IMPROVING PREDICTIVE MODELING VIA EFFECTIVE FEATURE SELECTION AND REPRESENTATION LEARNING

by

XIANGRUI LI

DISSERTATION

Submitted to the Graduate School,

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2020

MAJOR: COMPUTER SCIENCE

Approved By:

_____

Advisor Date

_____

_____

_____

DEDICATION

To my parents for their love.

ACKNOWLEDGEMENTS

I would like to express my gratitude and appreciation to my advisor Dr. Dongxiao Zhu for his support in the past years. His patience, knowledge and caring have led my way to the field of machine learning. Under his guidance, I have learned how to discover research topics, review related literatures and propose solutions to machine learning problems. The researching experience with him would be invaluable wealth for my future and career. I am also grateful to Dr. Ming Dong, Dr. Hengguang Li and Dr. Alexander Kotov for serving as my committee members on my PhD dissertation. Their insightful comments would benefit me for further improvements of my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

Predictive modeling (a.k.a. supervised learning) is a machine learning paradigm that learns a mapping from an input to outputs based on the historical data of input-output pairs (i.e., feature-target). For complex and high-dimensional data, fitting supervised machine learning models with raw input features can result in severe overfitting and poor generalization performance. This is due to that high dimensional data often has redundant features that contains much noise. Hence, effectively capturing the useful information in features is the key to build accurate predictive models.

In this dissertation, we develop novel methods for predictive modeling that can be categorized into (1) feature selection for high-dimensional numerical data (small n, large p) based on generalized linear models and (2) new loss function and in-training regularization for effective representation learning using deep neural networks (DNNs). As predictive modeling is practically important and useful in many applications, we also apply DNNs in healthcare informatics for cardiovascular disease prediction.

1.1 Feature Selection for Predictive Modeling

In the past few years, high dimensional data of small sample size, large feature size has been proliferated from various areas. These areas include text mining, bioinformatics, computer vision, health care and e-commerce. Due to the curse of dimensionality, traditional generalized linear models generally fail in those applications due to overfitting, resulting poor predictive performance for unseen data. With a large number of features, models are also difficult to interpret; yet in applications such as cancer classification, model interpretability is a major concern. To this end, dimension reduction is often desired.

Feature selection [43] is one of the most powerful dimension reduction techniques that

can simultaneously address those two challenges. For example, in bioinformatics [1], the genetic datasets consist of only a few examples with enormous expression data of thousands of genes. The task is to correctly predict the cancer phenotype of patients as well as identify the risk genes that are most correlated. Another example is the predictive modeling using electronic health record (EHR) in healthcare informatics (HI) [150, 151]. In HI, the EHR for each patient consists of different sets of features such demographic information, lab results and disease diagnosis. Identification of the true risk factors from those features can not only improve the predictive performance, but also help clinicians understand the disease progression. Hence, feature selection is well suited in those applications.

There are various feature selection techniques developed from different perspectives, such as information theory-based, similarity-based and statistical-based [72]. Among them, the penalized (generalized) linear model with sparsity-inducing regularization is one of the most popular methods, where the feature selection is incorporated into the model optimization process. For example, Lasso [144, 31] uses l_1-norm on the model parameters and achieves feature selection by forcing subset of parameters exactly to zero. In cases that features can be grouped, group Lasso (GLasso) [96, 160] extends Lasso by introducing the group structure and be capable of achieving sparsity at the group level. To achieve within-group sparsity, Sparse group Lasso (SGL) [148] combines Lasso and Group Lasso.

Part of my work also develops new models with sparsity-inducing regularizations. More specifically, we focus on (1) the multinomial classification problem where different classes have different important features and feature groups [80] and (2) finite mixture of regression where different mixture components may share same features as well as have its own

important features [78].

1.2 Representation Learning for Predictive Modeling

Deep neural networks (DNNs) have achieved great success for difficult predictive tasks in speech recognition, computer vision and healthcare informatics [12, 68, 103, 14]. This is due to DNN's capability of learning high-level feature representations, rendering better predictions based on those abstract features. As DNNs for supervised learning can be viewed as a pipeline of a feature extractor (i.e., the last hidder layer) and an output layer (i.e., regressor for regression, classier for classification), the effectiveness of the feature extractor is critical for DNN's predictive performance.

