January 2009 until September 2012 with 36-months maturity. However, we do not split the data set in yearly sub samples, since we want to keep as many observations from the minority class ($y = 1$) as possible. Hence, the data set that we use in our experiments has 53 698 accepted applications, including 6 528 defaults, and 536 459 rejected applications.¹⁰ That is, the acceptance ratio is 9.10% and default rate is 12.16%. For more details about the Lending Club data, see Table A.1 in Appendix A.

The second data set is provided by Santander Consumer Bank Nordics and consists of credit card applications arriving through their internet website. The applications were received during the period January 2011 until December 2016. During this period Santander accepted 126 520 applications and only 14 993 customers ended up as defaults. The number of rejected applications during this period is 232 898. Hence, the acceptance ratio is 35.20% and default rate 11.85%.

In addition to these two data sets, we have two small samples after September 2012 and December 2016 for Lending Club and Santander Bank respectively, which are used to produce well-calibrated estimates of class probabilities using the beta calibration approach [49]. These samples are not part of the experimental design explained in Section 4.2.

4.2. Experimental design

We conduct two different set of experiments. In the first experimental setup, we keep the original acceptance ratio, but we do not use more than 34 100 observations in total.¹¹ To construct this data set, we first split the original data in 70%–30% for training and testing respectively. Then, we down sample

⁹ The data can be obtain directly at the Lending Club's website, however they require the user to login. We obtain a complete version of the available data at the website https://github.com/nateGeorge/preprocess_lending_club_data, which is updated quarterly.

¹⁰ The number of accepted and rejected applications are not exactly the same as in [30], but the variable statistics are very similar and the default trend is the same. See Table A.1 for more information.

¹¹ This is done to allow a fair comparison to S3VM, which does not scale to larger datasets due to memory requirements. For the 34 100 observations, S3VM requires 123GB of memory to estimate the kernel matrix.

Table 2

Grid for hyperparameter optimization for Model 1 and 2 and for both data sets. The numbers within brackets specify the number of neurons in each hidden layers, i.e. [10 10] means two hidden layers with 10 neurons each. Finally, the superscript * and ** shows the final architecture for Model 1 and Model 2 respectively for the Lending Club data set used in Table 3. Similarly, *** and **** shows the final architecture for Model 1 and Model 2 respectively for the Santander Credit Cards data set used in Table 3.

Lending Club and Santander Credit Cards	
MLP Network	Number of hidden layers and dimensions
$q(\mathbf{z} \mathbf{x}, y)$	[10 10]*, [10 20], [10 30], [10 50], [100 70]***, [10 20 10], [10 30 10], [10 40 10]**, [10 50 10], [60 90 60]****
$p(\mathbf{x} \cdot)$	[10 10]*, [10 20], [10 30], [10 50], [70 100]***, [10 20 10], [10 30 10], [10 40 10]**, [10 50 10], [60 90 60]****
$p(\mathbf{z} y)$	[10]*,***,****,****
$q(\mathbf{a} \mathbf{x})$	[50], [10 10], [10 20], [10 30], [10 40]**, [10 50], [20 40], [20 50], [30 50], [30 60], [40 60]****
$q(y \cdot)$	[50], [60], [70]*, [80]**, [100]****, [120], [130]**
Parameter/hyperparameter	Value
\mathbf{z} dimension	30, 50*.,**.,****., 100***
\mathbf{a} dimension	30, 50*.,****.
β	0.008**, 0.01, 0.025, 0.14, 1.1*, 3****, 8***

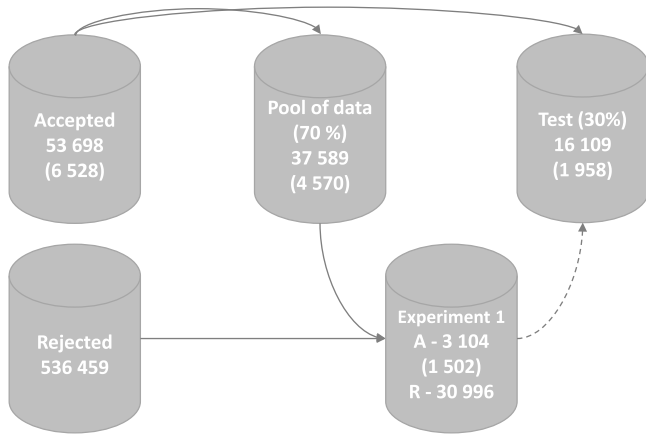


Fig. 3. Data partition used in the experiments in Table 3 for the Lending Club data set. Numbers in parentheses are the number of defaulted observations, and numbers in parenthesis in percentage are the proportion of accepted applications. The experiments with the Santander data set and in Table 4 follows the same logic, but in the last sampling ('Experiment 1' box) we sample the number of accepted and rejected applications as needed.

the majority class ($y = 0$) in the training set until it equals the number of observations for the minority class ($y = 1$). To achieve the correct acceptance ratio, this requires a random selection of both class labels. Note that the test data set is left as it is, i.e. it preserves the original default rate. Finally, we randomly select the number of reject applications in a way that these, together with the balanced training sample, do not exceed 34 100 observations, see Fig. 3.

In the second set of experiments,¹² we analyze the effect of varying the number of accepted (rejected) applications, while keeping the same number of rejected (accepted) applications. We follow the same approach as in the first experiments, splitting the data set into a training and test data set, down sampling the training set, and randomly selecting the number of reject applications.

For the Lending Club data set, we use all variables in Table A.1 to train all models, while for the Santander data we use a forward selection approach to select the explanatory variables that are included in the reclassification, fuzzy parcelling and augmentation

¹² S3VM is not included in this section since it takes around 356 h to evaluate each scenario in this section and in total we evaluate 12 different scenarios. In addition, it has the memory restrictions already mentioned. Similarly, the iterative procedure in the self-learning SVM is not feasible in this section.

methods.¹³ For the other models we use all variables in Table A.2. Finally, we do hyperparameter tuning using grid search with 10-cross validation for the MLP, SVM, S3VM, Model 1, and Model 2. The best architecture for the MLP and SVM is used as the base model in the self-training approaches for MLP and SVM. The details of the grid search are given in Table A.3.

4.3. Model implementation and training

Model 1 and Model 2 are implemented in Theano [50]. We use softplus activation functions in all hidden layers and linear activation functions in all output layers estimating μ and σ^2 . For the output layer in the classifiers $q_\phi(y|\cdot)$ we use softmax activation functions. Further, we use the Adam optimizer [51] with learning rate equal to $1e-4$ and $5e-5$ for training of Model 1 and Model 2 respectively. The rest of parameters in the Adam optimizer are the default values suggested in the original paper. We use $L = 1$ and $L_a = 1$ for both Model 1 and 2 in all experiments. Finally, both data sets are standardized before training and testing, and the class label y is one-hot-encoded. The model architectures used in the experiments in Table 3 are shown in Table 2.

It is important to mention that deep generative models are, in general, difficult to train [52,53]. The training of Model 1 and Model 2 in some cases become unstable, especially for the experiments where we vary the number of accepted and rejected applications. Moreover, it is sensitive to the initial weights. Hence, we use a Variational Autoencoder [45] to pretrain the weights in $q_\phi(\mathbf{z}|\mathbf{x}, y)$ and $p_\theta(\mathbf{x}|\mathbf{z})$ for Model 1. Similarly, we prewarm all weights θ and ϕ in Model 2. In both cases, we initialized the MLP weights as suggested in [54]. We also achieve more stable training in Model 2 by conditioning $p_\theta(\mathbf{x}|\mathbf{z}, y)$ on the class label y .

4.4. Benchmark reject inference

Table 3 compares the performance of Model 1 and Model 2 with other models when using the original acceptance ratio in the data sets. It can be seen that both Model 1 and Model 2 perform better than all supervised and semi-supervised models in terms of AUC, GINI, H measure and precision. Our results support previous findings that the reclassification, fuzzy parcelling and augmentation methods do not improve model performance. The reclassification approach is consistently the worst model. Further, the self-training approaches do not improve the performance of the base models MLP and SVM. Finally, S3VM has significantly worse performance than the base models for the Santander Credit Cards data set.

¹³ These three methods are based on the logistic regression. Hence, the forward selection approach prevents the logistic regression from overfitting and avoids numerical problems on its optimization.

Table 3

Model performance keeping the original acceptance ratios, i.e. 9.10% for Lending Club (LC) and 35.20% for Santander Credit Cards (SCC). The training data set is balanced by down sampling the majority class, and the threshold used to calculate recall and precision is based on the empirical default rate in the test data set. The last two columns show the runtime for one cross-validation and the format is given in mm:ss.cs, where mm, ss, and cs stands for minutes, seconds and centiseconds respectively.

	Lending Club (LC)					Santander Credit Cards (SCC)					Runtime	
	AUC	GINI	H-measure	Recall	Precision	AUC	GINI	H-measure	Recall	Precision	LC	SCC
MLP	0.6273	0.2547	0.0535	0.4454	0.1738	0.7091	0.4183	0.1326	0.7909	0.1772	00:01.28	00:04.53
SVM	0.6284	0.2567	0.0543	0.4632	0.1783	0.7388	0.4777	0.1689	0.7997	0.1895	00:06.59	00:14.42
Reclassification	0.5784	0.1567	0.0227	0.4906	0.1493	0.6415	0.2830	0.0625	0.9989	0.1187	00:05.04	00:01.15
Fuzzy Parceling	0.6198	0.2560	0.0540	0.4598	0.1772	0.6791	0.3582	0.0957	0.8676	0.1541	00:03.82	00:08.45
Augmentation	0.6219	0.2558	0.0541	0.4581	0.1777	0.6761	0.3523	0.0923	0.8735	0.1524	00:13.07	00:15.25
Self-learning MLP	0.5868	0.1737	0.0326	0.4504	0.1570	0.6726	0.3451	0.0877	0.8502	0.1519	00:18.80	00:20.53
Self-learning SVM	0.6206	0.2551	0.0535	0.4957	0.1731	0.7266	0.4532	0.1529	0.8494	0.1725	03:25.89	05:08.36
S3VM	0.6201	0.2402	0.0481	0.0000	NA	0.6520	0.3040	0.0733	1.0000	0.1185	09:17.00	06:20.12
Model 1	0.6294	0.2588	0.0554	0.4540	0.1788	0.7394	0.4788	0.1678	0.8326	0.1848	10:48.19	04:12.16
Model 2	0.6363	0.2755	0.0632	0.4688	0.1825	0.7431	0.4851	0.1764	0.6282	0.2303	12:24.06	05:54.33

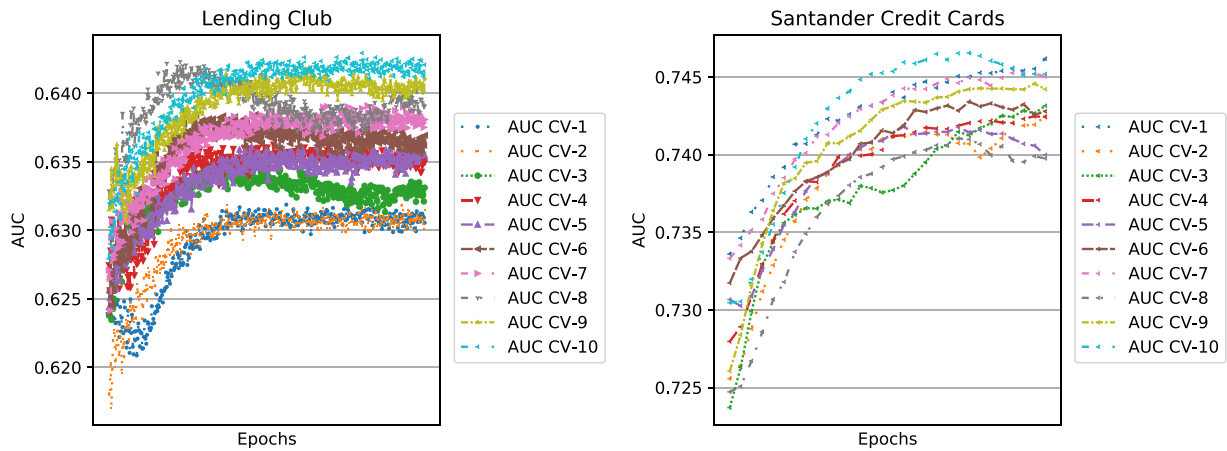


Fig. 4. The left panel shows the AUC performance for the Lending Club data set in the 10 cross-validations (CV), while the right panel shows the performance for the Santander Bank data set. Both diagrams correspond to Model 2.

We use the Platt scaling method [55] to get (pseudo) default probabilities from SVM and S3VM. It is interesting to see that we could not estimate the recall and precision for S3VM in the Lending Club data because the estimated default probabilities are concentrated around the average, with practically no dispersion, see Table A.4. S3VM estimates default probabilities for all applications below the default rate in the Lending Club data set, and above the default rate in the Santander data set.

Model 2 performs better than Model 1 in terms of all measures except for recall, and Figure Fig. 4 shows its AUC performance for both data sets. Remember that the main difference between these models is the classifier in Model 2, which uses a latent representation of the customers' data. Our results are hence in correspondence with previous studies showing the predictive power embedded in the latent transformations. It is further interesting to note that our proposed models for reject inference not only perform better, but also estimate higher variability in the predicted default probabilities, as shown in Table A.4 and Fig. A.1. This result supports previous findings that the default probability is underestimated if reject inference is ignored. Unfortunately, given the nature of the data sets in this research we are not able to draw any conclusion about the economic impact of this interesting detail.

It is worth mentioning that Model 2 is the algorithm that takes longer time to converge for the Lending Club data set, while for the Santander Credit Cards data set is S3VM. In any case, the runtime for both Model 2 and S3VM, in the experiments in Table 3, is moderate.

In Table 4, we analyze the impact of the number of accepted and rejected applications on model performance using Model 2

and the Lending Club data set. In the right panel, we can observe that the general trend is that the more rejected applications we add to Model 2, the higher model performance. In the left panel, we can see that the more accepted data we have available, the better model performance for the supervised models and the less difference compared to Model 2. Figure Fig. A.2 shows the AUC performance for this analysis. Note that Model 2 achieves the highest average AUC of 0.6404 in the *All* scenario, which includes 545 599 observations. This is 16 times more data compared to what self-training SVM and S3VM handled.

The runtime for Model 2 in the experiments that use all rejected applications has increased significantly compared to Table 3. In the scenario where we use all accepted and rejected applications, 545 599 observations in total, Model 2 takes about 4 h to converge. Note that this model has 16 080 learnable parameters, which are significantly more than the 502 parameters in the MLP. Generally, training deep learning architectures is computationally intensive and the computational complexity increases linearly with the number of parameters (including MLP architectures). However, training can be accelerated by distributing training in parallel across multiple GPUs.

5. Conclusion

In this research we develop two novel deep generative models for reject inference in credit scoring. Our models use the posterior distribution of the outcome of the loan to infer the unknown creditworthiness of the rejected applications. This is done by exact enumeration of the two possible outcomes of

Table 4

Left panel: Model performance, measured with AUC, as a function of accepted applications. In all six experiments to the left, we use all 536 459 rejected applications. Right panel: Model performance, measured with AUC, as a function of rejected applications. In all six experiments to the right, we use only 200 accepted applications. Numbers in parenthesis are the acceptance ratio for each experiment. The last two rows show the runtime for one cross-validation and the format is in hh:mm:ss, where hh, mm, and ss stands for hours, minutes, and seconds respectively. We do not include the runtime for the first five models because the difference with respect to the runtimes in Table 3 is negligible.

Lending Club												
No. observations	Accepted applications						Rejected applications					
	200 (0.04%)	600 (0.11%)	1 200 (0.22%)	2 000 (0.37%)	6 000 (1.11%)	All (1.67%)	30 997 (0.64%)	100 000 (0.20%)	200 000 (0.10%)	300 000 (0.07%)	400 000 (0.05%)	All (0.04%)
MLP	0.6002	0.6236	0.6237	0.6304	0.6299	0.6307	0.6037	0.6037	0.6037	0.6037	0.6037	0.6037
SVM	0.6039	0.6267	0.6253	0.6320	0.6302	0.6309	0.6054	0.6054	0.6054	0.6054	0.6054	0.6054
Reclassification	0.5786	0.5785	0.5812	0.5853	0.5806	0.5816	0.5616	0.5785	0.5783	0.5574	0.5693	0.5779
Fuzzy Parceling	0.6017	0.6240	0.6232	0.6295	0.6297	0.6302	0.6041	0.6026	0.6018	0.6031	0.6073	0.6006
Augmentation	0.6017	0.6216	0.6207	0.6301	0.6295	0.6304	0.6023	0.6028	0.6010	0.5967	0.5953	0.5979
Self-learning MLP	0.5824	0.5728	0.5734	0.5675	0.5858	0.5631	0.5640	0.5485	0.5706	0.5715	0.5758	0.5703
Model 2	0.6175	0.6269	0.6310	0.6344	0.6381	0.6404	0.6112	0.6075	0.6091	0.6107	0.6121	0.6175
Runtime												
Self-learning MLP	00:20:36	00:26:14	00:29:31	00:29:23	00:31:39	00:35:11	00:02:10	00:05:02	00:09:50	00:15:01	00:18:02	00:23:36
Model 2	02:39:02	02:41:75	02:55:19	03:24:13	03:42:17	04:03:10	00:14:18	00:38:07	01:09:02	01:39:48	02:00:54	02:39:02

the loan, which is an advantage compared to reject inference methods based on extrapolation. To the best of our knowledge, this is the first research that develops novel methods for reject inference in credit scoring coupling Gaussian mixtures and auxiliary variables in a semi-supervised framework with generative models.

The experiments show that our proposed models achieve higher model performance compared to many of the classical and machine learning approaches for reject inference in credit scoring, and the models' performance increases as we add more data for model training. Further, the efficient stochastic gradient optimization technique used in deep generative models scales to large data sets, which is an advantage over supervised and semi-supervised support vector machines. Note that even though the focus of this research is on credit scoring, our proposed models generalize to other research domains, e.g. image classification.

The higher model performance of our proposed methodology is further enhanced by adding latent representations of the customers' data to the classifier. This data representation captures the intrinsic structure of the data providing relevant information for classification. Since our proposed approach for reject inference in credit scoring offers flexible modeling possibilities, we hope that this research spurs future work on reject inference in credit scoring using deep generative model focusing on improving the training stability and classification power.

Table A.1

Lending club descriptive statistics.

	Variable	Mean	Std	Min	1 Quantile	Median	3 Quantile	Max
Accepts	Debt to income	14.51	7.19	0.00	9.06	14.44	19.82	34.99
	Loan amount	10 610.34	6 738.61	1000.00	5706.25	9 600.00	14 000.00	35 000.00
	Fico score	711.49	35.06	662.00	682.00	707.00	732.00	847.50
	State d1	0.43	0.49	0.00	0.00	0.00	1.00	1.00
	State d2	0.43	0.49	0.00	0.00	0.00	1.00	1.00
	State d3	0.10	0.29	0.00	0.00	0.00	0.00	1.00
	Employment length	3.97	3.18	0.00	1.00	3.00	6.00	10.00
Rejects	Debt to income	24.29	31.14	0.00	7.90	18.19	31.18	419.33
	Loan amount	13 330.74	10 361.51	1000.00	5000.00	10 000.00	20 000.00	35 000.00
	Fico score	638.15	74.10	385.00	595.00	651.00	690.00	850.00
	State d1	0.47	0.50	0.00	0.00	0.00	1.00	1.00
	State d2	0.37	0.48	0.00	0.00	0.00	1.00	1.00
	State d3	0.10	0.30	0.00	0.00	0.00	0.00	1.00
	Employment length	8.40	3.16	0.00	10.00	10.00	10.00	10.00

CRediT authorship contribution statement

Rogelio A. Mancisidor: Methodology, Investigation, Software, Writing - original draft. **Michael Kampffmeyer:** Methodology, Supervision, Validation, Writing - review & editing. **Kjersti Aas:** Methodology, Supervision, Validation, Writing - review & editing. **Robert Jenssen:** Methodology, Supervision, Validation, Writing - review & editing.

Acknowledgments

The authors would like to thank Santander Consumer Bank (Norway) for financial support and the real data set used in this research. This work was also supported by the Research Council of Norway [grant number 260205] and SkatteFUNN, Norway [grant number 276428].