Modern architectures of DNNs usually have an extremely large number of model parameters, which often outnumbers the available training data. Recent studies in theoretical deep learning have shown that DNNs can achieve good generalization even with the over-parameterization [102, 106]. Although over-parameterization may not be very damaging to DNN's overall generalizability, DNNs can still overfit the noise within the training data (e.g., sampling noise in data collection) due to its highly expressive power. This makes DNNs sensitive to small perturbations in testing data, for example, adversarial samples [38, 79, 77]. To alleviate overfitting of DNNs and learn better feature representations, many regularization methods have been proposed. These include classic ones such as early stopping, L_1 and L_2 regularization [37], and more recent ones such as dropout [131], batch normalization [58] and data-augmentation types of regularization (e.g., cutout [23], shake-shake [34]). There are also other machine learning regimes that can achieve regularization effect such as transfer learning [107] and multi-task learning [6, 123].

In addition to the regularization approach, another line of research focuses on design-

ing new loss functions to learn feature representation with more discriminative power. For predictive modeling, DNNs are usually trained with the loss functions of supervision (e.g., cross-entropy loss). In the training process, model learning is only supervised by the signal of loss functions and does not impose restrictions on the distribution of feature representations. To this end, the performance of DNNs can be further boosted by learning features with large inter-class separability and strong intra-class compactness. For example, [154] proposes the center loss that encourages the clustering effect for each class. [11] designs a virtual loss that inserts virtual class between each pair of true classes so that the inter-class separability is maximized. In [88], an angular margin is imposed to increase inter-class separability in terms of cosine distance.

One of our work [75] designs a new loss function which is originally motivated in exploring the property of the logistic and softmax loss functions. It can enhance the discriminative power of feature representations learned by DNN. Another work [76] proposes a regularization method that is embedded in stochastic gradient descent (SGD) procedure to reduce the adaption of model parameters to mini-batches.

1.3 Contribution

In this section, we describe our contributions of this dissertation as follows.

1.3.1 Feature Selection in Generalized Linear Models

For multinomial classification A regularized multinomial logistic model, termed as class-conditional sparse overlapping group lasso (CCSOGL), is proposed. In CCSOGL, we incorporate information of feature groups in CCSOGL in the class-conditional fashion. This flexibility makes CCSOGL capable of achieving class-specific sparsity pattern at the group level and further selecting relevant features within the selected feature group, which fits

many applications well and enhances model performances and interpretability. To solve the optimization problem, we develop a block coordinate descent algorithm that solves CCSOGL efficiently.

For mixture of gaussian regression We propose a novel penalized finite mixture of Gaussian regression model with structured feature selection that explicitly incorporates the information of parameter grouping via matrix $l_{2,1}$-norm. The $l_{2,1}$ penalty has one appealing property that it performs feature selection at the (parameter) group level, encouraging the same sparsity pattern across all mixture components. Our approach is more robust to noisy features and model noise as demonstrated in simulation studies. We further extend $l_{2,1}$ to flexible sparse $l_{2,1}$ penalty by incorporating l_1-norm. The sparse $l_{2,1}$ penalized FMR allows heterogeneous feature structures across mixture components while possesses robustness of $l_{2,1}$ penalty.

The resultant penalized MLE in FMR is formulated as a non-convex optimization problem. The standard approach for penalized FMR is EM-type algorithm; the non-smoothness of sparse $l_{2,1}$ penalty, however, poses substantial challenges in the M-step of EM algorithm. Inspired by a re-parametrization trick in [132], we combine block coordinate descent algorithm and majorizing-minimization scheme for numerical optimization in the M-step, with efficient closed-form updates in each iteration. Finally, we apply our method to evaluate its performance using simulation and real data sets.

1.3.2 Predictive Modeling for Healthcare Informatics

Cardiovascular disease prediction Two DNN models are developed for risk quantification in healthcare informatics.