Appendix A. Tables and figures

To replicate the data set presented in [30], we excluded all observations with missing values in any of the variables in Table A.1. Further, the allowed variable range, which we choose based on [30], is determined by the minimum and maximum values as shown in the table. The summary statistics in our data sample is not exactly the same as in [30], but the default trend is

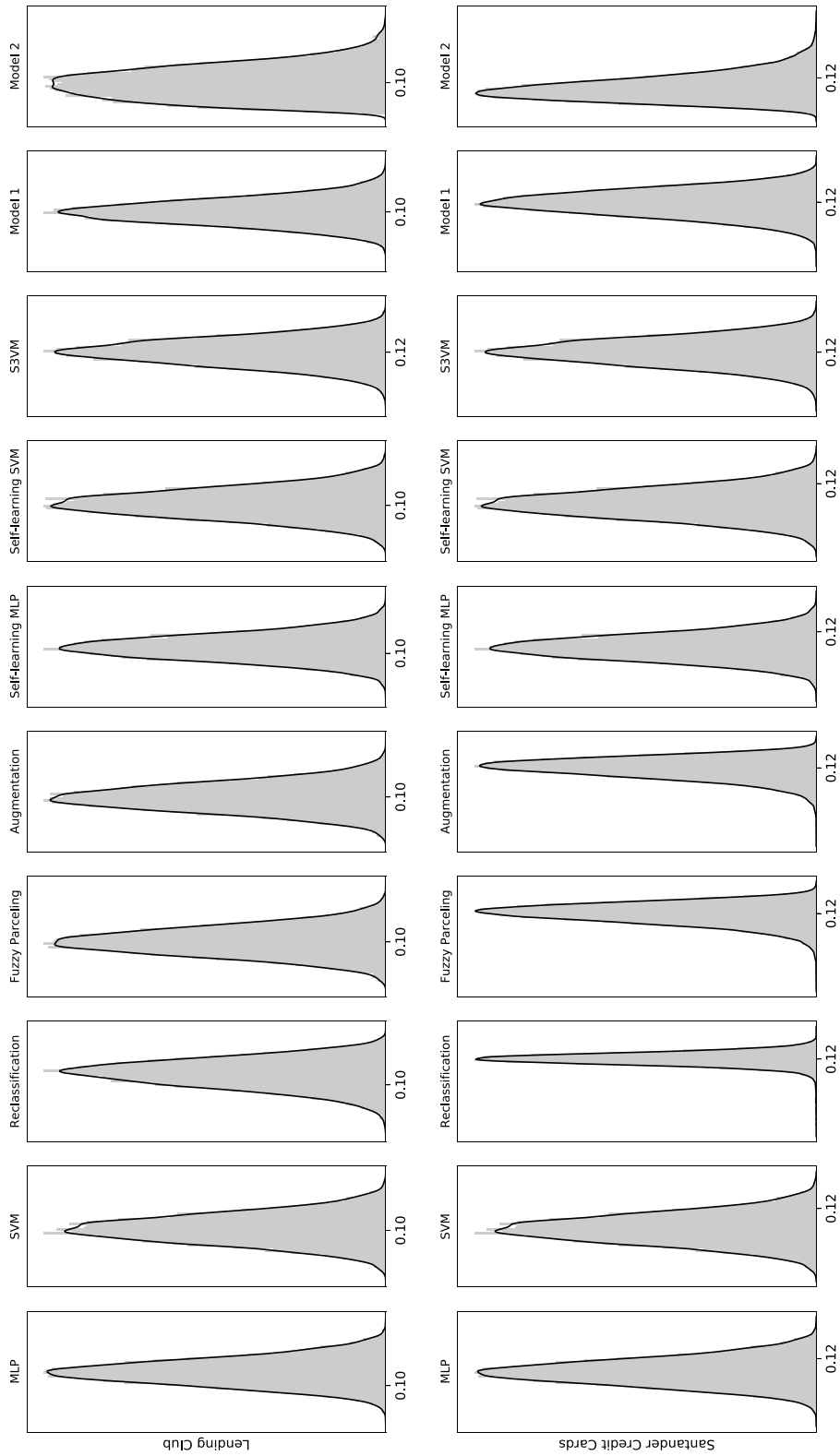


Fig. A.1. Empirical distribution of the default probability for the original acceptance ratio as explained in Section 4.2.

Table A.2
Santander credit cards descriptive statistics.

	Variable	Mean	Std	Min	1 Quantile	Median	3 Quantile	Max
Accepts	Var1	86 475.84	107 975.22	0.00	29 852.00	69 162.00	108 898.00	10 570 323.00
	Var2	152 205.11	1 778 838.75	0.00	0.00	0.00	4 376.00	393 676 928.00
	Var3	38.95	13.38	19.00	28.00	37.00	48.00	92.00
	Var4	976 647.69	16 125 692.00	-2.00	-2.00	-2.00	1 250 000.00	2 701 061 888.00
	Var5	903 518.75	3 228 558.75	-2.00	-2.00	-2.00	1 430 000.00	985 694 976.00
	Var6	807 869.63	13 848 935.00	0.00	0.00	0.00	1 075 000.00	2 667 096 064.00
	Var7	95 622.16	14 090 133.00	-2 664 925 952.00	-2.00	-2.00	79 000.00	984 075 008.00
	Var8	9.46	23.82	-2.00	-2.00	-2.00	4.63	100.00
	Var9	-0.44	1.86	-2.00	-2.00	-2.00	1.00	82.00
	Var10	-0.91	1.14	-2.00	-2.00	-2.00	0.00	4.00
	Var11	-1.99	0.15	-2.00	-2.00	-2.00	-2.00	3.00
	Var12	-0.63	2.06	-2.00	-2.00	-2.00	1.00	164.00
	Var13	-0.34	2.09	-2.00	-2.00	-2.00	1.00	164.00
	Var14	-1.98	0.32	-2.00	-2.00	-2.00	-2.00	26.00
	Var15	-0.47	1.73	-2.00	-2.00	-2.00	1.00	52.00
	Var16	-1.15	1.00	-2.00	-2.00	-2.00	0.00	1.00
	Var17	0.16	0.53	0.00	0.00	0.00	0.00	19.00
	Var18	0.95	2.25	0.00	0.00	0.00	1.00	67.00
	Var19	1.12	2.42	0.00	0.00	0.00	1.00	72.00
	Var20	1.57	3.27	0.00	0.00	0.00	2.00	97.00
	Var21	357 123.84	372 109.81	0.00	170 103.14	295 917.44	443 333.95	34 850 852.00
	Var22	8.29	8.53	0.00	3.97	6.91	10.29	760.94
	Var23	37 156.38	250 887.75	-12 873 071.00	-14 218.19	23 241.04	79 463.82	33 829 372.00
	Var24	16 168.70	432 254.88	-40 114 780.00	0.00	0.00	0.00	50 003 248.00
	Var25	9 037.99	60 101.17	-2 641 216.00	-4 085.00	5 520.00	19 799.25	6 169 685.00
	Var26	0.35	42.04	0.00	0.20	0.23	0.26	14 940.20
	Var27	0.47	0.50	0.00	0.00	0.00	1.00	1.00
	Var28	46.04	75.70	-29.00	-2.00	12.00	65.00	754.00
	Var29	6.71	34.72	-2.00	-2.00	-2.00	-2.00	412.00
	Var30	6.71	34.72	-2.00	-2.00	-2.00	-2.00	412.00
	Var31	1.08	0.97	0.00	0.53	0.90	1.36	43.75
	Var32	0.98	1.02	0.00	0.47	0.82	1.22	101.95
	Var33	0.98	1.01	0.00	0.47	0.81	1.22	99.13
	Var34	0.56	1.18	0.00	0.00	0.00	1.00	73.00
	Var35	0.49	0.50	0.00	0.00	0.00	1.00	1.00
	Var36	0.00	0.01	0.00	0.00	0.00	0.00	1.00
	Var37	0.58	0.49	0.00	0.00	1.00	1.00	1.00
	Var38	0.07	0.25	0.00	0.00	0.00	0.00	1.00
	Var39	0.21	0.41	0.00	0.00	0.00	0.00	1.00
	Var40	0.09	0.29	0.00	0.00	0.00	0.00	1.00
	Var41	0.06	0.23	0.00	0.00	0.00	0.00	1.00
	Var42	0.01	0.11	0.00	0.00	0.00	0.00	1.00
	Var43	0.37	0.48	0.00	0.00	0.00	1.00	1.00
	Var44	0.53	0.50	0.00	0.00	1.00	1.00	1.00
	Var45	0.09	0.28	0.00	0.00	0.00	0.00	1.00
	Var46	0.00	0.02	0.00	0.00	0.00	0.00	1.00
	Var47	0.65	0.48	0.00	0.00	1.00	1.00	1.00
	Var48	0.26	0.44	0.00	0.00	0.00	1.00	1.00
	Var49	0.06	0.23	0.00	0.00	0.00	0.00	1.00
	Var50	0.00	0.00	0.00	0.00	0.00	0.00	1.00
	Var51	0.03	0.18	0.00	0.00	0.00	0.00	1.00
	Var52	0.75	0.44	0.00	0.00	1.00	1.00	1.00
	Var53	0.25	0.44	0.00	0.00	0.00	1.00	1.00
	Var54	0.08	0.27	0.00	0.00	0.00	0.00	1.00
	Var55	0.16	0.36	0.00	0.00	0.00	0.00	1.00
	Var56	0.39	0.49	0.00	0.00	0.00	1.00	1.00
	Var57	0.30	0.46	0.00	0.00	0.00	1.00	1.00
	Var58	0.08	0.27	0.00	0.00	0.00	0.00	1.00
Rejects	Var1	57 198.23	68 931.46	0.00	12 800.00	43 182.50	80 412.00	3 635 832.00
	Var2	33 128.01	568 171.50	0.00	0.00	0.00	0.00	208 626 176.00
	Var3	34.60	12.16	1.00	25.00	32.00	42.00	95.00
	Var4	507 337.69	11 648 304.00	-2.00	-2.00	-2.00	105 937.50	2 701 061 888.00
	Var5	434 133.63	1 137 152.75	-2.00	-2.00	-2.00	0.00	72 376 000.00
	Var6	432 619.66	10 198 556.00	0.00	0.00	0.00	0.00	2 303 705 088.00
	Var7	1 499.88	10 159 168.00	-2 299 855 104.00	-2.00	-2.00	-2.00	72 376 000.00
	Var8	3.45	16.70	-2.00	-2.00	-2.00	0.00	100.00
	Var9	-1.16	1.51	-2.00	-2.00	-2.00	0.00	82.00
	Var10	-1.39	1.02	-2.00	-2.00	-2.00	0.00	4.00
	Var11	-1.87	0.95	-2.00	-2.00	-2.00	-2.00	36.00
	Var12	-1.24	1.67	-2.00	-2.00	-2.00	-2.00	105.00
	Var13	-1.06	1.77	-2.00	-2.00	-2.00	1.00	105.00
	Var14	-1.79	1.20	-2.00	-2.00	-2.00	-2.00	38.00
	Var15	-1.13	1.52	-2.00	-2.00	-2.00	1.00	43.00

(continued on next page)

Table A.2 (continued).

Variable	Mean	Std	Min	1 Quantile	Median	3 Quantile	Max
Var16	-1.52	0.87	-2.00	-2.00	-2.00	-2.00	1.00
Var17	0.26	0.74	0.00	0.00	0.00	0.00	87.00
Var18	3.28	6.06	0.00	0.00	1.00	4.00	166.00
Var19	3.54	6.30	0.00	0.00	1.00	4.00	172.00
Var20	4.62	7.90	0.00	0.00	2.00	5.00	176.00
Var21	250 519.14	242 146.78	0.00	112 918.59	212 571.75	337 357.29	13 897 584.00
Var22	5.80	5.55	0.00	2.64	4.94	7.84	308.84
Var23	23 313.24	179 360.19	-31 086 966.00	-15 761.49	16 862.45	61 574.02	11 590 733.00
Var24	2 551.04	171 498.02	-30 644 804.00	0.00	0.00	0.00	16 552 538.00
Var25	5 758.38	43 678.19	-6 499 649.00	-3 843.00	3 537.00	14 794.00	1 851 795.00
Var26	0.30	31.14	0.00	0.16	0.23	0.26	14 940.20
Var27	0.25	0.43	0.00	0.00	0.00	1.00	1.00
Var28	32.24	65.42	-43.00	-2.00	-2.00	43.00	804.00
Var29	6.67	32.32	-2.00	-2.00	-2.00	-2.00	377.00
Var30	6.67	32.32	-2.00	-2.00	-2.00	-2.00	377.00
Var31	0.77	0.70	0.00	0.35	0.67	1.05	36.99
Var32	0.69	0.67	0.00	0.31	0.59	0.93	38.16
Var33	0.69	0.66	0.00	0.31	0.59	0.93	38.90
Var34	0.36	1.07	0.00	0.00	0.00	0.00	97.00
Var35	0.27	0.45	0.00	0.00	0.00	1.00	1.00
Var36	0.00	0.01	0.00	0.00	0.00	0.00	1.00
Var37	0.51	0.50	0.00	0.00	1.00	1.00	1.00
Var38	0.07	0.26	0.00	0.00	0.00	0.00	1.00
Var39	0.23	0.42	0.00	0.00	0.00	0.00	1.00
Var40	0.14	0.34	0.00	0.00	0.00	0.00	1.00
Var41	0.05	0.22	0.00	0.00	0.00	0.00	1.00
Var42	0.00	0.07	0.00	0.00	0.00	0.00	1.00
Var43	0.20	0.40	0.00	0.00	0.00	0.00	1.00
Var44	0.75	0.43	0.00	0.00	1.00	1.00	1.00
Var45	0.05	0.22	0.00	0.00	0.00	0.00	1.00
Var46	0.00	0.02	0.00	0.00	0.00	0.00	1.00
Var47	0.74	0.44	0.00	0.00	1.00	1.00	1.00
Var48	0.16	0.37	0.00	0.00	0.00	0.00	1.00
Var49	0.06	0.23	0.00	0.00	0.00	0.00	1.00
Var50	0.06	0.00	0.00	0.00	0.00	0.00	1.00
Var51	0.04	0.19	0.00	0.00	0.00	0.00	1.00
Var52	0.55	0.50	0.00	0.00	1.00	1.00	1.00
Var53	0.45	0.50	0.00	0.00	0.00	1.00	1.00
Var54	0.09	0.29	0.00	0.00	0.00	0.00	1.00
Var55	0.16	0.37	0.00	0.00	0.00	0.00	1.00
Var56	0.38	0.48	0.00	0.00	0.00	1.00	1.00
Var57	0.28	0.45	0.00	0.00	0.00	1.00	1.00
Var58	0.09	0.28	0.00	0.00	0.00	0.00	1.00

Table A.3

Grid for hyperparameter optimization for Lending Club: The total number of model configurations are 132, 160 and 240 for MLP, SVM, and S3VM respectively. For the Santander data set the number of model configurations evaluated are 204, 160, and 240 for MLP, SVM, and S3VM respectively.

Lending Club							
MLP		SVM			S3VM		
Layers	1	C	5, 10, 13, 14, 15, 17, 19, 21, 23, 25			C	1, 5, 10, 13, 15, 17
Neurons	3, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60	Gamma	2, 1.5, 1, 0.5, 0.1, 0.01, 0.001, auto			Gamma	2.5, 2, 1.5, 1, 0.5
Activation	logistic, tanh, relu	Kernel	rbf, linear			Kernel	rbf, linear
Learning rate	constant, adaptive					LamU	0.5, 1, 1.5, 2
Solver	sgd, adam						
Santander Credit Cards							
Layers	1	C	5, 10, 13, 14, 15, 17, 19, 21, 23, 25			C	1, 5, 10, 13, 15, 17
Neurons	50, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 130, 140, 150	Gamma	2, 1.5, 1, 0.5, 0.1, 0.01, 0.001, auto			Gamma	2.5, 2, 1.5, 1, 0.5
Activation	logistic, tanh, relu	Kernel	rbf, linear			Kernel	rbf, linear
Learning rate	constant, adaptive					LamU	0.5, 1, 1.5, 2
Solver	sgd, adam						

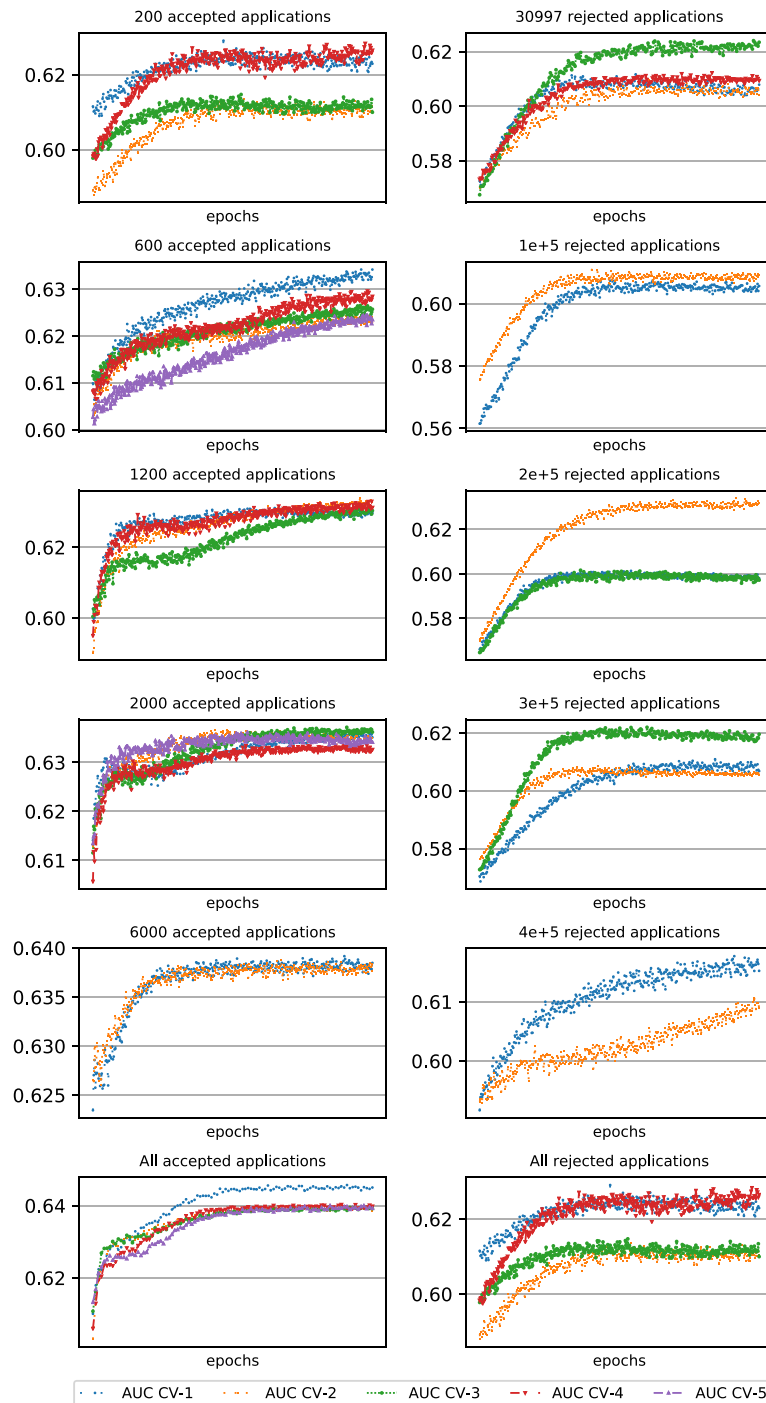


Fig. A.2. Model performance based on 5 cross-validations (CV) for the different scenarios analyzed in Table 4, using the Lending Club data set and Model 2. Since training for these scenarios in some cases become unstable, we keep only the results where Model 2 converged. Note that Model 2 achieves the highest AUC equal to 0.6450 in the *All* scenario in the left panel.

Table A.4

Empirical moment statistic for the default probability.

	Lending Club				Santander Credit Cards			
	Average	Std.	Kurtosis	Skewness	Average	Std.	Kurtosis	Skewness
MLP	0.1101	0.0096	-0.1027	0.0969	0.1180	0.0146	-0.0885	0.0563
SVM	0.1012	0.0130	-0.1505	0.0420	0.1202	0.0199	-0.1016	0.0517
Reclassification	0.1066	0.0083	-0.0635	-0.2861	0.1200	0.0011	6.1730	-0.8207
Fuzzy Parceling	0.1003	0.0132	-0.1389	0.0813	0.1198	0.0041	0.6406	-0.6061
Augmentation	0.0995	0.0131	-0.1487	0.0881	0.1198	0.0040	0.6285	-0.6151
Self-learning MLP	0.1055	0.0116	-0.0471	0.0770	0.1276	0.0058	0.2282	-0.5179
Self-learning SVM	0.1014	0.0130	-0.1494	0.0384	0.1257	0.0147	-0.1199	-0.0741
S3VM	0.1203	1.39e-6	-0.1173	-0.1297	0.1200	7.08e-7	0.7407	0.8687
Model 1	0.0985	0.0408	-0.5650	0.3368	0.1190	0.0367	-1.1459	-0.2455
Model 2	0.0999	0.0424	-0.5366	0.3819	0.0925	0.0340	0.8182	0.7802

the same (the default rate in 2009 is 12.59%, 2010 is 9.61%, 2011 is 10.32% and in 2012 is 13.76%) (see Figs. A.1 and A.2).

The second data set which we use in this research is provided by Santander Consumer Bank. The details that we can provide about this data set are limited by its proprietary nature. The descriptive statistics are shown in Table A.2.

Appendix B. Deriving the lower bounds

B.1. Lemma 1

Given two multivariate Gaussian distribution, with diagonal covariance matrix, $p(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\sigma}_1^2 \mathbf{I})$ and $q(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\sigma}_2^2 \mathbf{I})$, where $\boldsymbol{\mu}_i \in \mathbf{R}^d$ and $\boldsymbol{\sigma}_i^2 \in \mathbf{R}^d$, we have:

$$\int q(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} = \sum_{i=1}^d -\frac{1}{2} \log(2\pi \sigma_{1,i}^2) - \frac{\sigma_{2,i}^2}{2\sigma_{1,i}^2} - \frac{(\mu_{2,i} - \mu_{1,i})^2}{2\sigma_{1,i}^2}, \quad (\text{B.1})$$

where $\mu_{\cdot,i}$ and $\sigma_{\cdot,i}$ are the i 'th element of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ respectively.

Proof.

$$\begin{aligned} \int q(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} &= \int q(\mathbf{x}) \log \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \\ &\quad \times \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_1)\right) d\mathbf{x} \\ &= -\frac{1}{2} \log(2\pi \sigma_{1,i}^2) - \int q(\mathbf{x}) \frac{(x_i - \mu_{1,i})^2}{2\sigma_{1,i}^2} d\mathbf{x} - \dots \\ &\quad - \frac{1}{2} \log(2\pi \sigma_{1,d}^2) - \int q(\mathbf{x}) \frac{(x_d - \mu_{1,d})^2}{2\sigma_{1,d}^2} d\mathbf{x} \\ &= -\frac{1}{2} \log(2\pi \sigma_{1,i}^2) - \frac{\mathbb{E}_q[x_i^2] - 2\mathbb{E}_q[x_i]\mu_{1,i} + \mu_{1,i}^2}{2\sigma_{1,i}^2} - \dots \\ &\quad - \frac{1}{2} \log(2\pi \sigma_{1,d}^2) - \frac{\mathbb{E}_q[x_d^2] - 2\mathbb{E}_q[x_d]\mu_{1,d} + \mu_{1,d}^2}{2\sigma_{1,d}^2} \\ &= -\frac{1}{2} \log(2\pi \sigma_{1,i}^2) - \frac{\sigma_{2,i}^2 + \mu_{2,i}^2 - 2\mu_{2,i}\mu_{1,i} + \mu_{1,i}^2}{2\sigma_{1,i}^2} - \dots \\ &\quad - \frac{1}{2} \log(2\pi \sigma_{1,d}^2) - \frac{\sigma_{2,d}^2 + \mu_{2,d}^2 - 2\mu_{2,d}\mu_{1,d} + \mu_{1,d}^2}{2\sigma_{1,d}^2} \\ &= -\frac{1}{2} \log(2\pi \sigma_{1,i}^2) - \frac{\sigma_{2,i}^2 + (\mu_{2,i} - \mu_{1,i})^2}{2\sigma_{1,i}^2} - \dots \\ &\quad - \frac{1}{2} \log(2\pi \sigma_{1,d}^2) - \frac{\sigma_{2,d}^2 + (\mu_{2,d} - \mu_{1,d})^2}{2\sigma_{1,d}^2} \\ &= \sum_j^d -\frac{1}{2} \log(2\pi \sigma_{1,j}^2) - \frac{\sigma_{2,j}^2}{2\sigma_{1,j}^2} - \frac{(\mu_{2,j} - \mu_{1,j})^2}{2\sigma_{1,j}^2}. \quad (\text{B.2}) \end{aligned}$$

In the following sections we derive the lower bounds presented in the main text by taking the corresponding expectations, and using Lemma 1 where it is needed. We drop the subscripts θ and ϕ from the distributions $p(\cdot)$ and $q(\cdot)$, respectively, to do not clutter the notation. However, we use these subscripts in the parameters μ and σ to distinguish between them.

B.2. Model 1: supervised lower bound

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x},y)}[\log p(y)] = \int q(\mathbf{z}|\mathbf{x},y) \log p(y) d\mathbf{z}$$

$$\begin{aligned} &= \log \pi \\ \mathbb{E}_{q(\mathbf{z}|\mathbf{x},y)}[\log p(\mathbf{z}|y)] &= \int q(\mathbf{z}|\mathbf{x},y) \log p(\mathbf{z}|y) d\mathbf{z} \\ &= \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi^2) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\theta, \boldsymbol{\sigma}_\theta^2) d\mathbf{z} \\ &= -\sum_{j=1}^{\ell_z} \left(\frac{1}{2} \log(2\pi \sigma_{\theta_j,k}^2) + \frac{\sigma_{\phi_j}^2}{\sigma_{\theta_j,k}^2} \right. \\ &\quad \left. + \frac{(\mu_{\phi_j} - \mu_{\theta_j,k})^2}{\sigma_{\theta_j,k}^2} \right) \end{aligned}$$

$$\begin{aligned} \mathbb{E}_{q(\mathbf{z}|\mathbf{x},y)}[\log p(\mathbf{x}|\mathbf{z})] &= \int q(\mathbf{z}|\mathbf{x},y) \log p(\mathbf{x}|\mathbf{z}) d\mathbf{z} \\ &\approx \frac{1}{L} \sum_{l=1}^L \log \mathcal{N}(x_l|z_{l,i}) \\ \mathbb{E}_{q(\mathbf{z}|\mathbf{x},y)}[\log q(\mathbf{z}|\mathbf{x},y)] &= \int q(\mathbf{z}|\mathbf{x},y) \log q(\mathbf{z}|\mathbf{x},y) d\mathbf{z} \\ &= \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi^2) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi^2) d\mathbf{z} \\ &= -\sum_{j=1}^{\ell_z} \left(\frac{1}{2} \log(2\pi \sigma_{\phi_j}^2) + 1 \right) \end{aligned}$$

B.3. Model 1: unsupervised lower bound

$$\begin{aligned} \mathbb{E}_{q(\mathbf{z},y|\mathbf{x})}[\log p(y)] &= \sum_y \int q(y|\mathbf{x}) q(\mathbf{z}|\mathbf{x},y) \log p(y) d\mathbf{z} \\ &= \log \pi \\ \mathbb{E}_{q(\mathbf{z},y|\mathbf{x})}[\log p(\mathbf{z}|y)] &= \sum_y \int q(y|\mathbf{x}) q(\mathbf{z}|\mathbf{x},y) \log p(\mathbf{z}|y) d\mathbf{z} \\ &= \sum_y \pi_{y|\mathbf{x}} \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi^2) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\theta, \boldsymbol{\sigma}_\theta^2) d\mathbf{z} \\ &= -\sum_y \pi_{y|\mathbf{x}} \left[\sum_{j=1}^{\ell_z} \left(\frac{1}{2} \log(2\pi \sigma_{\theta_j,k}^2) \right. \right. \\ &\quad \left. \left. + \frac{\sigma_{\phi_j}^2}{\sigma_{\theta_j,k}^2} + \frac{(\mu_{\phi_j} - \mu_{\theta_j,k})^2}{\sigma_{\theta_j,k}^2} \right) \right] \\ \mathbb{E}_{q(\mathbf{z},y|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] &= \sum_y \int q(y|\mathbf{x}) q(\mathbf{z}|\mathbf{x},y) \log p(\mathbf{x}|\mathbf{z}) d\mathbf{z} \\ &\approx \frac{1}{L} \sum_{l=1}^L \log \mathcal{N}(x_l|z_{l,i}) \\ \mathbb{E}_{q(\mathbf{z},y|\mathbf{x})}[\log q(\mathbf{z}|\mathbf{x},y)] &= \sum_y \int q(y|\mathbf{x}) q(\mathbf{z}|\mathbf{x},y) \log q(\mathbf{z}|\mathbf{x},y) d\mathbf{z} \\ &= \sum_y \pi_{y|\mathbf{x}} \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi^2) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi^2) d\mathbf{z} \\ &= -\sum_y \pi_{y|\mathbf{x}} \sum_{j=1}^{\ell_z} \left(\frac{1}{2} \log(2\pi \sigma_{\phi_j}^2) + 1 \right) \\ \mathbb{E}_{q(\mathbf{z},y|\mathbf{x})}[\log q(y|\mathbf{x})] &= \sum_y \int q(y|\mathbf{x}) q(\mathbf{z}|\mathbf{x},y) \log q(y|\mathbf{x}) d\mathbf{z} \\ &= \sum_y q(y|\mathbf{x}) \log q(y|\mathbf{x}) \end{aligned}$$

B.4. Model 2: supervised lower bound

$$\begin{aligned}\mathbb{E}_{q(\mathbf{z}, \mathbf{a}|\mathbf{x}, y)}[\log p(y)] &= \int \int q(\mathbf{a}|\mathbf{x})q(\mathbf{z}|\mathbf{x}, y) \log p(y) dz da \\ &= \log \pi \\ \mathbb{E}_{q(\mathbf{z}, \mathbf{a}|\mathbf{x}, y)}[\log p(\mathbf{z}|y)] &= \int \int q(\mathbf{a}|\mathbf{x})q(\mathbf{z}|\mathbf{x}, y) \log p(\mathbf{z}|y) dz da \\ &= \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi, \boldsymbol{\sigma}_\phi^2) \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\theta, \boldsymbol{\sigma}_\theta^2) dz \\ &= - \sum_{j=1}^{\ell_z} \left(\frac{1}{2} \log(2\pi \sigma_{\theta_{j,k}}^2) + \frac{\sigma_{\phi_j}^2}{\sigma_{\theta_{j,k}}^2} + \frac{(\mu_{\phi_j} - \mu_{\theta_{j,k}})^2}{\sigma_{\theta_{j,k}}^2} \right) \\ \mathbb{E}_{q(\mathbf{z}, \mathbf{a}|\mathbf{x}, y)}[\log p(\mathbf{x}|\mathbf{z}, y)] &= \int \int q(\mathbf{a}|\mathbf{x})q(\mathbf{z}|\mathbf{x}, y) \log p(\mathbf{x}|\mathbf{z}, y) dz da \\ &\approx \frac{1}{L} \sum_{l=1}^L \log \mathcal{N}(x_i|z_{i,l}, y_i) \\ \mathbb{E}_{q(\mathbf{z}, \mathbf{a}|\mathbf{x}, y)}[\log p(\mathbf{a}) - \log q(\mathbf{a}|\mathbf{x})] &= \int \int q(\mathbf{a}|\mathbf{x})q(\mathbf{z}|\mathbf{x}, y) [\log p(\mathbf{a}) - \log q(\mathbf{a}|\mathbf{x})] dz da \\ &= \int q(\mathbf{a}|\mathbf{x}) \log p(\mathbf{a}) da - \int q(\mathbf{a}|\mathbf{x}) \log q(\mathbf{a}|\mathbf{x}) da \\ &= -\frac{1}{2} \sum_{c=1}^{\ell_a} (\sigma_{\phi_{ac}}^2 + \mu_{\phi_{ac}}^2 - (1 + \log \sigma_{\phi_{ac}}^2)) \\ \mathbb{E}_{q(\mathbf{z}, \mathbf{a}|\mathbf{x}, y)}[\log q(\mathbf{z}|\mathbf{x}, y)] &= \int \int q(\mathbf{a}|\mathbf{x})q(\mathbf{z}|\mathbf{x}, y) \log q(\mathbf{z}|\mathbf{x}, y) dz da \\ &= \int q(\mathbf{z}|\mathbf{x}, y) \log q(\mathbf{z}|\mathbf{x}, y) dz \\ &= \frac{1}{2} \sum_{j=1}^{\ell_z} (1 + \log \sigma_{\phi_{z_j}}^2)\end{aligned}$$

B.5. Model 2: unsupervised lower bound

$$\begin{aligned}\mathbb{E}_{q(\mathbf{z}, \mathbf{a}, y|\mathbf{x})}[\log p(y)] &= \int \sum_y \int q(\mathbf{a}|\mathbf{x})q(y|\mathbf{x}, \mathbf{a})q(\mathbf{z}|\mathbf{x}, y) \log p(y) dz da \\ &= \log \pi \\ \mathbb{E}_{q(\mathbf{z}, \mathbf{a}, y|\mathbf{x})}[\log q(y|\mathbf{x}, \mathbf{a})] &= \int \sum_y \int q(\mathbf{a}|\mathbf{x})q(y|\mathbf{x}, \mathbf{a})q(\mathbf{z}|\mathbf{x}, y) \\ &\quad \times \log q(y|\mathbf{x}, \mathbf{a}) dz da \\ &\approx \frac{1}{L_a} \sum_{l_a=1}^{L_a} \sum_y q(y|\mathbf{x}, \mathbf{a}_{l_a}) \log q(y|\mathbf{x}, \mathbf{a}_{l_a}) \\ \mathbb{E}_{q(\mathbf{z}, \mathbf{a}, y|\mathbf{x})}[\log p(\mathbf{z}|y)] &= \int \sum_y \int q(\mathbf{a}|\mathbf{x})q(y|\mathbf{x}, \mathbf{a})q(\mathbf{z}|\mathbf{x}, y) \log p(\mathbf{z}|y) dz da \\ &\approx \frac{1}{L_a} \sum_{l_a=1}^{L_a} \sum_y q(y|\mathbf{x}, \mathbf{a}_{l_a}) \int q(\mathbf{z}|\mathbf{x}, y_{l_a}) \log p(\mathbf{z}|y_{l_a}) dz \\ &\approx -\frac{1}{L_a} \sum_{l_a=1}^{L_a} \sum_y \pi_{y|\mathbf{x}, \mathbf{a}_{l_a}} \left[\sum_{j=1}^{\ell_z} \left(\frac{1}{2} \log(2\pi \sigma_{\theta_{j,k}}^2) + \frac{\sigma_{\phi_j}^2}{\sigma_{\theta_{j,k}}^2} \right) \right]\end{aligned}$$

$$\begin{aligned}&+ \left. \frac{(\mu_{\phi_j} - \mu_{\theta_{j,k}})^2}{\sigma_{\theta_{j,k}}^2} \right) \\ \mathbb{E}_{q(\mathbf{z}, \mathbf{a}, y|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}, y)] &= \int \sum_y \int q(\mathbf{a}|\mathbf{x})q(y|\mathbf{x}, \mathbf{a})q(\mathbf{z}|\mathbf{x}, y) \log p(\mathbf{x}|\mathbf{z}, y) dz da \\ &\approx \frac{1}{L_a} \sum_{l_a=1}^{L_a} \sum_y \pi_{y|\mathbf{x}, \mathbf{a}_{l_a}} \frac{1}{L_z} \sum_{l_z=1}^{L_z} \log \mathcal{N}(x_i|z_{i,l}, y_{l_a}) \\ \mathbb{E}_{q(\mathbf{z}, \mathbf{a}, y|\mathbf{x})}[\log p(\mathbf{a}) - \log q(\mathbf{a}|\mathbf{x})] &= \int \sum_y \int q(\mathbf{a}|\mathbf{x})q(y|\mathbf{x}, \mathbf{a})q(\mathbf{z}|\mathbf{x}, y) [\log p(\mathbf{a}) - \log q(\mathbf{a}|\mathbf{x})] dz da \\ &= \sum_y q(\mathbf{a}|\mathbf{x}) \left[\int q(y|\mathbf{x}, \mathbf{a}) \log p(\mathbf{a}) da - \int q(\mathbf{a}|\mathbf{x}) \log q(\mathbf{a}|\mathbf{x}) da \right] \\ &= -\frac{1}{2} \sum_y \pi_{y|\mathbf{x}, \mathbf{a}_{l_a}} \left[\sum_{c=1}^{\ell_a} (\sigma_{\phi_{ac}}^2 + \mu_{\phi_{ac}}^2 - (1 + \log \sigma_{\phi_{ac}}^2)) \right] \\ \mathbb{E}_{q(\mathbf{z}, \mathbf{a}, y|\mathbf{x})}[\log q(\mathbf{z}|\mathbf{x}, y)] &= \int \sum_y \int q(\mathbf{a}|\mathbf{x})q(y|\mathbf{x}, \mathbf{a})q(\mathbf{z}|\mathbf{x}, y) \log q(\mathbf{z}|\mathbf{x}, y) dz da \\ &\approx \frac{1}{L_a} \sum_{l_a=1}^{L_a} \sum_y q(y|\mathbf{x}, \mathbf{a}_{l_a}) \int q(\mathbf{z}|\mathbf{x}, y) \log q(\mathbf{z}|\mathbf{x}, y) dz \\ &= -\frac{1}{L_a} \sum_{l_a=1}^{L_a} \sum_y \pi_{y, \mathbf{a}_{l_a}} \left[\frac{1}{2} \sum_{j=1}^{\ell_z} (1 + \log \sigma_{\phi_{z_j}}^2) \right]\end{aligned}$$

References

- [1] Raymond Anderson, *The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Management and Decision Automation*, Oxford University Press, 2007.
- [2] Michael Bucker, Maarten van Kampen, Walter Krämer, Reject inference in consumer credit scoring with nonignorable missing data, *J. Bank. Financ.* 37 (3) (2013) 1040–1045.
- [3] Ha-Thu Nguyen, Reject inference in application scorecards: evidence from France, *EconomiX Working Papers 2016–10*, University of Paris Nanterre, EconomiX, 2016, URL <https://ideas.repec.org/p/drm/wpaper/2016-10.html>.
- [4] G. Gary Chen, Thomas Astebro, The economic value of reject inference in credit scoring, *Department of Management Science, University of Waterloo*, 2001.
- [5] Andrew Marshall, Leilei Tang, Alistair Milne, Variable reduction, sample selection bias and bank retail credit scoring, *J. Empir. Financ.* 17 (3) (2010) 501–512.
- [6] David J. Hand, William E. Henley, Can reject inference ever work? *IMA J. Manag. Math.* 5 (1) (1993) 45–55.
- [7] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, Max Welling, Semi-supervised learning with deep generative models, in: *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.
- [8] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, Ole Winther, Auxiliary deep generative models, 2016, arXiv preprint [arXiv:1602.05473](https://arxiv.org/abs/1602.05473).
- [9] Chuhan Wu, Fangzhao Wu, Sixing Wu, Zhigang Yuan, Junxin Liu, Yongfeng Huang, Semi-supervised dimensional sentiment analysis with variational autoencoder, *Knowl.-Based Syst.* 165 (2019) 30–39.
- [10] Xianghua Fu, Yanzhi Wei, Fan Xu, Ting Wang, Yu Lu, Jianqiang Li, Joshua Zhexue Huang, Semi-supervised aspect-level sentiment classification model based on variational autoencoder, *Knowl.-Based Syst.* 171 (2019) 81–92.
- [11] Yin Zheng, Huachun Tan, Bangsheng Tang, Hanning Zhou, et al., Variational deep embedding: A generative approach to clustering, *1(2)(2016) 5*, arXiv preprint [arXiv:1611.05148](https://arxiv.org/abs/1611.05148).
- [12] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, Samy Bengio, Generating sentences from a continuous space, 2015, arXiv preprint [arXiv:1511.06349](https://arxiv.org/abs/1511.06349).
- [13] Xianxu Hou, Linlin Shen, Ke Sun, Guoping Qiu, Deep feature consistent variational autoencoder, in: *Applications of Computer Vision (WACV)*, 2017 IEEE Winter Conference on, IEEE, 2017, pp. 1133–1141.

- [14] Siddique Latif, Rajib Rana, Junaid Qadir, Julien Epps, Variational autoencoders for learning latent representations of speech emotion, 2017, arXiv preprint [arXiv:1712.08708](https://arxiv.org/abs/1712.08708).
- [15] Rogelio Andrade Mancisidor, Michael Kampffmeyer, Kjersti Aas, Robert Jensen, Learning latent representations of bank customers with the variational autoencoder, 2019, arXiv preprint [arXiv:1903.06580](https://arxiv.org/abs/1903.06580).
- [16] Derrick N. Joanes, Reject inference applied to logistic regression for credit scoring, *IMA J. Manag. Math.* 5 (1) (1993) 35–43.
- [17] A.J. Feelders, Credit scoring and reject inference with mixture models, *Intell. Syst. Account. Financ. Manage.* 9 (1) (2000) 1–8.
- [18] Jonathan Banasik, John Crook, Lyn Thomas, Sample selection bias in credit scoring models, *J. Oper. Res. Soc.* 54 (8) (2003) 822–832.
- [19] Jonathan Crook, John Banasik, Does reject inference really improve the performance of application scoring models? *J. Bank. Financ.* 28 (4) (2004) 857–874.
- [20] Geert Verstraeten, Dirk Van den Poel, The impact of sample bias on consumer credit scoring performance and profitability, *J. Oper. Res. Soc.* 56 (8) (2005) 981–992.
- [21] J. Banasik, J. Crook, Credit scoring, augmentation and lean models, *J. Oper. Res. Soc.* 56 (9) (2005) 1072–1081, <http://dx.doi.org/10.1057/palgrave.jors.2602017>.
- [22] So Young Sohn, H.W. Shin, Reject inference in credit operations based on survival analysis, *Expert Syst. Appl.* 31 (1) (2006) 26–29.
- [23] John Banasik, Jonathan Crook, Reject inference, augmentation, and sample selection, *European J. Oper. Res.* 183 (3) (2007) 1582–1594.
- [24] Y. Kim, S.Y. Sohn, Technology scoring model considering rejected applicants and effect of reject inference, *J. Oper. Res. Soc.* 58 (10) (2007) 1341–1347.
- [25] L-Ding Wu, David J. Hand, Handling selection bias when choosing actions in retail credit applications, *European J. Oper. Res.* 183 (3) (2007) 1560–1568.
- [26] J. Banasik, J. Crook, Reject inference in survival analysis by augmentation, *J. Oper. Res. Soc.* 61 (3) (2010) 473–485.
- [27] Sebastián Maldonado, Gonzalo Paredes, A semi-supervised approach for reject inference in credit scoring using svms, in: *Industrial Conference on Data Mining*, Springer, 2010, pp. 558–571.
- [28] Gongyue Gary Chen, Thomas Astebro, Bound and collapse bayesian reject inference for credit scoring, *J. Oper. Res. Soc.* 63 (10) (2012) 1374–1387.
- [29] Billie Anderson, J. Michael Hardin, Modified logistic regression using the EM algorithm for reject inference, *Int. J. Data Anal. Tech. Strateg.* 5 (4) (2013) 359–373.
- [30] Zhiyong Li, Ye Tian, Ke Li, Fanyin Zhou, Wei Yang, Reject inference in credit scoring using semi-supervised support vector machines, *Expert Syst. Appl.* 74 (2017) 105–114.
- [31] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, Ole Winther, Improving semi-supervised learning with auxiliary deep generative models, in: *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2015.
- [32] David C. Hsia, Credit scoring and the equal credit opportunity act, *Hastings Law J.* 30 (1978) 371.
- [33] C. Rosenberg, M. Hebert, H. Schneiderman, Semi-supervised self-training of object detection models, in: *2005 Seventh IEEE Workshops on Applications of Computer Vision*, vol. 1, 2005, pp. 29–36, <http://dx.doi.org/10.1109/ACVMOT.2005.107>.
- [34] Fabian Gieseke, Antti Airola, Tapio Pahikkala, Oliver Kramer, Sparse quasi-newton optimization for semi-supervised support vector machines, in: *ICPRAM* (1), 2012, pp. 45–54.
- [35] David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams, Learning internal representations by error propagation, Technical Report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- [36] Corinna Cortes, Vladimir Vapnik, Support-vector networks, *Mach. Learn.* 20 (3) (1995) 273–297.
- [37] Lyn C. Thomas, A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers, *Int. J. Forecast.* 16 (2) (2000) 149–172.
- [38] Dennis Ash, Steve Meester, Best practices in reject inferencing, in: *Conference on Credit Risk Modeling and Decisioning*: Philadelphia, PA., 01, 2002.
- [39] James J. Heckman, The common structure of statistical models of truncation, sample selection and limited dependent variables and a simple estimator for such models, in: *Annals of Economic and Social Measurement*, vol. 5, number 4, NBER, 1976, pp. 475–492.
- [40] James J. Heckman, Sample selection bias as a specification error, *Econometrica* 47 (1) (1979) 153–161.
- [41] William J. Boyes, Dennis L. Hoffman, Stuart A. Low, An econometric analysis of the bank credit scoring problem, *J. Econometrics* 40 (1) (1989) 3–14.
- [42] William Greene, Sample selection in credit-scoring models, *Jpn. World Econ.* 10 (3) (1998) 299–316.
- [43] Patrick Puhani, The heckman correction for sample selection and its critique, *J. Econ. Surv.* 14 (1) (2000) 53–68.
- [44] Ye Tian, Jian Luo, A new branch-and-bound approach to semi-supervised support vector machine, *Soft Comput.* 21 (1) (2017) 245–254, <http://dx.doi.org/10.1007/s00500-016-2089-y>.
- [45] Diederik P. Kingma, Max Welling, Auto-encoding variational bayes, 2013, arXiv preprint [arXiv:1312.6114](https://arxiv.org/abs/1312.6114).
- [46] Danilo Jimenez Rezende, Shakir Mohamed, Daan Wierstra, Stochastic backpropagation and approximate inference in deep generative models, 2014, arXiv preprint [arXiv:1401.4082](https://arxiv.org/abs/1401.4082).
- [47] Håvard Kvamme, Nikolai Sellereite, Kjersti Aas, Steffen Sjørnsen, Predicting mortgage default using convolutional neural networks, *Expert Syst. Appl.* 102 (2018) 207–217.
- [48] Cheng Zhang, Judith Butepage, Hedvig Kjellstrom, Stephan Mandt, Advances in variational inference, *IEEE Trans. Pattern Anal. Mach. Intell.* (2018).
- [49] Meelis Kull, Telmo Silva Filho, Peter Flach, Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers, in: *Artificial Intelligence and Statistics*, 2017, pp. 623–631.
- [50] Theano Development Team, Theano: a Python framework for fast computation of mathematical expressions, 2016, arXiv e-prints, [abs/1605.02688](https://arxiv.org/abs/1605.02688), <http://arxiv.org/abs/1605.02688>.
- [51] Diederik P. Kingma, Jimmy Ba, Adam: a method for stochastic optimization, 2014, arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [52] Mengchen Liu, Jiaxin Shi, Kelei Cao, Jun Zhu, Shixia Liu, Analyzing the training processes of deep generative models, *IEEE Trans. Vis. Comput. Graphics* 24 (1) (2018) 77–87.
- [53] Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, Satoshi Yagi, Student-t variational autoencoder for robust density estimation, in: *IJCAI*, 2018, pp. 2696–2702.
- [54] Xavier Glorot, Yoshua Bengio, Understanding the difficulty of training deep feedforward neural networks, in: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.
- [55] John Platt, et al., Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in: *Advances in Large Margin Classifiers*, vol. 10 (3), 1999, pp. 61–74.