We propose the auxiliary-task augmented network (ATAN), a model that predicts the

risk of cardiovascular disease with introducing clinically relevant measures as auxiliary predictive tasks. With auxiliary tasks, ATAN takes advantage of multi-task learning (MTL) — an approach of regularization and implicit data augmentation. Without assumption of homogeneous feature representation for all tasks in classic multi-task frameworks, ATAN explicitly models the shared feature representation for all tasks, as well as task-specific representation, and combines them together using a weighting mechanism to capture the clinical relevance. By the weighting mechanism, we conceptually quantify the relevance between the primary and auxiliary target.

We also introduce a unified DNN model, termed as deep mixture of neural networks (DMNN), that simultaneously predicts clinical outcomes and discover patient subgroups. DMNN consists of an embedding network with gating (ENG) and several local predictive networks (LPNs). ENG embeds raw input features into high-level feature representation that is further used as input for LPNs. Unlike the existing DNN models without subgroup identification, patients will be grouped that share similar functional relations between inputs and clinical outcomes via the gating mechanism of ENG. Each functional relation is modeled by one LPN. The subgroup discoveries enable us to apply existing interpretation techniques to identify subgroup-specific risk factors. By explaining the local input-outcome relations captured by LPN within each patient subgroup, the subgroup-specific sets of risk factors enable us to discover health disparities.

1.3.3 Effective Feature Learning of DNNs

In-negative-class reweighted logistic loss for DNNs For logistic loss (LGL)and soft-max loss (SML), we provide a theoretical derivation on the relation of model predicted probability, class weights in the loss function and sample sizes in a system of equations.

We depict the learning property for LGL and SML for classification problems based on those probability equations. The multi-modality neglect problem in LGL is then identified which is the main obstacle for LGL's application in multi-class classification. To remedy this problem, we propose a novel learning objective, in-negative class reweighted LGL, as a competitive alternative for LGL and SML for multi-class classification.

Learning compact features via in-training representation alignment We propose a training strategy in-training feature alignment (ITRA) for training DNNs. ITRA augments conventional SGD with regularization by additionally forcing feature alignment of different mini-batches using maximum mean discrepancy (MMD) to reduce mini-batch over-adaption. We show in in-depth analysis that ITRA enjoys three theoretical merits that can improve the feature learning of DNNs: (1) learning compact feature representations; (2) reducing over-adaption to mini-batches; (3) accommodating multi-modalities of the data distribution. ITRA can be combined with existing regularization approaches and applied on a broad range of network architectures and loss functions.

1.4 Organization

The rest of this dissertation is organized as follows. In Chapter 2, we introduce the CCSOGL method for multi-class logistic regression. In Chapter 3, we describe the robust feature selection using $l_{2,1}$-norm for finite mixture of Gaussian regression. In Chapter 4, we apply DNNs under the framework of multi-task learning and mixture model for disease risk prediction. In Chapter 5, we derive the learning property of logistic and softmax losses for deep neural network and further propose an improved version of logistic loss for classification. Chapter 6 introduces a new training strategy that acts as a regularization method for training DNNs. Finally, Chapter 7 concludes this dissertation with discussion.

CHAPTER 2 FEATURE SELECTION FOR MULTINOMIAL CLASSIFICATION

Regularized multinomial logistic model is widely used in multi-class classification problems. For high dimension data, various regularization methods achieving sparsity have been developed and applied successfully to many real-world applications such as bioinformatics, health informatics and text mining. In many cases there exists intrinsic group structures among the features, incorporating the group information in the model can enhance model performance. In multi-class classification, different classes may relate to different feature groups. With these considerations, we propose a class-conditional regularization of the multinomial logistic model (CCSOGL) to enable the discovery of class-specific feature groups. To solve the model, we developed an efficient cyclic block coordinate descent based algorithm.

2.1 Introduction

Multinomial logistic regression is one of the most popular discriminative methods for multi-class classification problems. It directly models the probabilities of a sample belonging to each class. The typical approach for model training is to maximize likelihood function, which usually requires more samples than features. Otherwise the models may be overfitted and are of high variances. In many modern applications such as multi-type cancer and document classification, this condition is often not satisfied as there have more features than samples in the data. A lot more parameters need to be learned as one feature corresponds to multiple parameters across multiple classes, resulting in a more complex model training and possible overfitting.

To overcome this situation, feature selection, which in many applications is of great value itself and makes the model more interpretable, has attracted much interest from the

research community. Various sparsity-inducing regularization methods have been developed and achieved great success in analyzing high dimensional data.