Chapter 11

Paper III

Generating Customer’s Credit Behavior with Deep Generative Models

Rogelio A. Mancisidor^{a,†,*}
rogelio.a.mancisidor@uit.no

Michael Kampffmeyer^{a,b,†}
michael.c.kampffmeyer@uit.no

Kjersti Aas^b
kjersti@nr.no

Robert Jenssen^{a,b,†}
robert.jenssen@uit.no

^a Department of Physics and Technology, Faculty of Science and
Technology, UiT The Arctic University of Norway, Hansine Hansens veg 18, Tromsø 9037, Norway

^bNorwegian Computing Center, P.O. Box 114 Blindern, Oslo, Norway

[†]RAM, MK, and RJ are all with the UiT Machine Learning Group: <http://machine-learning.uit.no>

*Corresponding author

September 23, 2020

Abstract. Banks collect data \mathbf{x}_1 in loan applications to decide whether to grant credit and accepted applications generate new data \mathbf{x}_2 throughout the loan period. Hence, banks have two measurement-modalities, which provide a complete picture about customers. If we can generate \mathbf{x}_2 conditioned on \mathbf{x}_1 keeping the relationship between these two modalities, credit and behavior scoring may be enabled simultaneously (at the time \mathbf{x}_1 is obtained) to support cross-selling, launching of new products or marketing campaigns. Therefore, we develop a novel conditional bi-modal discriminative (CBMD) model for credit scoring, which is able to generate \mathbf{x}_2 based on \mathbf{x}_1 and can classify the outcome of loans in an unified framework. The idea behind CBMD is to learn shared (among modalities) latent representations that are useful to generate \mathbf{x}_2 using the available data \mathbf{x}_1 during the application process. The classifier model introduced in CBMD encourages the generative process to generate \mathbf{x}_2 accurately. Further, CBMD optimizes a novel objective function introduced in this research, which maximizes mutual information between the latent representation \mathbf{z} and view \mathbf{x}_2 to improve the generative process in the model. We benchmark the generative process of our proposed model and CBMD outperforms other multi-learning models. Similarly, the classification performance of CBMD is tested under different scenarios and it achieves higher or on a par model performance compared to the state-of-the-art in multi-modal learning models.

Keywords: Multi-modal learning, Credit Scoring, Deep Generative Models, Representation Learning

1 Introduction

Retail banks model the relationship between customers’ information \mathbf{x} and the outcome y of a loan to decide whether to grant credit, where $y = 0$ if a customer repays the loan otherwise $y = 1$. Traditionally, \mathbf{x} has been limited to information captured prior the application process, even though banks have access to more data that is generated by granted applications throughout the loan period, e.g. repayment or purchase behavior. Therefore, banks have two measurement-modalities that provide complementary information about a given customer. The first data modality, or view of data, is generated before the

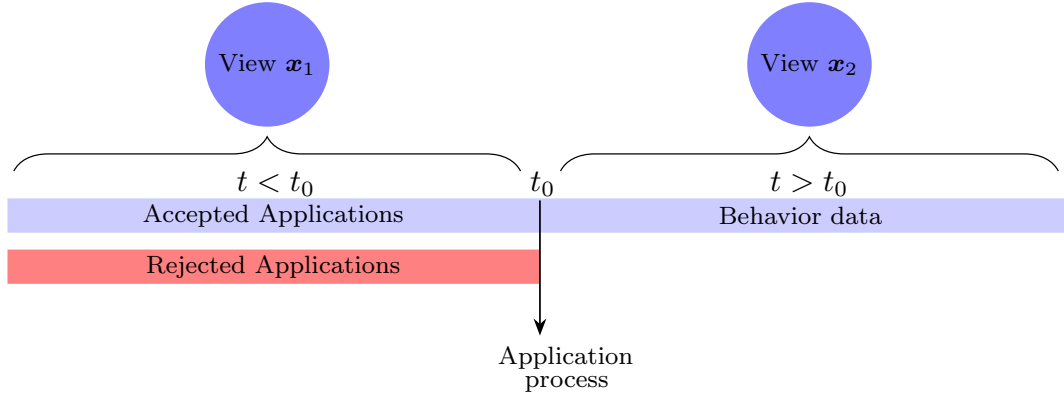


Figure 1: Bi-modal credit data. At the time of the applications process t_0 , only \mathbf{x}_1 is available. This data view, which commonly is composed of socio-demographic features, is generated during $t < t_0$ and is used in credit scoring models. After the loan is granted, a new view of data \mathbf{x}_2 is generated, providing complementary information about the customer. View \mathbf{x}_2 is used to develop behavior models or to support cross-selling activities among others.

loan is granted and we denote it as \mathbf{x}_1 . The second view is generated throughout the loan period and we called this modality \mathbf{x}_2 , see Figure 1. Commonly, banks use \mathbf{x}_1 to develop credit scoring models, while \mathbf{x}_2 can be used to develop behavior models or to support cross-selling activities, launching of new products or marketing campaigns in banks.

Multi-modal learning designs models that utilize different measurements-modalities of the same object to learn common data representations between modalities. Examples of multi-modalities, or views of data, are audio, video, and text, words and context, or credit data before and after the application process. A traditional application for multi-modal learning is downstream classification in two steps [1, 2]. That is, a common data representation between modalities is learned in the first stage and then, in the second stage, it is used to train a classifier model.

Some multi-modal learning models are able to generate the input modalities using autoencoder-like architectures¹, which clearly requires that measurement-modalities are available at test time. This is not the case in the context of credit scoring, where \mathbf{x}_2 is not available at the same time as \mathbf{x}_1 . If we can generate \mathbf{x}_2 conditioned on \mathbf{x}_1 keeping the relationship between these two modalities, credit and behavior scoring may be enabled simultaneously (at the time \mathbf{x}_1 is obtained) to support cross-selling, launching of new products or marketing campaigns. Therefore, the main motivation for this research is to develop a novel bi-modal methodology that generates the view \mathbf{x}_2 based on \mathbf{x}_1 , which is our best source of information for future customer behavior.

To that end, we develop a conditional bi-modal discriminative (CBMD) model that i) learns to generate \mathbf{x}_2 conditioned on view \mathbf{x}_1 and a private latent representation \mathbf{z} , and ii) can classify class labels y using data representations. Note that \mathbf{x}_1 is not only our best source of information about \mathbf{x}_2 , but using \mathbf{x}_1 to generate \mathbf{x}_2 helps to keep the relationship between these modalities for each customer. Further, the reason to include a classifier model is to further improve the generative properties in CBMD. The optimization of the classifier encourages the generative process to generate \mathbf{x}_2 accurately, similarly to the synergy between representation learning and classifier training shown by [3] in classification tasks. Therefore, our proposed methodology generates \mathbf{x}_2 , learns latent representations and trains a classifier in an unified framework. This makes our proposed CBMD model useful to generate future credit data and for downstream classification in scenarios where only \mathbf{x}_1 is available at test time². Note that extending the CBMD model to multiple views of data is possible by concatenating multiple views with \mathbf{x}_1 .

The contributions of this paper are as follows: i) we develop the first bi-modal learning methodology for credit scoring, which generates the view \mathbf{x}_2 conditioned on view \mathbf{x}_1 and can classify the outcome of

¹Such an architecture is designed to reconstruct the input data, i.e. $f(\mathbf{x}_2) = \mathbf{z}$ and $f(\mathbf{z}) = \hat{\mathbf{x}}_2$ where $f(\cdot)$ is a neural network.

²This is the same test scenario considered in [1, 2]

loans using private latent representations, ii) we show how can we utilize the generative properties of our proposed CBMD model to generate future credit data, and iii) we introduce a novel lower bound that maximizes mutual information between the common latent representation \mathbf{z} and view \mathbf{x}_2 , which helps to improve the generative process of our proposed CBMD model.

The rest of the paper is organized as follows. Section 2 reviews the related work on multi-modal learning and Section 3 presents the proposed model. Further, Section 4 explains the data sets used in this research and presents the benchmark results. Finally, Section 5 discusses the main findings of this research.

2 Related Work

This section reviews the research on multi-modal learning focusing on the development from the seminal canonical correlation analysis (CCA) [4] to models that optimize a lower bound derived with variational inference and use neural networks to do amortized inference about model parameters. To facilitate model comparison, we use a common notation for all models where different data modalities are represented by \mathbf{x} and are distinguished with a subscript, common latent transformations are represented by \mathbf{z} , private latent representations are denoted by \mathbf{h} and a subscript referring to their data modality. Finally, labels are denoted by y . The plate notation for variational-based models included in this section are shown in Table 1.

Canonical correlation analysis finds linear projections by maximizing correlation between the transformations in multi-modal data. The objective is to learn the underlying semantic in the different modalities [5]. Originally, CCA deals only with linear projections of the data, but a kernel version of CCA was introduced in [5, 6, 7, 8, 9] to handle non-linearities³.

Both CCA and kernel-CCA maximize

$$\{f, g\} = \arg \max_{f, g} \frac{\text{cov}(f(\mathbf{x}_1), g(\mathbf{x}_2))}{\sqrt{\text{var}(f(\mathbf{x}_1)) \cdot \text{var}(g(\mathbf{x}_2))}}, \quad (1)$$

where $f(\mathbf{x}_1)$ and $g(\mathbf{x}_2)$ are the projections of views \mathbf{x}_1 and \mathbf{x}_2 , subject to the constraints that $f_j(\mathbf{x}_1)$ is uncorrelated with $f_i(\mathbf{x}_1)$, $g_j(\mathbf{x}_2)$ is uncorrelated with $g_i(\mathbf{x}_2)$, and $f_j(\mathbf{x}_1)$ is uncorrelated with $g_i(\mathbf{x}_2)$ for all $i \neq j$. The difference between CCA and kernel-CCA is that the former assumes linear projections i.e. $f(\mathbf{x}_1) = \mathbf{v}^T \mathbf{x}_1$, while the latter uses linear combinations of the kernel k_1 evaluated at the data set, i.e. $f(\mathbf{x}_1) = \sum_{i=1}^N \alpha_i k_1(\mathbf{x}_1, \mathbf{x}_{1,i})$, where α_i determines the direction of the projections. Similar functions are used for the projection $g(\mathbf{x}_2)$.

A probabilistic interpretation of CCA is presented in [10]. The views $\mathbf{x}_1 \in \mathbb{R}^{d_1}$ and $\mathbf{x}_2 \in \mathbb{R}^{d_2}$ are generated given a common latent representation \mathbf{z} , that is

$$\begin{aligned} \mathbf{z} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \\ \mathbf{x}_1 | \mathbf{z} &\sim \mathcal{N}(\mathbf{W}_1 \mathbf{z} + \boldsymbol{\mu}_1, \boldsymbol{\Psi}_1), \\ \mathbf{x}_2 | \mathbf{z} &\sim \mathcal{N}(\mathbf{W}_2 \mathbf{z} + \boldsymbol{\mu}_2, \boldsymbol{\Psi}_2), \end{aligned}$$

where $\min(d_1, d_2) \geq d \geq 1$ and $\mathbf{W}_1, \mathbf{W}_2, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Psi}_1$, and $\boldsymbol{\Psi}_2$ are parameters defining a Gaussian distribution $\mathcal{N}(\cdot)$. These parameters are commonly estimated using the expectation-maximization (EM) algorithm [11] and their updating equations can be found in [10]. Furthermore, [10] show that linear discriminant analysis (LDA) [12] is a special case of CCA where one of the views is the label y .

Deep canonical correlation analysis [15] (DCCA) couple together deep neural networks and CCA with the objective to train neural networks able to maximize the correlation $\rho(f(\mathbf{x}_1), g(\mathbf{x}_2))$ between view \mathbf{x}_1 and \mathbf{x}_2 . DCCA cannot only handle non-linearities, but can also capture high-level abstractions of the data in each of the multiple hidden layers. Note that the correlation objective function is a function of the entire data set, i.e. it is a fully batch objective function, and therefore it can be costly for large data sets. In a similar approach, [1] develop a model called deep canonically correlated autoencoder

³The method presented in [9] is an approximation based on random Fourier features.

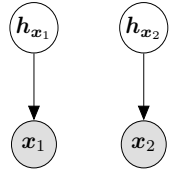
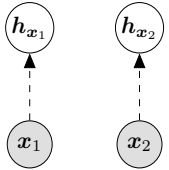
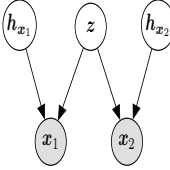
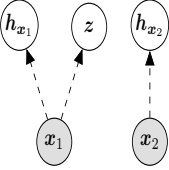
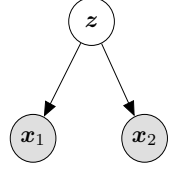
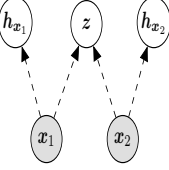
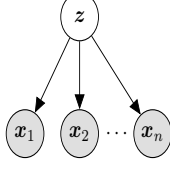
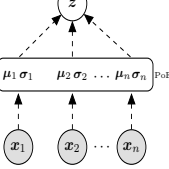
(Year) Author	Generative Model	Inference Model	Learning Approach
(2015) Wang W. [1]			<ul style="list-style-type: none"> • Unsupervised representation learning • Loss function: AE + ACC • Training: SGD
(2016) Wang W. [2]			<ul style="list-style-type: none"> • Unsupervised representation learning • Loss function: VI lower bound • Training: SGD
(2016) Suzuki M. [13]			<ul style="list-style-type: none"> • Unsupervised representation learning • Loss function: VI lower bound • Training: SGD
(2018) Wu M. [14]			<ul style="list-style-type: none"> • Unsupervised representation learning • Loss function: VI lower bound with product of experts (PoE) • Training: SGD

Table 1

(DCCA), where the objective function minimizes reconstruction error for both views (as in regular autoencoders) and optimizes canonical correlation between the learned representations (as in CCA). The main difference between DCCA and DCCA is that the latter can reconstruct both view \mathbf{x}_1 and \mathbf{x}_2 , and DCCA scales to large data sets using stochastic gradient descent to optimize its objective function.

A problem with DCCA is that the CCA term in its objective function dominates the optimization procedure [1]. As a consequence, the reconstruction of \mathbf{x}_1 and \mathbf{x}_2 is poor. To overcome this problem, [2] use variational inference and deep generative models to generate latent representations of the input views and to reconstruct them. The authors in [2] presented two model versions. In the first version, which is called variational CCA (VCCA), the authors use a common latent variable to generate both views, while the second version uses common and private latent variables to generate views \mathbf{x}_1 and \mathbf{x}_2 . When only common latent variables are used, it is not clear how to specify the inference model, i.e. $q(\mathbf{z}|\mathbf{x}_1)$ or $q(\mathbf{z}|\mathbf{x}_2)$. Therefore, the authors propose the objective function $\mathcal{L} = \mu\mathcal{L}_{q(\mathbf{z}|\mathbf{x}_1)} + (1 - \mu)\mathcal{L}_{q(\mathbf{z}|\mathbf{x}_2)}$, where $\mathcal{L}_{q(\mathbf{z}|\mathbf{x}_1)}$ ($\mathcal{L}_{q(\mathbf{z}|\mathbf{x}_2)}$) is the loss function when $q(\mathbf{z}|\mathbf{x}_1)$ ($q(\mathbf{z}|\mathbf{x}_2)$) defines the inference model and $\mu \in [0, 1]$ is a weight parameter controlling the importance of each term in the objective function.

A supervised extension of VCCA is proposed by [3], which combines multi-modal learning and classification in one unified framework. The authors propose a discriminative multi-modal deep generative model (DMDGM) that generates both modalities based on private and common hidden variables. Further, the classification in DMDGM is done using the available views at test time, e.g. $q(y|\mathbf{x}_1)$ or $q(y|\mathbf{x}_1, \mathbf{x}_2)$. This is not the only model where classification is addressed in a unified objective function, [16] develops a semi-supervised deep generative model for missing modalities where the latent variable is shared across modalities. To further improve the flexibility of the latent space, the authors model the inference process as a Gaussian mixture model (GMM). However, it is worth mentioning that modeling the inference

Table 1 Continued			
(Year) Author	Generative Model	Inference Model	Learning Approach
(2018) Du C. [16]			<ul style="list-style-type: none"> • Semi-supervised classification • Loss function: VI lower bound • Training: SGD
(2018) Vedantam R. [17]			<ul style="list-style-type: none"> • Supervised representation learning • Loss function: VI lower bound • Training: SGD
(2019) Du F. [3]			<ul style="list-style-type: none"> • Supervised classification • Loss function: VI lower bound • Training: SGD

Table 1: Overview over some generative and inference models presented in Section 2. We have harmonize the notation in all previous models with the one used in this paper. That is, given a bi-modal data, view \mathbf{x}_1 is available during training and test time, while view \mathbf{x}_2 is only available during training. Furthermore, common latent variables are denoted by \mathbf{z} , while private latent representations are represented by $\mathbf{h}_{\mathbf{x}_1}$ and $\mathbf{h}_{\mathbf{x}_2}$.

process as GMM harms the tightness of the lower bound since the entropy of a GMM is intractable.

The joint multimodal variational autoencoder (JMVAE) is introduced in [13]. The first model presented by the authors replaces missing views with zeros, e.g. $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2) \approx q(\mathbf{z}|\mathbf{x}_1, \mathbf{0})$ if \mathbf{x}_2 is missing. The second model presented in [13] includes two individual inference models $q(\mathbf{z}|\mathbf{x}_1)$ and $q(\mathbf{z}|\mathbf{x}_2)$, and one global inference model $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$. Further, the objective function includes two Kullback-Leibler (KL) divergence terms, $KL[q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)||q(\mathbf{z}|\mathbf{x}_1)]$ and $KL[q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)||q(\mathbf{z}|\mathbf{x}_2)]$, which force $q(\mathbf{z}|\mathbf{x}_1)$ and $q(\mathbf{z}|\mathbf{x}_2)$ to be close to $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$. The authors argue that including these two KL terms is equivalent to minimizing the lower bound of variation of information (VaI). This is not the only model optimizing the information theoretic measure VaI, [18] use restricted Boltzmann machines to develop a multi-modality model, which objective function is fully derived from a VaI perspective.

All previous models in this section assume data with only two views. A model that generalizes to more than two modalities is presented in [14]. Their deep generative model assumes that the posterior distribution $p(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ is proportional to the product of individual posteriors $p(\mathbf{z}|\mathbf{x}_1) \dots p(\mathbf{z}|\mathbf{x}_n)$ normalized by the prior distribution $p(\mathbf{z})$. Additionally, they assume that individual posteriors are approximated by variational densities $q(\mathbf{z}|\mathbf{x}_i)$ for $i = 1, \dots, n$. Hence, the joint posterior distribution is a product of experts (PoEs). Another model using PoEs is presented in [17]. However, in this case, the authors use a PoEs to deal with missing modalities, i.e. $q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2) \propto p(\mathbf{h}_{\mathbf{x}_2}) \prod_{k \in \mathcal{O}} q(\mathbf{h}_{\mathbf{x}_2}|x_2^k)$, where \mathcal{O} are the observed attributes in view \mathbf{x}_2 .

Our proposed CBMD model uses a private distribution $p(\mathbf{z}|\mathbf{x}_1)$ that is conditioned on view \mathbf{x}_1 to generate the future view \mathbf{x}_2 with a generative process $p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})$. Such a generative mechanism keeps the relationship between \mathbf{x}_1 and \mathbf{x}_2 and allows us to generate \mathbf{x}_2 at the same time a loan application is received. Inspired by the methodology introduced in [19], CBMD optimizes a lower bound which maximizes mutual information between latent representations \mathbf{z} and view \mathbf{x}_2 to effectively learn amortized inference distributions and generates quality $\hat{\mathbf{x}}_2$ samples. Note that CBMD uses latent representations for both drawing \mathbf{x}_2 and for downstream classification; hence, we restrict the empirical section in this research to

multi-modal learning models that also use data representations for reconstruction and classification in a test scenario where only \mathbf{x}_1 is available.

3 Conditional Bi-Modal Discriminative model

Before we introduced our proposed CBMD model, we define some variables that are used throughout this section. Let \mathbf{x}_1 be the view of data available at the time a loan application is received. Common features in this view of data are: age, income, gender, geographical location, etc. Once an application is approved, customers generate new information generating view \mathbf{x}_2 . The sort of information in this view can be updated values for features in \mathbf{x}_1 , e.g. latest income, current age, latest marital status etc. Other kind of features in \mathbf{x}_2 can be repayment or purchase behavior. In the context of this research, we have access to class labels y , where $y = 0$ denotes if a customer repaid a loan, otherwise $y = 1$. Finally, we assume that there is a common latent representation $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$ with prior $p(\mathbf{z}|\mathbf{x}_1)$ and a private posterior representation $q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2)$ with prior $p(\mathbf{z}_2)$. Both latent representations contain high-level information of both views of data providing complementary information about the outcome of the loan.

3.1 Deriving the CBMD lower bound

We observe labeled bi-modal data $\{(\mathbf{x}_1^{(1)}, \mathbf{x}_2^{(1)}, y^{(1)}), \dots, (\mathbf{x}_1^{(N)}, \mathbf{x}_2^{(N)}, y^{(N)})\}$ that is generated at different point in times, where only view \mathbf{x}_1 is available at application time. Further, view \mathbf{x}_2 and class label y are generated after a loan application is granted.

We focus on learning a shared latent representation \mathbf{z} and a private representation $\mathbf{h}_{\mathbf{x}_2}$ that can be used for downstream classification and to generate \mathbf{x}_2 . For that purpose, we assume a conditional private distribution $p(\mathbf{z}|\mathbf{x}_1)$ for the view of data available at test time and an uninformative private distribution $p(\mathbf{h}_{\mathbf{x}_2})$ for the future credit data. Under this scenario, the joint generative process is given by

$$p(\mathbf{x}_2, \mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1) = p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})p(\mathbf{z}|\mathbf{x}_1)p(\mathbf{h}_{\mathbf{x}_2}), \quad (2)$$

where $p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})$ is the generative process for future credit scoring data. Note that the posterior distribution of the latent variable, which is exactly the shared latent representation that we want to learn,

$$p(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2) = \frac{p(\mathbf{x}_2, \mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1)}{\int \int p(\mathbf{x}_2, \mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1) d\mathbf{z} d\mathbf{h}_{\mathbf{x}_2}} \quad (3)$$

requires a marginal distribution that is not available in closed form. Therefore, we approximate the true posterior distribution $p(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$ in Equation 3 with the parametric model $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$.

Taking the log of the marginal distribution in Equation 3 we obtain the lower bound

$$\begin{aligned} \log p(\mathbf{x}_2|\mathbf{x}_1) &= \log \int \int p(\mathbf{x}_2, \mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1) d\mathbf{z} d\mathbf{h}_{\mathbf{x}_2} \\ &= \log \int \int q(\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1, \mathbf{x}_2) \frac{p(\mathbf{x}_2, \mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1)}{q(\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1, \mathbf{x}_2)} d\mathbf{z} d\mathbf{h}_{\mathbf{x}_2} \\ &= \log \mathbb{E}_{q(\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1, \mathbf{x}_2)} \left[\frac{p(\mathbf{x}_2, \mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1)}{q(\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1, \mathbf{x}_2)} \right] \\ &\geq \mathbb{E}_{q(\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1, \mathbf{x}_2)} \left[\log \frac{p(\mathbf{x}_2, \mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1)}{q(\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1, \mathbf{x}_2)} \right], \end{aligned} \quad (4)$$

where the inequality is a result of the concavity of log and Jensen's inequality. Equation 4 is the variational lower bound $\mathcal{L}(\mathbf{x}_2, \mathbf{x}_1)$ on the conditional log-likelihood $\log p(\mathbf{x}_2|\mathbf{x}_1)$, which in principle can be optimized using the stochastic variational gradient Bayes (SVGB) approach introduced in [20].

Expanding the lower bound in Equation 4 and assuming $q(\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_1, \mathbf{x}_2) = q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2)$, we get

that

$$\begin{aligned}
\mathcal{L}(\mathbf{x}_2, \mathbf{x}_1) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)}[\log p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}) + \log p(\mathbf{z}|\mathbf{x}_1) - \log q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)] \\
&\quad + \mathbb{E}_{q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2)}[\log p(\mathbf{h}_{\mathbf{x}_2}) - \log q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2)] \\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)}[\log p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})] - KL[q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)||p(\mathbf{z}|\mathbf{x}_1)] \\
&\quad - KL[q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2)||p(\mathbf{x}_2)].
\end{aligned} \tag{5}$$

While in some cases optimizing Equation 5 should be sufficient to do amortized inference and to reconstruct \mathbf{x}_2 correctly, it has been shown that this formulation of the lower bound has two main problems [19, 21]. First, it can fail to learn an amortized inference distribution $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$ that correctly approximates $p(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$. Second, the model can focus on reconstructing \mathbf{x}_2 ignoring the latent data representation \mathbf{z} , which implies that \mathbf{z} does not depend on \mathbf{x}_1 and \mathbf{x}_2 .

To solve the aforementioned challenges, we propose a new variational lower bound for bi-modality data in credit scoring. The main advantage of our proposed lower bound is that it maximizes mutual information between \mathbf{z} and \mathbf{x}_2 in a flexible objective function, helping to improve the reconstruction of view \mathbf{x}_2 . Further, our proposed lower bound in this research is better suited to learn amortized inference distributions $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$.

Note that the mutual information $I(\mathbf{x}_2, \mathbf{z})$ can be written as

$$\begin{aligned}
I(\mathbf{x}_2, \mathbf{z}) &= \mathbb{E}_{q(\mathbf{x}_2, \mathbf{z}|\mathbf{x}_1)} \left[\log \frac{q(\mathbf{x}_2, \mathbf{z}|\mathbf{x}_1)}{q(\mathbf{x}_2|\mathbf{x}_1)q(\mathbf{z}|\mathbf{x}_1)} \right] \\
&= \mathbb{E}_{q(\mathbf{x}_2, \mathbf{z}|\mathbf{x}_1)} \left[\log \frac{q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)q(\mathbf{x}_2|\mathbf{x}_1)}{q(\mathbf{x}_2|\mathbf{x}_1)q(\mathbf{z}|\mathbf{x}_1)} \right] \\
&= \mathbb{E}_{q(\mathbf{x}_2, \mathbf{z}|\mathbf{x}_1)}[\log q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2) - \log q(\mathbf{z}|\mathbf{x}_1) + \log p(\mathbf{z}|\mathbf{x}_1) - \log p(\mathbf{z}|\mathbf{x}_1)] \\
&= KL[q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)||p(\mathbf{z}|\mathbf{x}_1)] - KL[q(\mathbf{z}|\mathbf{x}_1)||p(\mathbf{z}|\mathbf{x}_1)],
\end{aligned} \tag{6}$$

hence adding the mutual information term $(1 - \omega)I(\mathbf{x}_2, \mathbf{z})$ to Equation 5 we obtain

$$\begin{aligned}
\mathcal{L}(\mathbf{x}_1, \mathbf{x}_2) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)}[\log p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})] - KL[q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)||p(\mathbf{z}|\mathbf{x}_1)] \\
&\quad - KL[q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2)||p(\mathbf{x}_2)] + (1 - \omega)[KL[q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)||p(\mathbf{z}|\mathbf{x}_1)] - KL[q(\mathbf{z}|\mathbf{x}_1)||p(\mathbf{z}|\mathbf{x}_1)]] \\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)}[\log p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})] - KL[q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2)||p(\mathbf{h}_{\mathbf{x}_2})] \\
&\quad - \omega KL[q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)||p(\mathbf{z}|\mathbf{x}_1)] + (1 - \omega)KL[q(\mathbf{z}|\mathbf{x}_1)||p(\mathbf{z}|\mathbf{x}_1)],
\end{aligned} \tag{7}$$

where $\omega \in [0, 1]$ is a weight hyperparameter. The last KL divergence $KL[q(\mathbf{z}|\mathbf{x}_1)||p(\mathbf{z}|\mathbf{x}_1)]$ can be replaced by any strict divergence term [19], e.g. maximum mean discrepancy divergence (MMD) [22]. We choose the squared MMD, which is

$$\text{MMD}[\mathcal{F}, p, q] = \mathbb{E}_{p(\mathbf{x}, \mathbf{x}')} [k(\mathbf{x}, \mathbf{x}')] - 2\mathbb{E}_{p(\mathbf{x}), q(\mathbf{z})} [k(\mathbf{x}, \mathbf{z})] + \mathbb{E}_{q(\mathbf{z}, \mathbf{z}')} [k(\mathbf{z}, \mathbf{z}')], \tag{8}$$

where \mathcal{F} be a unit ball in a universal reproducing kernel Hilbert space \mathcal{H} , p and q are Borel probability measures and $k(\cdot, \cdot)$ is a universal kernel. We use a Gaussian kernel in our proposed model to obtain the objective function

$$\begin{aligned}
\mathcal{L}(\mathbf{x}_1, \mathbf{x}_2) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)}[\log p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})] - KL[q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2)||p(\mathbf{h}_{\mathbf{x}_2})] \\
&\quad - \omega KL[q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)||p(\mathbf{z}|\mathbf{x}_1)] + (1 - \omega)\lambda \text{MMD}[q(\mathbf{z}|\mathbf{x}_1)||p(\mathbf{z}|\mathbf{x}_1)],
\end{aligned} \tag{9}$$

where λ counteracts the loss imbalance between \mathcal{X}_2 and \mathcal{Z} spaces [19]. Equation 9 give us more flexibility to reconstruct all features in view \mathbf{x}_2 utilizing the shared latent representation \mathbf{z} and to learn amortized inference distributions $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$.

However, we are interested in developing a model that, in addition to generate view \mathbf{x}_2 , can also classify the outcome of the loan. Further, given that we have a supervised data set, we want to use label information to learn shared latent representations. Hence, we add a classification loss $q(y|\mathbf{z}, \mathbf{h}_{\mathbf{x}_2})$ and

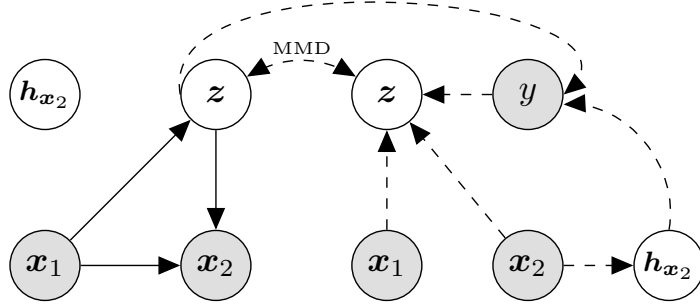


Figure 2: Plate notation for our proposed bi-modality discriminative model for credit scoring. The left side shows the generative model, where the prior distribution of \mathbf{z} is condition on the view \mathbf{x}_1 . The right side shows the inference model, where we explicitly optimize maximum mean discrepancy to minimize the information preference problem.

replace $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$ by $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, y)$ in Equation 9 to obtain the following final loss function in our proposed model

$$\mathcal{J} = -\mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, y) - \alpha \log q(y|\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}), \quad (10)$$

where α controls the importance of the classification loss in the objective function, and its plate notation is shown in Figure 2.

We minimize Equation 10 using SVGB and automatic differentiation routines in Theano [23]. Note that the reconstruction term of Equation 9 can be efficiently estimated using the *reparameterization trick* [20], the KL divergence term has a closed-form expression [20, 24], and the MMD divergence is approximated numerically by sampling from $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, y)$ and $p(\mathbf{z}|\mathbf{x}_1)$ for a given mini-batch of data as suggested by [22].

Finally, we assume the following density functions in our proposed CBMD model

$$\begin{aligned} p(\mathbf{h}_{\mathbf{x}_2}) &\sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\ p(\mathbf{z}|\mathbf{x}_1) &\sim \mathcal{N}(\mathbf{z}|\mathbf{x}_1; \boldsymbol{\mu} = f_{\boldsymbol{\theta}}(\mathbf{x}_1), \boldsymbol{\sigma}^2 = f_{\boldsymbol{\theta}}(\mathbf{x}_1)), \\ p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}) &\sim \mathcal{N}(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}; \boldsymbol{\mu} = f_{\boldsymbol{\theta}}(\mathbf{x}_1, \mathbf{z}), \boldsymbol{\sigma}^2 = f_{\boldsymbol{\theta}}(\mathbf{x}_1, \mathbf{z})), \\ q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, y) &\sim \mathcal{N}(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, y; \boldsymbol{\mu} = f_{\boldsymbol{\phi}}(\mathbf{x}_1, \mathbf{x}_2, y), \boldsymbol{\sigma}^2 = f_{\boldsymbol{\phi}}(\mathbf{x}_1, \mathbf{x}_2, y)), \\ q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2) &\sim \mathcal{N}(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2; \boldsymbol{\mu} = f_{\boldsymbol{\phi}}(\mathbf{x}_2), \boldsymbol{\sigma}^2 = f_{\boldsymbol{\phi}}(\mathbf{x}_2)), \\ q(y|\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}) &\sim \text{Bernoulli}(y|\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}; \boldsymbol{\pi}_{y|\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}} = f_{\boldsymbol{\phi}}(\mathbf{z}, \mathbf{h}_{\mathbf{x}_2})), \end{aligned} \quad (11)$$

where \mathcal{N} denotes the Gaussian distribution and $f(\cdot)$ is a multilayer perceptron (MLP) network [25]. That is, the density parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}^2$, and $\boldsymbol{\pi}_{y|\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}}$ are parametrized using neural networks with learnable parameters denoted by $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

The first density in Equation 11 is non-informative about the future credit data, while the second equation learns a latent representation (\mathbf{z}) based on the available information (\mathbf{x}_1) during the loan application process. In other words, $p(\mathbf{z}|\mathbf{x}_1)$ represents our prior beliefs about the shared representation \mathbf{z} and it is based on information available during the application process. The third density learns a data generating process to draw future credit scoring data (\mathbf{x}_2) based on available information (\mathbf{x}_1) and its latent representation (\mathbf{z}). The fourth density function learns a shared (posterior) latent representation for credit scoring data. To that end, $p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})$ uses credit scoring data generated before (\mathbf{x}_1) and after a loan application is approved (\mathbf{x}_2). To further improve the posterior latent representation, we add information about the class label (y). The fifth density learns a latent representation for future credit data. Finally, the last density function classifies the outcome of a loan y using latent representations (\mathbf{z} and $\mathbf{h}_{\mathbf{x}_2}$) for credit scoring data, and encourages latent representations to capture higher-level of abstractions that are useful for classification and to generate view \mathbf{x}_2 .

4 Experiments and Results

The motivation for the experiments is threefold. First, we compare the generative process of our proposed methodology with existing multi-modal learning models using two views. Second, we show how financial institutions can utilize the generative network in the CBMD model to generate future data. Finally, we compare the predictive power of the learned data representation for all models. All experiments assume that only view \mathbf{x}_1 is available during test time.

The models included in this section are CCA [4], KCCA [9], DCCA [15], and DCCAE [1], which all are based on canonical correlation. We also include in the comparison VCCA [2] and JMVAE [13] that are variational-based methods⁴. To allow a fair comparison to CBMD, all models are tested without pre-trained weights (as in [1]) or without adding generative adversarial networks [26] to further improve reconstructed values as in [13].

The only manipulation that we must do in the experiments where we compare the predictive power for all models, is to fix the variance parameter in the generative networks for VCCA and JMVAE. If not, downstream classification is poor as mentioned in [2] and confirmed by our empirical analysis. It is worth mentioning that, in our experiments, VCCA is more prone to poor classification if the variance parameters are learned during the optimization process.

4.1 Data description

We use two real and publicly available data sets in this section⁵. The first data set corresponds to customers at Banco Santander and it contains 200 (anonymized) numerical features for purchase prediction, i.e. which customer will make a future transaction regardless of the amount. A training and test data set are available, but we only use the training data set since the test data set has no label information. The training data set contains 200 000 observations and there are 20 098 customers that made a purchase, which corresponds to 10.05% of customers. Given that behavioral models have higher model performance than credit scoring models [27], we assume that features with high predictive power⁶ correspond to view \mathbf{x}_2 . Therefore, in the experiments conducted in Section 4.3.2, we select the top 50 features as view \mathbf{x}_2 , while the rest of the features correspond to view \mathbf{x}_1 . Given the number of features in this data set, we also tested all models under a more challenging scenario where view \mathbf{x}_1 and \mathbf{x}_2 contain 100 features each.

The second data set consists of peer-to-peer loan applications from January 2009 to December 2013 at Lending Club⁷. We only include accepted loans with 36-months maturity and some observations have been excluded using the same criteria as in [24, 29]. This data set contains 89 998 accepted applications, where 10 896 are defaulted loans, i.e. default rate is 12.11%. For this data set, we choose view \mathbf{x}_1 to be all common features in accepted and rejected applications, which are only 5 features. View \mathbf{x}_1 contains categorical variables that are transformed to one-hot-encoders, and the resulting input vector has 18 variables. View \mathbf{x}_2 contains 72 features, of these we select features that were both continuous and with empirical distributions resembling Gaussian densities. Ending up with 8 features used in the experiments conducted in Section 4.3.2. This choice is driven by the fact that view \mathbf{x}_1 has only 5 original features. Details about data views in the Lending Club data set are shown in Section A in the appendix.

4.2 Model training and testing

We use MLP networks with softplus activation functions in all hidden layers to parameterized $\boldsymbol{\mu}$, $\boldsymbol{\sigma}^2$ and $\pi_{y|z, \mathbf{h}_{\mathbf{x}_2}}$ in Equation 11. For the output layers parameterizing $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$, we use linear activation functions, while for the classifier we use a softmax activation function. The minimization of the loss function is done using the Adam optimizer [30] with learning rate equal to 1e-4. The final architectures that we used in our proposed model, as well as all architectures used in the grid-search to tune the MLPs,

⁴In our experiments, we use the implementations for CCA, KCCA, DCCA, DCCAE, and VCCA at <https://ttic.uchicago.edu/~wwang5/>. While, results for JMVAE are based on our own implementation.

⁵Banco Santander data set: <https://www.kaggle.com/c/santander-customer-transaction-prediction/data>. Lending Club data set: https://github.com/nateGeorge/preprocess_lending_club_data

⁶We use the method introduced in [28] to estimate feature importance.

⁷Lending Club is the world's largest peer-to-peer lending company and it was the first peer-to-peer lender to register its offerings as securities with the Securities and Exchange Commission in the U.S., and to offer loan trading on a secondary market.

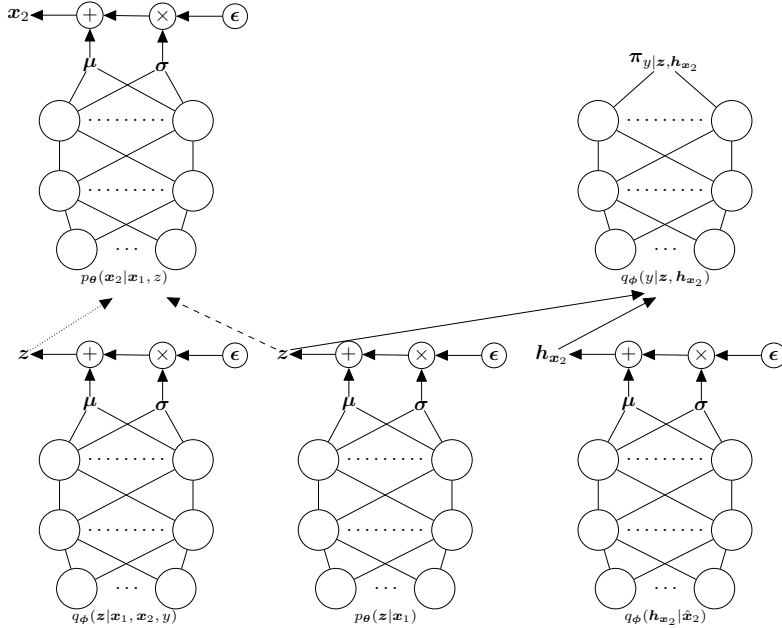


Figure 3: Forward propagation in our proposed model. The dotted arrow indicates a forward pass during training, which is replaced by the dashed arrow at test time. Solid arrows depict a common forward propagation during training and test.

are shown in Table C1 in the appendix. All CCA-based and variational-based models are trained with similar architectures to CBMD for a fair comparison. Further, for DCCAE we tune the λ parameter by grid search as suggested in [1]. Similarly, we tune the α and variance parameters by grid search in JMVAE and VCCA respectively. Finally, both data sets are scaled between 0 and 1 for better training stability.