In applications such as cancer and text classification, prior knowledge is often available that there exists some intrinsic group structures among the features. As the structure of feature groups is known, incorporating this information in building a sparsity-inducing model could potentially not only lead to better models but also achieve sparsity on a larger scale. While it might be too restrictive to assume that all classes share the same structure of features, it is more reasonable to allow the class-specific structures of features and feature groups vary across classes. Moreover, as we are motivated by many real-world problems, including all features from the chosen groups in the model might overfit the model as well thereby within-group sparsity is desired.

To further motivate our work, we briefly discuss two exemplar applications in cancer and text classification. In cancer classification, genes are grouped into overlapping gene sets (pathways). Different cancers are regulated by different pathways, and within each pathway are regulated by a subset of genes [87, 108]. For another example, in document classification, different document classes are related to different topics, and each topic is represented by a set of different keywords. Successful identification of the relevant feature groups and features within each group is crucial for this classification task.

In this chapter, we propose a regularized multinomial logistic model, called class-conditional sparse overlapping group lasso (CCSOGL), to specifically incorporate the considerations in motivation. Our CCSOGL formulation has several contributions to the field:

- We incorporate information of feature groups in CCSOGL in the class-conditional

fashion. This flexibility makes CCSOGL capable of achieving class-specific sparsity pattern at the group level and further selecting relevant features within the group, which fits many applications well and enhances model performances and interpretability.

- We present a block coordinate descent algorithm solving CCSOGL efficiently.
- We evaluate the effectiveness and robustness of CCSOGL with benchmark datasets and compare CCSOGL with other state-of-art sparsity-inducing methods for multinomial classification.

2.2 Related Work

Lasso [144] and its variants [31] are among the regularization methods that induce sparsity. In cases that features are grouped according to prior knowledge, group Lasso (GLasso) [96, 160] extends Lasso by introducing the group structure and be capable of achieving sparsity at the group level. Sparse group Lasso (SGL) [148] develops group Lasso and further introduces within-group sparsity: it first selects feature groups; then within the selected groups, it selects features. To handle overlapping feature groups, overlapping group Lasso (OGL) [59] and sparse overlapping group Lasso (SOGL) [120] are proposed using feature duplication. Lasso, GLasso and SGL, first developed in regression and binary logistic model, were later generalized to the multinomial problem [128, 129, 148]. In the GLasso multinomial model, under the implicit assumption that all classes are related to the same set of features, parameter coefficients corresponding to the same feature are grouped. (That is, in multinomial GLasso, coefficients are grouped without the need of prior knowledge.) Multinomial SGL likewise achieves within group sparsity in addition to

group sparsity.

Figure 1 presents an illustrative example of sparsity pattern induced by different regularization methods, including (a)Lasso, (b)GLasso and (c)SGL and our new methods (d)CCOGL and (e)CCSOGL (Panel (d) is a special case of (e) in our formulation. See Section 2.3.2). In this figure, each heat-map represents a parameter coefficient matrix with each row corresponding to one feature and each column to one class; the small cyan rectangles represent the selected features. In panel (d), each long vertical rectangle in one class represents selected feature groups specific to that class (class-specific topics in text classification or pathways in cancer classification). Panel (e) extends (d) by selecting features within selected groups. The sparsity pattern in panel (e) is often of primary interest in real-world problems as in the motivation described above. Note that Panel (d) and (e) show sparsity patterns, at the feature group level, different from (b) and (c) ("vertical" vs. "horizontal"). In (d) and (e), CCOGL and CCSOGL explicitly uses prior knowledge about feature groups (for example, pathways in cancer) and allows group structures varying across all classes. In contrast, Panel (b) GLasso and (c) SGL do not use the prior knowledge and implicitly assume all classes relate to features from a same feature set.

2.3 Methods

Let $\{(x_i, y_i)\}_{i=1}^n$ represent the set of n samples, where $x_i \in \mathbf{R}^P$ is the P-dimensional input vector of features for the i-th sample, and y_i is the output. The design matrix X is organized as an $n \times P$ matrix. We first introduce the sparse overlapping group lasso (SOGL) in linear and binary logit model [120] and then formulate our CCSOGL as a penalty in the multinomial logistic regression.