During training we have a supervised data set containing both views \mathbf{x}_1 and \mathbf{x}_2 , as well as the class label y . At test time we assume that only view \mathbf{x}_1 is available. Therefore, at training time we draw samples from $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, y)$ to reconstruct view \mathbf{x}_2 using the generative process $p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})$ in our proposed CBMD model. While at test time, we need to rely on the conditional prior distribution $p(\mathbf{z}|\mathbf{x}_1)$ to draw \mathbf{z} . Then, we can use that \mathbf{z} representation to generate \mathbf{x}_2 using $p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})$. In other words, we generate future credit data (\mathbf{x}_2) based on current information about the loan application (\mathbf{x}_1) and based on the prior distribution ($p(\mathbf{z}|\mathbf{x}_1)$) of the shared latent representation. Note that the conditional prior distribution in our proposed model is more informative than the classical choice $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

We observed in our experiments that, during training, generating \mathbf{z} from $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, y)$ leads to unstable classification of y . Therefore, we use the prior distribution $p(\mathbf{z}|\mathbf{x}_1)$ to generate \mathbf{z} and to classify the class label using $q(y|\mathbf{z}, \mathbf{h}_{\mathbf{x}_2})$ during training and at test time. We hypothesize that the prior distribution reproduce better the test scenario compared to the posterior distribution⁸. Likewise, we generate $\mathbf{h}_{\mathbf{x}_2}$ from $q(\mathbf{z}|\hat{\mathbf{x}}_2)$ at training and test time, drawing $\hat{\mathbf{x}}_2$ from the generative process $p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})$. For clarity, Figure 3 shows the forward propagation during training and test time in our proposed methodology.

Inspired by JMVAE, we tried to bring together the private latent representation $q(\mathbf{h}_{\mathbf{x}_2}|\mathbf{x}_2)$ and the shared representation $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, y)$, but using MMD as divergence measure and the sampling approach described at the end of Section 3.1. While we do not see a clear benefit in the generative process of the model or in the predictive power of it, we see faster model convergence.

⁸In [2] latent representations conditioned on the available view at test time also give better performance.

Feature Name	True \mathbf{x}_2		CBMD $\hat{\mathbf{x}}_2$		JMVAE $\hat{\mathbf{x}}_2$		DCCAE $\hat{\mathbf{x}}_2$	
	Average	Std. deviation	Average	Std. deviation	Average	Std. deviation	Average	Std. deviation
feature 1	0.17728	0.11037	0.18052	0.03246	0.18232	1.49011e-08	0.17613	0.01057
feature 2	0.68461	0.15871	0.71402	0.14718	0.68072	1.19209e-07	0.70614	0.05208
feature 3	0.74140	0.14333	0.75308	0.13515	0.74729	5.96046e-08	0.74132	0.02919
feature 4	0.19370	0.08624	0.19388	0.03549	0.19306	1.48926e-08	0.16939	0.01109
feature 5	0.46439	0.20650	0.41057	0.21793	0.46315	1.98431e-08	0.39902	0.09241
feature 6	0.23878	0.12198	0.24361	0.03986	0.23649	1.49216e-08	0.21474	0.02407
feature 7	0.20347	0.15674	0.20166	0.12713	0.20162	6.12372e-09	0.21303	0.03387
feature 8	0.22988	0.19139	0.23108	0.16712	0.22704	1.49011e-08	0.23958	0.04155

Table 2: Average and standard deviation values for the true and reconstructed \mathbf{x}_2 features in the test data set using CBMD, JMVAE, and DCCAE. All models are able to capture the empirical mean for each feature. However, JMVAE and DCCAE fail at capturing the variation across different customers.

4.3 Experimental design

We use 70% of the data to learn a common data representation for both data modalities, which is further used to generate the view \mathbf{x}_2 and to train a multilayer perceptron (MLP) classifier, except for CBMD that trains a classifier at the same time as it learns shared data representations and generates \mathbf{x}_2 . For this 70% of the data, we down-sample the majority class ($y = 0$) to balance both class labels. Further, we use 25% of the data to test the predictive power of the classifier for all models and the quality of the reconstructed view \mathbf{x}_2 using JMVAE and CBMD. The test data set preserves the original balance between the two classes. Finally, we use the remaining 5% of the data to calibrate class probabilities using the beta calibration approach [31]. For all experiments we do a 10-cross-validation.

4.3.1 Generating view \mathbf{x}_2

Of all models tested in this research, only DCCAE, JMVAE and CBMD are able to generate view \mathbf{x}_2 based on the available view \mathbf{x}_1 during test time. Models with autoencoder-like architectures, e.g. VCCA or DMDGM, learn to reconstruct $\hat{\mathbf{x}}_2$ based on \mathbf{x}_2 and therefore cannot be used under the test scenario in this research. Note that both JMVAE and CBMD estimate a posterior distribution for view \mathbf{x}_2 . Hence, using a quadratic loss function $\mathcal{L} = (\mathbf{x}_2 - \hat{\mathbf{x}}_2)^2$ we obtain a point estimate $\hat{\mathbf{x}}_2^* = \arg \min \mathbb{E}[\mathcal{L} = (\mathbf{x}_2 - \hat{\mathbf{x}}_2)^2 | \mathbf{x}_1, \mathbf{z}]$. Taking the first derivative of the expectation with respect to $\hat{\mathbf{x}}_2$ and forcing the result equal to 0, we obtain $\hat{\mathbf{x}}_2^* = \mathbb{E}[\mathbf{x}_2 | \mathbf{x}_1, \mathbf{z}]$. This expectation is exactly what JMVAE and CBMD parametrize with MLPs (see Equation 11), and it is our choice for a point estimate in this section. On the other hand, DCCAE utilizes deterministic neural networks to generate \mathbf{x}_2 , hence their output is a single point estimate. Note that to draw \mathbf{x}_2 values with DCCAE, we use latent representations generated with \mathbf{x}_1 .

Table 2 shows true and generated average and standard deviation values for all features in view \mathbf{x}_2 in the test data set⁹. Interestingly, all models estimate highly accurate the support of the empirical distribution for each feature. However, JMVAE clearly fails at recognizing the dispersion in each feature. This results is most likely due to the information preference problem, meaning that $p(\mathbf{x}_2 | \mathbf{z})$ is basically the same for all \mathbf{z} [19]. Similarly, DCCAE does not match the empirical standard deviation for all features. On the other hand, our proposed CBMD model matches the variation for all features in view \mathbf{x}_2 .

Figure 4 shows histograms for all true (solid curve) and generated features in view \mathbf{x}_2 for the Lending Club and using the generative model in CBMD, JMVAE, and DCCAE depicted by the dashed curve, dotted curve and dotted vertical line respectively. It is interesting to see that CBMD centers it mass in the main mode of complex densities such as feature 2 and 3. Further, skewed densities like feature 5, 7, and 8 are also reconstructed correctly. On the other hand, both JMVAE and DCCAE fail to capture the dispersion across different customers.

To further analyze the quality of the drawn \mathbf{x}_2 variables, we create 5 equally-sized groups with different risk profiles based on posterior class probabilities estimated with $q(y | \mathbf{z}, \mathbf{h}_{\mathbf{x}_2})$. Group A has the lowest class posterior probability, while group E has the highest class posterior probability. Table 3 shows these 5 groups, together with true and generated average values for all features in the test data set. True values are shown in the first row for each group, while in the second and third row we generate \mathbf{x}_2 using the

⁹Generated values for features in view \mathbf{x}_2 , in Figure 4 and Table 2, are expectations from the conditional posterior distribution $p(\mathbf{x}_2 | \mathbf{x}_1, \mathbf{z})$ for CBMD and JMVAE, while for DCCAE are the outputs of a neural network.

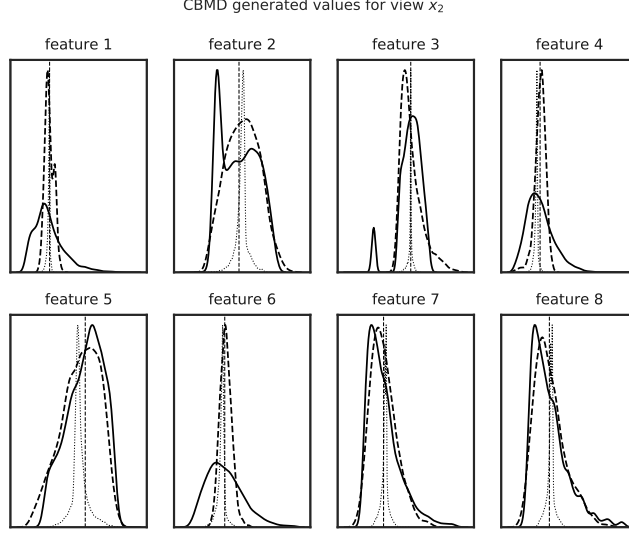


Figure 4: Solid curves show the true empirical distributions for all features in view \mathbf{x}_2 in the Lending Club test data set. While the dashed and dotted curves show the empirical distributions for the generated features using CBMD and DCCAE respectively. The dotted vertical line shows generated values using JMVAE.

Group & model	feature 1	feature 2	feature 3	feature 4	feature 5	feature 6	feature 7	feature 8	rmse	
A	true \mathbf{x}_2	0.1992	0.6304	0.8565	0.1531	0.2349	0.2222	0.2447	0.3129	
	$\hat{\mathbf{x}}_2(\omega^*)$	0.1904	0.7332	0.8291	0.1559	0.2171	0.2119	0.2352	0.3096	0.0386
	$\hat{\mathbf{x}}_2(\omega = 1)$	0.2134	0.6481	0.7885	0.1661	0.2600	0.2070	0.2398	0.2976	0.0284
	$\hat{\mathbf{x}}_2 CBM(\omega^*)$	0.1938	0.6600	0.7845	0.1562	0.2466	0.2039	0.2300	0.2878	0.0304
B	true \mathbf{x}_2	0.1852	0.6242	0.8199	0.1602	0.4077	0.2142	0.2627	0.3239	
	$\hat{\mathbf{x}}_2(\omega^*)$	0.1830	0.6729	0.7515	0.1681	0.3627	0.2150	0.2466	0.3067	0.0348
	$\hat{\mathbf{x}}_2(\omega = 1)$	0.1997	0.6317	0.7260	0.1804	0.4090	0.2060	0.2465	0.2880	0.0373
	$\hat{\mathbf{x}}_2 CBM(\omega^*)$	0.1754	0.6399	0.7108	0.1611	0.4011	0.19	0.2411	0.2767	0.0441
C	true \mathbf{x}_2	0.1826	0.6317	0.7973	0.1636	0.4764	0.2134	0.2674	0.3205	
	$\hat{\mathbf{x}}_2(\omega^*)$	0.1771	0.6342	0.7191	0.1743	0.4287	0.2147	0.2469	0.2952	0.0347
	$\hat{\mathbf{x}}_2(\omega = 1)$	0.1922	0.6274	0.6966	0.1859	0.4780	0.2047	0.2368	0.2669	0.0428
	$\hat{\mathbf{x}}_2 CBM(\omega^*)$	0.1659	0.6299	0.6808	0.1619	0.4706	0.1816	0.2407	0.263	0.0487
D	true \mathbf{x}_2	0.1768	0.6362	0.7834	0.1694	0.5044	0.2140	0.2576	0.3016	
	$\hat{\mathbf{x}}_2(\omega^*)$	0.1682	0.6068	0.6967	0.1788	0.4766	0.2130	0.2392	0.2762	0.0359
	$\hat{\mathbf{x}}_2(\omega = 1)$	0.1855	0.6250	0.6733	0.1914	0.5293	0.2048	0.2264	0.2475	0.0467
	$\hat{\mathbf{x}}_2 CBM(\omega^*)$	0.1581	0.6241	0.6608	0.1628	0.5208	0.1753	0.232	0.2441	0.0516
E	true \mathbf{x}_2	0.1663	0.6396	0.7646	0.1805	0.5322	0.2240	0.2318	0.2638	
	$\hat{\mathbf{x}}_2(\omega^*)$	0.1585	0.5829	0.6766	0.1890	0.5278	0.2212	0.2177	0.2406	0.0385
	$\hat{\mathbf{x}}_2(\omega = 1)$	0.1790	0.6240	0.6477	0.2071	0.5889	0.2151	0.2108	0.2203	0.0505
	$\hat{\mathbf{x}}_2 CBM(\omega^*)$	0.1582	0.6229	0.6462	0.1728	0.5735	0.1825	0.2086	0.2105	0.0515
highest π	0.1310	0.5704	0.6526	0.2138	0.5884	0.2518	0.1280	0.1260		
lowest π	0.1798	0.6656	0.9791	0.1146	0.0122	0.1547	0.0348	0.0471		

Table 3: We use estimated class probabilities using CBMD to create 5 equally-sized groups (A-E). Further, we show true \mathbf{x}_2 average values and generated $\hat{\mathbf{x}}_2$ average values using our proposed lower bound and the classical lower bound denoted by ω^* and $\omega = 1$ respectively. The last column shows root mean squared error.

optimal ω^* value and $\omega = 1$ in our proposed lower bound (Equation 9). The latter corresponds to the classical lower bound in generative models. We can see that in all groups, but A, the optimal ω^* value generates relatively more accurate features as suggested by the root mean squared error (rmse), and for some features in some groups the generated \mathbf{x}_2 values are highly accurate. Note that in group A the high rmse for ω^* is mainly driven by feature 2. The last row in each group, $\hat{\mathbf{x}}_2 CBM(\omega^*)$, shows generated features with our proposed model at the optimal ω^* value, but without the discriminative model. We can observe that the classifier in CBMD encourages the generative model to draw \mathbf{x}_2 accurately.

Table 3 shows from another perspective why models should not use fixed variance parameters in the

generative process, as is the case for JMVAE and VCCA. Such a practice impedes a model to capture the variability among customers. Similarly, using deterministic neural networks to generate \mathbf{x}_2 , as in DCCAE, makes it more challenging to capture the variation across customers.

The 5 groups that we created are shown in the left panel of Figure 5, which are two-dimensional t-sne [32] components of the latent space $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x}_1)$ for the test data set. Note that the 5 groups that we have defined are clustered in well-defined structures with minimal overlap. Furthermore, the right panel of Figure 5 shows a colormap of the same t-sne components where the color is given by the posterior class probability estimated using $q(y|\mathbf{z}, \mathbf{h}_{\mathbf{x}_2})$. Note that there is a smooth transition across the two dimensions. This is a characteristic of the learned latent space with deep generative models, which preserves the spatial coherence of creditworthiness [33].

Anchor customers to generate future credit data

Financial institutions use repayment or purchase behavior data for launching new products, cross-selling or marketing campaigns. After these types of data sets are generated, bank analysts collect behavior data and analyze it to support the aforementioned activities. We introduce an alternative approach where we use the generative process in CBMD to draw future credit data \mathbf{x}_2 before customers generate it. To that end, we define *anchor customers*, which serve as point of reference to generate \mathbf{x}_2 .

Suppose a bank wants to launch a new private loan for high-risk customers. In this case, we define as anchor customer the client in group E with the highest posterior class probability. This customer is depicted in the right panel of Figure 5 by a red scatter point in the zoom box at the top-left corner. We use view \mathbf{x}_1 for the anchor customer to draw its latent representation, i.e. $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x}_1)$. Further, we use the generated latent representation \mathbf{z} together with \mathbf{x}_1 to generate future credit data for the anchor customer using the generative process $\mathbf{x}_2 \sim p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})$. Note that deep generative models, such as CBMD, assume per-point latent variables and density functions, hence the Gaussian generative process for the i 'th anchor customer is given by $\mathcal{N}(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)})$. At the bottom of Table 3, we show average values for all features in view \mathbf{x}_2 for the anchor customer with highest posterior class probability. We can see that the average high-risk customer will have low values for risk scores (feature 3), just as expected. Similarly, the bottom row in Table 3 shows average values for an anchor customer with the lowest class probability, which is depicted with a yellow scatter in the zoom box at the top-right corner of Figure 5.

Anchor customers should be selected according with how future data \mathbf{x}_2 will be used. In our above example, we generated future data for a given segment A-E using one anchor customer and calculated average values for all variables. Another possibility is to select a set of customers within a given segment of interest. In that case, we can use the expectation of $\mathbf{x}_2^{(i)}$ for the i 'th customer as a point estimate, which is simply the parameter $\boldsymbol{\mu}^{(i)}$ in $\mathcal{N}(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)})$.

4.3.2 Classification using data representations

Even though the main motivation to include a classifier model in our proposed methodology is to generate accurate features in view \mathbf{x}_2 , in Table 4 we compare classification performance for all benchmark models in terms of AUC, GINI, and H-measure to provide different angles from which to examine the classification performance¹⁰. Given that the Santander Bank data set has 200 input features, we train all models in two different scenarios. In the first scenario view \mathbf{x}_1 has 150 features and view \mathbf{x}_2 has 50 features, while in the second scenario both views have 100 features. Model M- \mathbf{x}_1 provides a baseline for the traditional credit scoring approach where only view \mathbf{x}_1 is used.

Our experiments show that on average CCA-based models perform better for credit scoring than VCCA and JMVAE. This result has been explained in [1] and it happens when the views in the data sets are uncorrelated. Remember that the objective function in CCA-based models maximize canonical correlation. Further, it is interesting to see that both CCA and KCCA have slightly higher performance than the base model for the Lending Club data set. On the other hand, DCCA, DCCAE, and VCCA have the lowest model performance for the Lending Club data set. However, DCCA achieves on-pair

¹⁰In credit scoring models, score-specific performance metrics, e.g. recall or precision, are not common to use since banks use the probabilities $\pi_{y|\mathbf{z}, \mathbf{h}_{\mathbf{x}_2}}$ to rank customers.

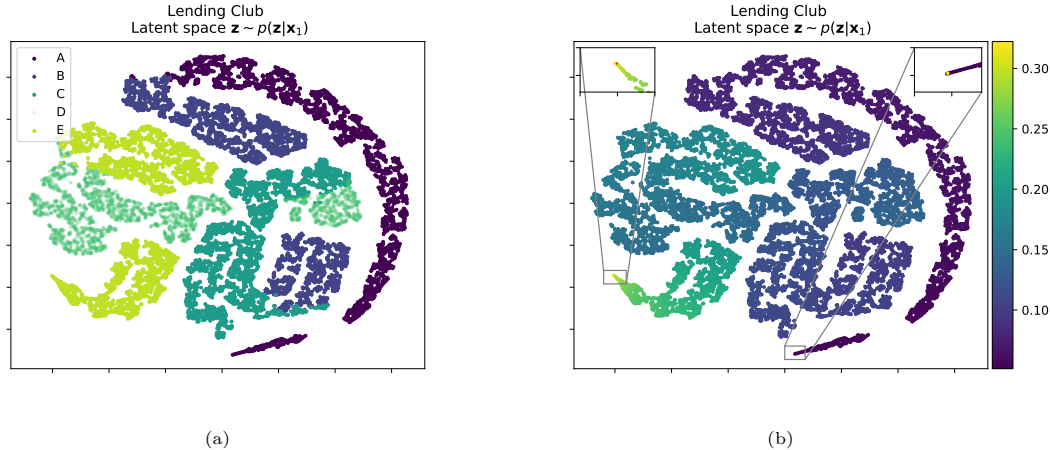


Figure 5: Best viewed in color. Two-dimensional t-sne components of the latent space $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x}_1)$ for the Lending Club test data set. The left panel shows the 5 groups that we created based on average values for posterior class probabilities, while the right panel shows a colormap of the same t-sne components where the color is given by the posterior class probability estimated by the CBMD model. Note the smooth transition across the two dimension.

model performance compared to the baseline model for the Santander data set with 50 features in view \mathbf{x}_2 . It is important to note that [1] used pre-trained weights for DCCAE. We do not follow such practices to allow a fair comparison with CBMD. Hence, it might be possible to improve DCCAE performance by doing so.

Our proposed CBMD model performs slightly better than the baseline in all 3 experiments. Similarly, we also observe that CBMD achieves higher performance than most models. The only model with higher performance than CBMD is KCCA, which achieves the highest performance for the Santander data set with 150 features in view \mathbf{x}_1 . However, when we reduce the number of features in view \mathbf{x}_1 to 100, CBMD has a marginal improvement in performance compared to both KCCA and the baseline model. The fact that none of the models in the benchmark analysis are able to achieve a significant improvement over the baseline, may suggest that the views of data are not conditional independent given the data representations, which is an assumption in downstream classification tasks with multi-modal learning models [2].

It is important to mention that CBMD does not need to use fixed values for the variance parameters in the generative network $p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})$, as opposed to VCCA and JMVAE, since CBMD is able to learn these parameters during the optimization procedure. It is also worth mentioning that we use the same model architecture and hyperparameter values in the experiment where both \mathbf{x}_1 and \mathbf{x}_2 have 100 features as in the experiments where \mathbf{x}_1 has 150 features. If we tune the ω parameter in CBMD for the experiments with 100 features in both views, we obtain an average AUC of 0.63414.

The motivation for the lower bound introduced in this research is to better fit the data distribution $p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z})$ and to use efficiently the latent representation \mathbf{z} . However, we also observe a small impact on classification performance. Figure 6 shows average AUC as a function of ω in Equation 9. A value $\omega = 1$ corresponds to the classical lower bound (Equation 4), while $\omega = 0$ maximizes mutual information $I_{q(\mathbf{x}_2, \mathbf{z}|\mathbf{x}_1)}(\mathbf{x}_2, \mathbf{z})$. Finally, ω values between 0 and 1 maximize our proposed lower bound in Equation 9. We can see that our proposed objective function achieves higher AUC compared to the classical lower bound, both for the Santander (solid lines) and Lending Club (dashed line) data sets.

5 Conclusion

In this research, we develop a novel conditional bi-modal discriminative (CBMD) model that is able to generate the view of data \mathbf{x}_2 conditioned on view \mathbf{x}_1 , which is our best source of information about future customer behavior. Further, CBMD can classify the outcome of loans using private latent representations. The generative process in our proposed model keeps the relationship between the views \mathbf{x}_1 and \mathbf{x}_2 for

Model name	Lending Club ($\mathbf{x}_1 : 18 \ \mathbf{x}_2 : 8$)			Santander Bank ($\mathbf{x}_1 : 150 \ \mathbf{x}_2 : 50$)			Santander Bank ($\mathbf{x}_1 : 100 \ \mathbf{x}_2 : 100$)		
	AUC	GINI	H-measure	AUC	GINI	H-measure	AUC	GINI	H-measure
M- \mathbf{x}_1	0.61986	0.23972	0.04720	0.73844	0.47688	0.18509	0.63245	0.26490	0.06035
CCA	0.62004	0.24009	0.04733	0.73299	0.46597	0.17779	0.63141	0.26282	0.05919
KCCA	0.61996	0.23993	0.04684	0.74495	0.48989	0.19382	0.63152	0.26303	0.05822
DCCA	0.60783	0.21566	0.03787	0.74002	0.48004	0.18740	0.62420	0.24841	0.05246
DCCAE	0.60798	0.21597	0.03797	0.73756	0.47511	0.18273	0.62282	0.24564	0.05169
VCCA	0.60909	0.21818	0.04062	0.73621	0.47243	0.18211	0.63060	0.26120	0.05801
JMVAE	0.61920	0.23840	0.04654	0.68974	0.37948	0.11839	0.59354	0.18708	0.03200
CBMD	0.62049	0.24098	0.04764	0.74014	0.48028	0.18764	0.63395	0.26790	0.06146

Table 4: The first model M- \mathbf{x}_1 uses only view \mathbf{x}_1 to classify y with a MLP model. All CCA-based models, VCCA and JMVAE use shared data representations to classify y in a two-stage approach. On the other hand, our proposed CBMD model classifies labels in a unified framework. Average AUC, GINI, and H-measure are shown in the above table. The last two rows show the average normalized root mean square error for the reconstructed view \mathbf{x}_2 using CBMD and JMVAE.

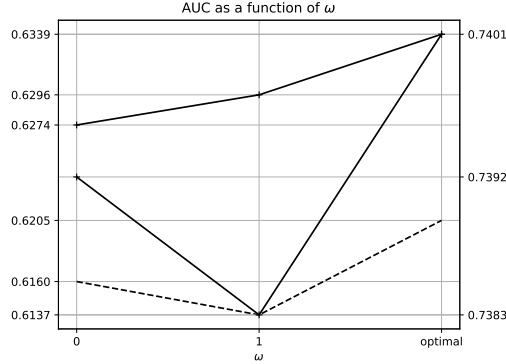


Figure 6: Average AUC performance for the Santander (solid lines) and Lending Club (dashed line) data set. For $\omega = 1$ CBMD optimizes the classical lower bound in generative models, while $\omega = 0$ optimizes mutual information between \mathbf{z} and \mathbf{x}_2 . For ω values in between, CBMD optimizes the lower bound introduced in this paper. Note that the *optimal* ω value, 0.8 for Lending club and 0.05 for Santander data set, achieves AUC.

each customer and it is useful in scenarios where only one view is available at test time. Even though we show in this research how to generate view \mathbf{x}_2 in the context of credit scoring, our proposed methodology generalizes to other research fields. Similarly, extending CBMD to multiple measurement-modalities is possible by concatenating multiple views in the conditional prior distribution.

Our proposed CBMD model optimizes a novel objective function that maximizes mutual information between latent data representation \mathbf{z} and view \mathbf{x}_2 . This loss function learns an amortized inference distribution for $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2, y)$, which contributes to an efficient generative model for view \mathbf{x}_2 . Therefore, we do not need to fix the variance parameters in the generative process as VCCA and JMVAE do. To further improve the generative process, we introduce a classifier model that encourages the generative model to draw \mathbf{x}_2 accurately. Our empirical results suggest that including the classification loss and the mutual information term in the objective function effectively improve the accuracy of generated features in view \mathbf{x}_2 . Finally, our proposed objective function also achieves higher AUC compared to the classical lower bound in generative models.

To the best of our knowledge, this research presents the first credit scoring model based on bi-modal learning able to generate future credit data \mathbf{x}_2 and therefore it opens an interesting avenue for future research. Likewise, our proposed methodology offers new possibilities on how banks could implement the use of generated \mathbf{x}_2 values in their activities that involve the prediction of customer’s credit behavior.

Acknowledgments

The authors would like to thank Santander Consumer Bank Nordics and the Research Council of Norway [grant number 260205 and 276428] for financial support for this research.

6 Appendix

A Data sets

We select view \mathbf{x}_1 for the Lending Club data set using the common features for accepted and rejected applications, since this is the case in real loan application process. These features are loan amount, Fico scores, address state, debt to income ratio, and employment length. Further, we follow the practice as in [24, 29] and create 4 different groups using address state, which are further transformed to one-hot encoders. Similarly, given that employment length has 11 different categories, we also convert it to one-hot encoders. Therefore, view \mathbf{x}_1 has 18 features.

From the remaining 72 features for accepted applications, we select those variables whose empirical distribution resembles a Gaussian density. Remember that our proposed CBMD model assumes a multivariate Gaussian distribution for view \mathbf{x}_2 . Given that we only have 5 original features for view \mathbf{x}_1 , we select 8 features for view \mathbf{x}_2 and can be found in Table B1.

Table B1: Lending Club views of data

Variable name	
View \mathbf{x}_1	Loan amount Fico score Address state Debt to income ratio Employment length
View \mathbf{x}_2	(feature 1) days_earliest_cr_line (feature 2) days_last_pymnt_d (feature 3) last_risk_score (feature 4) open_acc (feature 5) revolv_util (feature 6) total_acc (feature 7) total_pymnt (feature 8) total_rec_prncp

B Model architectures

Table C1 shows all architectures tested for hyperparameter optimization for our proposed CBMD model, JMVAE, VCCA, and DMDGM model. We use the notation for CBMD to specify the different MLP networks, but all models have a similar network just with different inputs. For example, JMVAE uses $q(\mathbf{z}|\mathbf{x}_1, \mathbf{x}_2)$ as inference network. For models with two inference or generative networks, e.g. JMVAE has $p(x_1|\mathbf{z})$ and $p(x_2|\mathbf{z})$, we use the same architecture for both networks.

Lending Club	
MLP Network	Number of hidden layers and dimensions
$p(\mathbf{x}_2 \mathbf{x}_1, \mathbf{z})$	[50], [60], [70], [80], [100], [120], [150], [200] [50 50], [60 60], [70 70], [80 80], [100 100]***, [120 120], [150 150]**, [200 200]*, [50 50 50], [60 60 60], [70 70 70], [80 80 80], [100 100 100], [120 120 120], [150 150 150], [200 200 200]
$p(\mathbf{z} \mathbf{x}_1)$	[20], [30], [40], [50], [60], [70], [80], [100]*, [120], [150]
$q(\mathbf{z} \mathbf{x}_1, \mathbf{x}_2, y)$	[40]***, [50], [60], [70], [80], [100]***, [120], [150], [200] [50 50], [60 60], [70 70], [80 80], [100 100], [120 120], [150 150], [200 200], [50 50 50], [60 60 60], [70 70 70], [80 80 80], [100 100 100], [120 120 120], [150 150 150], [200 200 200]
$q(y \mathbf{z})$	[50], [60], [70],[80], [100], [120], [150], [50 50], [60 60], [70 70],[80 80], [100 100]*, [120 120], [150 150]
Parameter/hyperparameter	Value
\mathbf{z} dimension	10, 20, 30, 40, 50*~***~****~*****, 70, 90, 110, 130, 150, 170
ω	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8*, 0.9, 1
λ	1000, 2000, 3000, 4000*
α	1, 5, 10, 15, 20*~*****, 30, 40, 50
Santander Bank	
$p(\mathbf{x}_2 \mathbf{x}_1, \mathbf{z})$	[100 100], [200 200], [300 300], [500 500], [700 700], [900 900], [100 100 100],[200 200 200], [300 300 300],[500 500 500], [700 700 700], [900 900 900]*
$p(\mathbf{z} \mathbf{x}_1)$	[100], [200], [300]*, [400], [500]
$q(\mathbf{z} \mathbf{x}_1, \mathbf{x}_2, y)$	[100 100], [200 200], [300 300], [500 500], [700 700], [900 900], [100 100 100], [200 200 200], [300 300 300],[500 500 500], [700 700 700]*, [900 900 900]
$q(y \mathbf{z})$	[100], [200], [300], [400], [500], [700]*, [900]
Parameter/hyperparameter	Value
\mathbf{z} dimension	100, 200, 300, 400*~***~****~*****, 500, 600, 700, 800, 900
ω	0, 0.1*, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
λ	1000, 2000, 3000, 4000*
α	1, 5, 10, 15, 20*~***, 30, 40, 50

Table C1: Grid for hyperparameter optimization for CBMD, JMVAE, and VCCA. The numbers within brackets specify the number of neurons in each hidden layers, i.e. [10 10] means two hidden layers with 10 neurons each. Superscripts *, **, *** show the final architecture for CBMD, JMVAE, and VCCA, respectively.

References

- [1] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International Conference on Machine Learning*, pages 1083–1092, 2015.
- [2] Weiran Wang, Xinchun Yan, Honglak Lee, and Karen Livescu. Deep variational canonical correlation analysis. *arXiv preprint arXiv:1610.03454*, 2016.
- [3] Fang Du, Jianshe Zhang, Junying Hu, and Rongrong Fei. Discriminative multi-modal deep generative models. *Knowledge-Based Systems*, 173:74–82, 2019.
- [4] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.
- [5] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.
- [6] Shotaro Akaho. A kernel method for canonical correlation analysis. *arXiv preprint cs/0609071*, 2006.
- [7] Pei Ling Lai and Colin Fyfe. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(05):365–377, 2000.
- [8] Thomas Melzer, Michael Reiter, and Horst Bischof. Nonlinear feature extraction using generalized canonical correlation analysis. In *International Conference on Artificial Neural Networks*, pages 353–360. Springer, 2001.
- [9] David Lopez-Paz, Suvrit Sra, Alex Smola, Zoubin Ghahramani, and Bernhard Schölkopf. Randomized nonlinear component analysis. In *International conference on machine learning*, pages 1359–1367, 2014.
- [10] Francis R Bach and Michael I Jordan. A probabilistic interpretation of canonical correlation analysis. *Technical report 688, Department of Statistics UC, Berkeley*, 2005.
- [11] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.
- [12] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.
- [13] Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. Joint multimodal learning with deep generative models. *arXiv preprint arXiv:1611.01891*, 2016.
- [14] Mike Wu and Noah Goodman. Multimodal generative models for scalable weakly-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5575–5585, 2018.
- [15] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255, 2013.
- [16] Changde Du, Changying Du, Hao Wang, Jinpeng Li, Wei-Long Zheng, Bao-Liang Lu, and Huiguang He. Semi-supervised deep generative modelling of incomplete multi-modality emotional data. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 108–116. ACM, 2018.
- [17] Ramakrishna Vedantam, Ian Fischer, Jonathan Huang, and Kevin Murphy. Generative models of visually grounded imagination. *arXiv preprint arXiv:1705.10762*, 2017.
- [18] Kihyuk Sohn, Wenling Shang, and Honglak Lee. Improved multimodal deep learning with variation of information. In *Advances in neural information processing systems*, pages 2141–2149, 2014.
- [19] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.
- [20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

- [21] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.
- [22] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2007.
- [23] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.
- [24] Rogelio A Mancisidor, Michael Kampffmeyer, Kjersti Aas, and Robert Jenssen. Deep generative models for reject inference in credit scoring. *Knowledge-Based Systems*, page 105758, 2020.
- [25] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- [26] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [27] Raymond Anderson. *The credit scoring toolkit: theory and practice for retail credit risk management and decision automation*. Oxford University Press, 2007.
- [28] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.
- [29] Zhiyong Li, Ye Tian, Ke Li, Fanyin Zhou, and Wei Yang. Reject inference in credit scoring using semi-supervised support vector machines. *Expert Systems with Applications*, 74:105–114, 2017.
- [30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [31] Meelis Kull, Telmo Silva Filho, and Peter Flach. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Artificial Intelligence and Statistics*, pages 623–631, 2017.
- [32] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.
- [33] Rogelio A Mancisidor, Michael Kampffmeyer, Kjersti Aas, and Robert Jenssen. Learning latent representations of bank customers with the variational autoencoder. *Expert Systems with Applications*, page 114020, 2020.

Appendix A

Multilayer Perceptron Model and the Backpropagation algorithm

Let us adopt the notation used in Theodoridis and Koutroumbas (2011) to introduce multilayer perceptron (MLP) neural networks and the backpropagation algorithm (Rumelhart et al., 1988).

An MLP is formed by L fully connected layers and in each layer there are k_r neurons, for $r = 1, 2, \dots, L$. Further, let the weight vector at the j 'th neuron in the r 'th layer be given by $\mathbf{w}_j^r = (w_{j0}^r, w_{j1}^r, \dots, w_{jk_{r-1}}^r)^T$, where w_{j0}^r is the bias term and $y_0^r = 1$. The dimension of \mathbf{w}_j^r is $k_{r-1} + 1$. The j 'th neuron computes a weighted average with inputs from all neurons in the previous layer, i.e.

$$v_j^r = \sum_{k=0}^{k_{r-1}} w_{jk}^r \times y_k^{r-1}, \quad (\text{A.1})$$

where y_k^{r-1} is

$$y_k^{r-1} = a(v_k^{r-1}), \quad (\text{A.2})$$

and $a(\cdot)$ is a nonlinear activation function. This is shown in Figure A.1.

Backpropagation algorithm

Assume we have a data set with N pairs $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y)\}$,

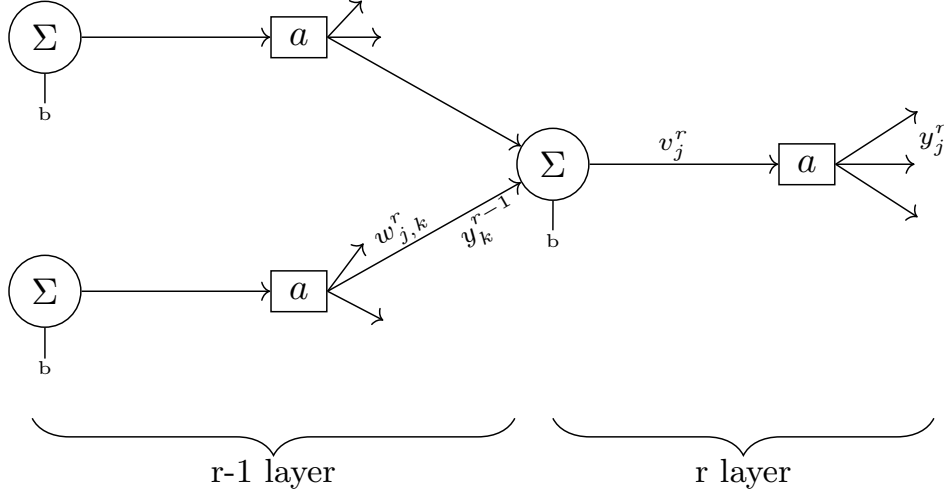


Figure A.1: Fully connected neurons in multilayer perceptron neural networks.

where $\mathbf{x}_i \in \mathbb{R}^\ell$ and $y \in [0, 1]$. Further, let the objective function be

$$\mathcal{L} = \sum_{i=1}^N \epsilon(i), \quad (\text{A.3})$$

where $\epsilon(i)$, at the output layer, is

$$\epsilon(i) = \frac{1}{2} \sum_{m=1}^{k_L} e_m(i) \equiv \frac{1}{2} \sum_{m=1}^{k_L} (y_m(i) - \hat{y}_m(i))^2. \quad (\text{A.4})$$

The backpropagation algorithm minimizes the loss function by adjusting the weight vectors \mathbf{w}_j^r , for $r = 1, \dots, L$ and $j = 1, \dots, k_r$, in the direction of deepest steep in the objective function at each step τ . See Figure A.2

The iteration starts at the output layer L and propagates backwards using the chain rule of differentiation

$$\nabla \epsilon(\mathbf{w}_j^r) = \frac{\partial v_j^r}{\partial \mathbf{w}_j^r} \frac{\partial \epsilon}{\partial v_j^r}. \quad (\text{A.5})$$

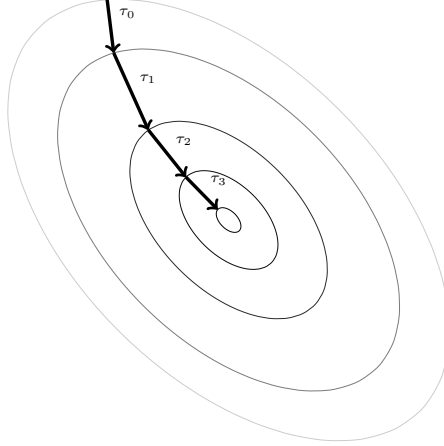


Figure A.2: Gradient descent algorithm. At each iteration the loss function is minimized by adjusting the weights \mathbf{w}_j^r in the direction of deepest step.

Note that the error ϵ and the signal v_j^r is with respect to the i 'th observation in the data set, but we omit the subscript to do not clutter the notation. In what follows, we continue with this simplified notation.

Let us obtain the first gradient in the right hand side of Equation A.5, which is the same for all layers,

$$\begin{aligned}
 \frac{\partial v_j^r}{\partial \mathbf{w}_j^r} &= \begin{pmatrix} \frac{\partial}{\partial w_{j0}} v_j^r \\ \frac{\partial}{\partial w_{j1}} v_j^r \\ \vdots \\ \frac{\partial}{\partial w_{jk_{r-1}}} v_j^r \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial w_{j0}} \sum_{k=0}^{k_{r-1}} w_{jk}^r y_k^{r-1} \\ \frac{\partial}{\partial w_{j1}} \sum_{k=0}^{k_{r-1}} w_{jk}^r y_k^{r-1} \\ \vdots \\ \frac{\partial}{\partial w_{jk_{r-1}}} \sum_{k=0}^{k_{r-1}} w_{jk}^r y_k^{r-1} \end{pmatrix} \\
 &= \begin{pmatrix} 1 \\ y_1^{r-1} \\ \vdots \\ y_{k_{r-1}}^{r-1} \end{pmatrix} \\
 &= \mathbf{y}^{r-1}.
 \end{aligned} \tag{A.6}$$

The second term of Equation A.5 at the output layer L is

$$\begin{aligned}\frac{\partial \epsilon}{\partial v_j^L} &= \frac{\partial \frac{1}{2} \sum_{m=1}^{k_L} (a(v_m^{k_L}) - y_m)^2}{\partial v_j^L} \\ &= e_j a'(v_j^L)\end{aligned}\quad (\text{A.7})$$

where a' is the derivative of the activation function $a(\cdot)$ and we define

$$\frac{\partial \epsilon}{\partial v_j^r} \equiv \delta_j^r, \quad (\text{A.8})$$

hence $\delta_j^L = e_j a'(v_j^L)$.

To derive the second term of Equation A.5 for layers $r < L$ we need to take into account the influence of the signal v_j^{r-1} on all v_k^r neurons in the r 'th layer, for $k = 1, 2, \dots, k_r$. Using the chain rule of differentiation

$$\frac{\partial \epsilon}{\partial v_j^{r-1}} = \sum_{k=1}^{k_r} \frac{\partial \epsilon}{\partial v_k^r} \frac{\partial v_k^r}{\partial v_j^{r-1}} \quad (\text{A.9})$$

or equivalently

$$\begin{aligned}\delta_j^{r-1} &= \sum_{k=1}^{k_r} \delta_k^r \frac{\partial v_k^r}{\partial v_j^{r-1}} \\ &= \sum_{k=1}^{k_r} \delta_k^r \frac{\partial \sum_{m=0}^{k_r-1} w_{km}^r a(v_m^{r-1})}{\partial v_j^{r-1}} \\ &= \sum_{k=1}^{k_r} \delta_k^r w_{kj}^r a'(v_j^{r-1}) \\ &\equiv e_j^{r-1} a'(v_j^{r-1}),\end{aligned}\quad (\text{A.10})$$

where $e_j^{r-1} = \sum_{k=1}^{k_r} \delta_k^r w_{kj}^r$.

We can now use Equation A.7 and A.10 to find δ_j^r for $r = 1, 2, \dots, L$ and $j = 1, 2, \dots, k_r$ and update the weights at each iteration τ using the equation

$$\mathbf{w}_j^r(\tau + 1) = \mathbf{w}_j^r(\tau) - \mu \sum_{i=1}^N \delta_j^r(i) \mathbf{y}_j^{r-1}(i), \quad (\text{A.11})$$

where μ is the learning rate.

Finally, there are different continuous differentiable activation functions that we can choose from, e.g. sigmoid, softplus, relu, tanh etc. In any case, the updating scheme is given by Equation A.11 where we insert the appropriate expression for a' . See Figure A.1 for some common examples for activation functions. Finally, the objective function does not necessarily need to be Equation A.3. Another popular objective function for classification problems is cross-entropy.

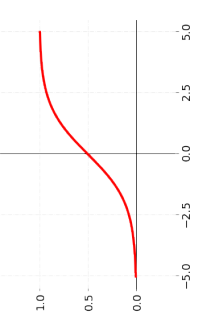
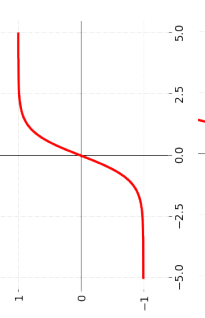
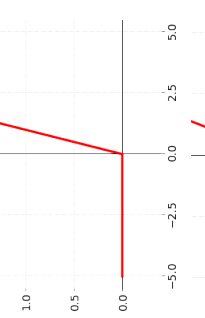
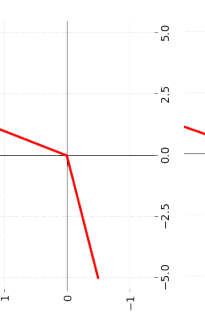
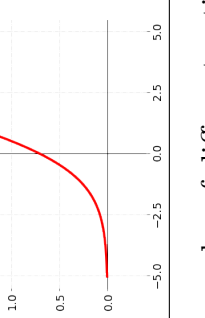
Name	Diagram	Function	Derivative
Sigmoid		$f(x) = \frac{1}{1 + \exp^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \frac{2}{1 + \exp^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Relu		$f(x) = \begin{cases} 0 & \text{if } x \leq 0. \\ x & \text{otherwise.} \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \leq 0. \\ 1 & \text{otherwise.} \end{cases}$
Leaky Relu		$f(x) = \begin{cases} ax & \text{if } x \leq 0. \\ x & \text{otherwise.} \end{cases}$	$f'(x) = \begin{cases} a & \text{for } x \leq 0. \\ 1 & \text{otherwise.} \end{cases}$
Softplus		$f(x) = \log(1 + \exp(x))$	$f'(x) = \frac{1}{1 + \exp^{-x}}$

Table A.1: Examples of different activation functions together with their partial derivatives.

Appendix B

Segment based credit scoring

The rationale behind the segment-based credit scoring approach is that in a given loan portfolio, e.g. credit cards or mortgage loan portfolio, there are different groups of customers, whose creditworthiness is described by different variables. Therefore, developing one classifier for each group should achieve higher accuracy compared to developing only one classifier for all customers. However, the challenge is how do we identify such groups with different creditworthiness profiles. In this section, we extend our work done in Paper I and compare the performance of a segment-based approach with the performance of a classical approach for credit scoring where only one model is developed for all customers. To that end, we develop one classifier for each of the clusters found in Paper I. As shown in Paper I, the clustering structure in the latent space \mathbf{z} have statistically different creditworthiness and therefore the clusters are well suited for the credit scoring segment-based approach.

B.0.1 Methodology

We divide the data set $\mathbf{X} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y)\}$ according with the label y_i . That is, we use 30% of data where $y = 0$ to train a VAE using our semi-supervised approach introduced in Paper I. After training the VAE, we generate the latent representation \mathbf{z} for the remaining data (100% $y = 1$ and 70% $y = 1$) with the previously trained VAE. Further, for each cluster revealed in the latent space, we use 70%-30% of the data set for training and testing an MLP classifier, see Figure (B.1). Note that the test data set

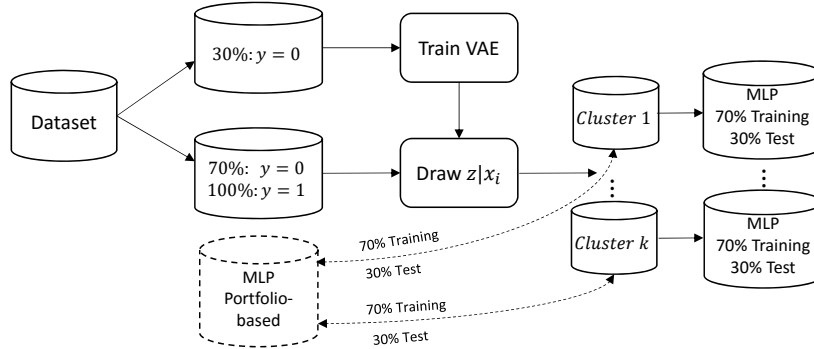


Figure B.1: We use 30% of the data where $y = 0$ for training a VAE and to generate the latent space z for the remaining data. Based on the clusters in the latent space, we develop MLP classifier using a classical 70%-30% partition for training and testing respectively.

keeps the original class ratio between the majority and minority class, while for the training data set we generate synthetic variables to balance the data before training the MLPs. Finally, we use grid-search for hyper-parameter tuning and we measure model performance based on the average values of a 10 cross-validation. We use the H-measure as the main metric to assess performance because the H-measure is well suited for class imbalanced data sets and it is more appropriate for comparing multiple models. For further details refer to Hand (2009); Hand and Anagnostopoulos (2014). We also measure model performance using other well-known metrics, e.g., the area under the receiver operating characteristic curve (AUC), the Gini coefficient, and the Kolmogorov-Smirnov (KS) test.

B.0.2 Results

Norwegian car loan data set

The classification results for the Norwegian car loan dataset are shown in Table (B.1). Based on the H-measure, the performance of the segment-based approach is better than that of the portfolio-based approach for all clusters. Note that the difference in H-measure between the two approaches is smallest for cluster 3, which may suggest that the largest cluster drives model performance in the portfolio-based approach. For clusters with high-risk profiles, there are large difference in performance, suggesting that the

Norwegian car loan				
Performance metric	Cluster	Segment-based	Portfolio-based	p -value
H-measure	1	0.2774	0.2310	0.0509
	2	0.1665	0.1453	0.4299
	3	0.2174	0.2076	0.0854
	4	0.1760	0.1471	0.0186
	5	0.1302	0.1193	0.0931
AUC	1	0.7688	0.7430	
	2	0.6791	0.6701	
	3	0.7756	0.7706	
	4	0.7395	0.7188	
	5	0.7021	0.6923	
Gini	1	0.5377	0.4860	
	2	0.3582	0.3402	
	3	0.5511	0.5412	
	4	0.4790	0.4377	
	5	0.4043	0.3846	
KS	1	0.4648	0.4098	
	2	0.3441	0.3280	
	3	0.4318	0.4199	
	4	0.3821	0.3489	
	5	0.3410	0.3299	

Table B.1: Average model performance, for the segment-based credit scoring approach and for the portfolio-based approach, based on a 10-cross-validation approach. Note that best models are selected based on the H-measure.

underlying risk drivers in such clusters are different. Hence, it is advantageous to build different classifier models for groups of customers with distinct risk profiles.

To get a better overview of model performance as measured by the H-measure, we conduct an unpaired t-test. This statistical test checks whether the average difference in model performance between the two approaches is significantly different from zero. The p -values of the t-test for the Norwegian car loan dataset are shown in the third column of Table (B.1). It is not surprising that the difference in H-measures for cluster 2 are not significant, given the number of observations and few customers from the minority class (only 87) in that cluster. The difference in H-measures for clusters 3 and 5 are significant at the 10% level, while those for clusters 1 and 4 are significant at the 5% level. The H-measure, as well as the rest of performance metrics in Table (B.1), consistently rank the model performance for the segment-based approach on top of the portfolio-based approach. Hence, the insight provided by the t-test should be taken as informative and not as conclusive.

Kaggle and Finnish car loan dataset

Table (B.2) shows the performance of the segment-based and portfolio-based approach for the Kaggle and Finnish car loan dataset. For the Kaggle

Performance metric	Cluster	Kaggle			Finnish car loan		
		Segment-based	Portfolio-based	p -value	Segment-based	Portfolio-based	p -value
H-measure	1	0.2918	0.2842	0.0410	0.1949	0.2388	0.4141
	2	0.1145	0.1067	0.2230	0.0946	0.1001	0.5870
	3	0.0828	0.0691	0.0883	0.1838	0.1817	0.8305
	4	0.0765	0.0660	0.0626			
AUC	1	0.8047	0.8018		0.5916	0.6247	
	2	0.6680	0.6629		0.6491	0.6596	
	3	0.6284	0.6126		0.7366	0.7370	
	4	0.6270	0.6189				
Gini	1	0.6094	0.6036		0.1832	0.2495	
	2	0.3360	0.3258		0.2983	0.3193	
	3	0.2569	0.2253		0.4733	0.4741	
	4	0.2540	0.2379				
KS	1	0.4711	0.4633		0.3290	0.3504	
	2	0.2550	0.2480		0.2797	0.3009	
	3	0.2218	0.1817		0.3735	0.3688	
	4	0.2067	0.2034				

Table B.2: Average model performance, for the segment-based credit scoring approach and for the portfolio-based approach, based on a 10-cross-validation approach. Note that best models are selected based on the H-measure.

dataset, the performance of the segment-based approach is better for all clusters. We can see the same pattern as for the Norwegian car loan dataset. The smallest performance gain of the segment-based approach is for the largest cluster, and there are larger differences in performance for the high-risk clusters.

For the Finnish dataset, the segment-based approach does not increase model performance. This result can be driven by the total number of customers from the minority class which is only 939 for the entire dataset. Furthermore, these 939 customers are spread across three different clusters, i.e. 25, 228 and 685 customers from the minority class in cluster 1, 2 and 3 respectively. It is challenging to train classifier models when the number of customers from the minority class is small.

B.0.3 Conclusion

Our results show that for portfolios with sufficient number of customers across the different clusters, developing one classifier for each cluster in the latent space achieves higher model performance relative to the traditional approach for credit scoring. The clusters in the latent space are revealed by our proposed methodology in Paper I.

The segment-based credit scoring performs better in clusters with high-risk

profiles, suggesting that the underlying risk drivers in each cluster might be different. We let for future research a comprehensive study about segment-based credit scoring where we attempt to develop a more complex approach for segment-based credit scoring and where we compare the clusters revealed by our proposed methodology in Paper I with other clustering techniques, e.g. k-means.

Bibliography

- Abdou, H., Pointon, J., and El-Masry, A. (2008). Neural nets versus conventional techniques in credit scoring in Egyptian banking. *Expert Systems with Applications*, 35(3):1275–1292.
- Abdou, H. A. (2009). Genetic programming for credit scoring: The case of egyptian public sector banks. *Expert systems with applications*, 36(9):11402–11417.
- Abdou, H. A., Mitra, S., Fry, J., and Elamer, A. A. (2019). Would two-stage scoring models alleviate bank exposure to bad debt? *Expert Systems with Applications*, 128:1–13.
- Alemi, A. A., Poole, B., Fischer, I., Dillon, J. V., Saurous, R. A., and Murphy, K. (2018). Fixing a Broken ELBO. *arXiv:1711.00464 [cs, stat]*.
- Altman, E. I. and Saunders, A. (1997). Credit risk measurement: Developments over the last 20 years. *Journal of banking & finance*, 21(11-12):1721–1742.
- Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.
- Anderson, R. (2007). *The credit scoring toolkit: theory and practice for retail credit risk management and decision automation*. Oxford University Press.
- Angelini, E., di Tollo, G., and Roli, A. (2008). A neural network approach for credit risk evaluation. *Quarterly Review of Economics and Finance*, 48(4):733–755.

- Baesens, B., Setiono, R., Mues, C., and Vanthienen, J. (2003). Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3):312–329.
- Baesens, B., Van Gestel, T., Stepanova, M., Van Den Poel, D., and Vanthienen, J. (2005). Neural network survival analysis for personal loan data. *Journal of the Operational Research Society*, 56(9):1089–1098.
- Bajgier, S. M. and Hill, A. V. (1982). An experimental comparison of statistical and linear programming approaches to the discriminant problem. *Decision Sciences*, 13(4):604–618.
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.
- Blanco, A., Pino-Mejías, R., Lara, J., and Rayo, S. (2013). Credit scoring models for the microfinance industry using neural networks: Evidence from Peru. *Expert Systems with Applications*, 40(1):356–364.
- Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.
- Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.
- Breiman, L., Friedman, J., Olshen, R. A., and Stone, C. J. (1984). Classification and regression trees chapman & hall. *New York*.
- Burda, Y., Grosse, R., and Salakhutdinov, R. (2016). Importance Weighted Autoencoders. *arXiv:1509.00519 [cs, stat]*.
- Byanjankar, A., Heikkila, M., and Mezei, J. (2015). Predicting credit risk in peer-to-peer lending: A neural network approach. *Proceedings - 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*, pages 719–725.
- Chuang, C. L. and Huang, S. T. (2011). A hybrid neural network approach for credit scoring. *Expert Systems*, 28(2):185–196.
- Çınlar, E. (2011). *Probability and stochastics*, volume 261. Springer Science & Business Media.

- Cremer, C., Morris, Q., and Duvenaud, D. (2017). Reinterpreting Importance-Weighted Autoencoders. *arXiv:1704.02916 [stat]*.
- Desai, V. S., Conway, D. G., Crook, J. N., and Overstreet, G. A. (1997). Credit-scoring models in the credit-union environment using neural networks and genetic algorithms. *IMA Journal of Management Mathematics*, 8(4):323–346.
- Dieng, A. B., Kim, Y., Rush, A. M., and Blei, D. M. (2019). Avoiding Latent Variable Collapse With Generative Skip Models. *arXiv:1807.04863 [cs, stat]*.
- Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.
- Domke, J. and Sheldon, D. (2018). Importance Weighting and Variational Inference. *arXiv:1808.09034 [cs, stat]*.
- Durand, D. (1941). *Risk elements in consumer installment financing*. National Bureau of Economic Research, New York.
- Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.
- Freed, N. and Glover, F. (1981). Applications and implementation: A linear programming approach to the discriminant problem. *Decision Sciences*, 12(1):68–74.
- Fu, X., Wei, Y., Xu, F., Wang, T., Lu, Y., Li, J., and Huang, J. Z. (2019). Semi-supervised aspect-level sentiment classification model based on variational autoencoder. *Knowledge-Based Systems*, 171:81–92.
- Gal, Y. (2016). Uncertainty in deep learning. *University of Cambridge*, 1:3.
- Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889.

- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.
- Hand, D. J. (2009). Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine learning*, 77(1):103–123.
- Hand, D. J. and Anagnostopoulos, C. (2014). A better beta for the h measure of classification performance. *Pattern Recognition Letters*, 40:41–46.
- Hardy Jr, W. E. and Adrian Jr, J. L. (1985). A linear programming alternative to discriminant analysis in credit scoring. *Agribusiness*, 1(4):285–292.
- Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework.
- Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.
- Hsieh, N. C. (2005). Hybrid mining approach in the design of credit scoring models. *Expert Systems with Applications*, 28(4):655–665.
- Hsu, W.-N., Zhang, Y., and Glass, J. (2017). Learning latent representations for speech generation and transformation. *arXiv preprint arXiv:1704.04222*.
- Jensen, H. L. (1992). Using Neural Networks for Credit Scoring. *Managerial Finance*, 18(6):15–26.
- Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

- Khandani, A. E., Kim, A. J., and Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787.
- Khashman, A. (2010). Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. *Expert Systems with Applications*, 37(9):6233–6239.
- Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.
- Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.
- Lai, K. K., Yu, L., Wang, S., and Zhou, L. (2006). Neural network meta-learning for credit scoring. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4113 LNCS:403–408.
- Latif, S., Rana, R., Qadir, J., and Epps, J. (2017). Variational autoencoders for learning latent representations of speech emotion. *arXiv preprint arXiv:1712.08708*.
- Lazarsfeld, P. F. (1950). The logical and mathematical foundation of latent structure analysis. *Studies in Social Psychology in World War II Vol. IV : Measurement and Prediction*, pages 362–412.
- Lazarsfeld, P. F. (1954). A conceptual introduction to latent structure analysis. *Mathematical thinking in the social sciences*, 1:349–387.
- LeCun, Y. (2018). The power and limits of deep learning: In his iri medal address, yann lecun maps the development of machine learning techniques

- and suggests what the future may hold. *Research-Technology Management*, 61(6):22–27.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Lee, T. S. and Chen, I. F. (2005). A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. *Expert Systems with Applications*, 28(4):743–752.
- Lee, T. S., Chiu, C. C., Lu, C. J., and Chen, I. F. (2002). Credit scoring using the hybrid neural discriminant technique. *Expert Systems with Applications*, 23(3):245–254.
- Lessmann, S., Baesens, B., Seow, H.-V., and Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1):124–136.
- Lessmann, S., Seow, H., Baesens, B., and Thomas, L. C. (2013). Benchmarking state-of-the-art classification algorithms for credit scoring: A ten-year update. In *Credit Research Centre, Conference Archive*.
- Li, Y. and Turner, R. E. (2016). Rényi Divergence Variational Inference. *arXiv:1602.02311 [cs, stat]*.
- Lucas, J., Tucker, G., Grosse, R., and Norouzi, M. (2019). Don't Blame the ELBO! A Linear VAE Perspective on Posterior Collapse. *arXiv:1911.02469 [cs, stat]*.
- Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. (2016). Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*.
- Malhotra, R. and Malhotra, D. K. (2003). Evaluating consumer loans using neural networks. *Omega*, 31(2):83–96.
- Mbuvha, R., Boulkaibet, I., and Marwala, T. (2019). Bayesian automatic relevance determination for feature selection in credit default modelling. In *International Conference on Artificial Neural Networks*, pages 420–425. Springer.

- Mnih, V., Heess, N., Graves, A., et al. (2014). Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212.
- Morgan, J. N. and Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association*, 58(302):415–434.
- Munkhdalai, L., Lee, J. Y., and Ryu, K. H. (2020). A hybrid credit scoring model using neural networks and logistic regression. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, pages 251–258. Springer.
- Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.
- Neagoie, V.-E., Ciotec, A.-D., and Cucu, G.-S. (2018). Deep convolutional neural networks versus multilayer perceptron for financial prediction. In *2018 International Conference on Communications (COMM)*, pages 201–206. IEEE.
- Pławiak, P., Abdar, M., Pławiak, J., Makarenkov, V., and Acharya, U. R. (2020). Dghnl: A new deep genetic hierarchical network of learners for prediction of credit scoring. *Information Sciences*, 516:401–418.
- Quinlan, J. (1993). C4.5: Program for machine learning.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.
- Rampasek, L. and Goldenberg, A. (2017). Dr. vae: Drug response variational autoencoder. *arXiv preprint arXiv:1706.08203*.
- Ranganath, R., Altosaar, J., Tran, D., and Blei, D. M. (2018). Operator Variational Inference. *arXiv:1610.09033 [cs, stat]*.
- Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822.
- Rezende, D. J. and Mohamed, S. (2016). Variational Inference with Normalizing Flows. *arXiv:1505.05770 [cs, stat]*.

- Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic back-propagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.
- Ruiz, F. R., AUEB, M. T. R., and Blei, D. (2016). The generalized reparameterization gradient. In *Advances in neural information processing systems*, pages 460–468.
- Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.
- Salimans, T., Kingma, D., and Welling, M. (2015). Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226.
- Shen, F., Zhao, X., Li, Z., Li, K., and Meng, Z. (2019). A novel ensemble classification model based on neural networks and a classifier optimisation technique for imbalanced credit risk evaluation. *Physica A: Statistical Mechanics and its Applications*, 526:121073.
- Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016). Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746.
- Su, J., Wu, S., Zhang, B., Wu, C., Qin, Y., and Xiong, D. (2018). A neural generative autoencoder for bilingual word embeddings. *Information Sciences*, 424:287–300.
- Sun, T. and Vasarhelyi, M. A. (2018). Predicting credit card delinquencies: An application of deep neural networks. *Intelligent Systems in Accounting, Finance and Management*, 25(4):174–189.
- Šušteršič, M., Mramor, D., and Zupan, J. (2009). Consumer credit scoring models with limited data. *Expert Systems with Applications*, 36(3 PART 1):4736–4744.
- Tam, K. Y. and Kiang, M. (1990). Predicting bank failures: A neural network approach. *Applied Artificial Intelligence an International Journal*, 4(4):265–282.

- Tam, K. Y. and Kiang, M. Y. (1992). Managerial applications of neural networks: the case of bank failure predictions. *Management science*, 38(7):926–947.
- Theodoridis, S. and Koutroumbas, K. (2011). *Pattern recognition*. Academic Press.
- Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International journal of forecasting*, 16(2):149–172.
- Titus, A. J., Bobak, C. A., and Christensen, B. C. (2018). A new dimension of breast cancer epigenetics - applications of variational autoencoders with dna methylation. In *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 4: BIOINFORMATICS*,, pages 140–145. INSTICC, SciTePress.
- Tran, D., Ranganath, R., and Blei, D. M. (2016). The Variational Gaussian Process. *arXiv:1511.06499 [cs, stat]*.
- Tsai, C. F. and Wu, J. W. (2008). Using neural network ensembles for bankruptcy prediction and credit scoring. *Expert Systems with Applications*, 34(4):2639–2649.
- Wainwright, M. J., Jordan, M. I., et al. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.
- Wang, C., Han, D., Liu, Q., and Luo, S. (2018). A deep learning approach for credit scoring of peer-to-peer lending using attention mechanism lstm. *IEEE Access*, 7:2161–2168.
- Waterhouse, S. R., MacKay, D., and Robinson, A. J. (1996). Bayesian methods for mixtures of experts. In *Advances in neural information processing systems*, pages 351–357.
- Way, G. P. and Greene, C. S. (2017a). Evaluating deep variational autoencoders trained on pan-cancer gene expression. *arXiv preprint arXiv:1711.04828*.

- Way, G. P. and Greene, C. S. (2017b). Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. *BioRxiv*.
- West, D. (2000). Neural network credit scoring models. *Computers and Operations Research*, 27(11-12):1131–1152.
- Wu, C., Wu, F., Wu, S., Yuan, Z., Liu, J., and Huang, Y. (2019). Semi-supervised dimensional sentiment analysis with variational autoencoder. *Knowledge-Based Systems*, 165:30–39.
- Yobas, M. B., Crook, J. N., and Ross, P. (2000). Credit scoring using neural and evolutionary techniques. *IMA Journal of Management Mathematics*, 11(2):111–125.
- Zekic-Susac, M., Sarlija, N., and Bensic, M. (2004). Small business credit scoring: A comparison of logistic regression, neural network, and decision tree models. *Proceedings of the International Conference on Information Technology Interfaces, ITI*, pages 265–270.
- Zhang, C., Butepage, J., Kjellstrom, H., and Mandt, S. (2018a). Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*.
- Zhang, H., He, H., and Zhang, W. (2018b). Classifier selection and clustering with fuzzy assignment in ensemble model for credit scoring. *Neurocomputing*, 316:210–221.
- Zhao, Z., Xu, S., Kang, B. H., Kabir, M. M. J., Liu, Y., and Wasinger, R. (2015). Investigation and improvement of multi-layer perception neural networks for credit scoring. *Expert Systems with Applications*, 42(7):3508–3516.
- Zheng, Y., Tan, H., Tang, B., Zhou, H., et al. (2016). Variational deep embedding: A generative approach to clustering. arxiv preprint. *arXiv preprint arXiv:1611.05148*, 1(2):5.
- Zhu, B., Yang, W., Wang, H., and Yuan, Y. (2018). A hybrid deep learning model for consumer credit scoring. In *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 205–208. IEEE.