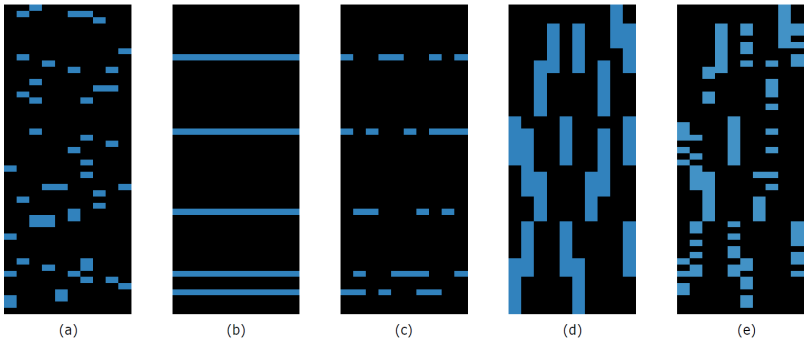

Figure 1: An illustrative example: sparsity pattern of coefficient matrix induced by different methods: (a) Lasso. (b) group lasso : coefficients corresponding to the same feature being grouped. (c) sparse group lasso, extending (b) by introducing within-group sparsity. (d) CCOGL and (e) CCSOGL represent our main contributions in this chapter. In (d), coefficients are grouped class-wise according to predefined (possibly overlapping) feature groups. (e) is an extension of (d), introducing within-group sparsity.

2.3.1 Sparse Overlapping Group Lasso.

Suppose that there is a group structure $G = \{G_1, \cdots, G_J\}$ among P features $\{f_1, \cdots, f_P\}$ that each feature f_p $(1 \leq p \leq P)$ is assigned to at least one group G_j $(1 \leq j \leq J)$. In linear regression and binary classification, let β_0 and $\beta = (\beta_1, \cdots, \beta_P)^T$ denote the intercept and coefficient vector respectively.

The key idea of SOGL [120] is to decompose the coefficient vector β into a sum of group-support vectors, denoted by $\omega_\beta = \{\omega^1, \cdots, \omega^J : \sum_{j=1}^{J} \omega^j = \beta\} \subset \mathbf{R}^P$. Each support vector ω^j satisfies a property that if $f_p \in G_j$, $\omega_p^j \in \mathbf{R}$, otherwise $\omega_p^j = 0$. For instance, in a simple case of 4 features $\{f_1, f_2, f_3, f_4\}$ and 3 groups $G_1 = \{f_1, f_3, f_4\}$, $G_2 = \{f_1, f_2\}$,

$G_3 = \{f_2, f_4\}$, β is decomposed as a sum of three group-support vectors:

$$\beta = \omega^1 + \omega^2 + \omega^3$$

$$G_1 : \omega^1 = (\omega_1^1, 0, \omega_3^1, \omega_4^1)^T$$

$$G_2 : \omega^2 = (\omega_1^2, \omega_2^2, 0, 0)^T$$

$$G_3 : \omega^3 = (0, \omega_2^3, 0, \omega_4^3)^T.$$

Based on this decomposition, the sparse overlapping group lasso is defined as

$$g(\beta) = \inf_{\omega_\beta} \sum_{j=1}^{J} \big(a||\omega^j||_1 + b||\omega^j||_2\big), \tag{2.1}$$

where $a > 0$ and $b \geq 0$ determine the trade-off between l_1 and l_2 norm. One key property of $g(\beta)$ is that it is a norm (*Lemma 4.1* in [120]), meaning that SOGL penalized linear and binary logit model are convex programs:

$$\min_{\beta_0, \beta} E(\beta_0, X\beta) + \lambda g(\beta), \tag{2.2}$$

where $E(\beta_0, \beta)$ is the square loss in linear regression or negative $\log$-likelihood in binary logit model, λ is the tuning parameter.

2.3.2 CCSOGL Classifier.

In the multi-classification problem of K classes, multinomial logistic regression models using *softmax* function calculate probabilities of multiple class memberships as below:

$$P(y = k|x) = \frac{\exp(g_k(x))}{\sum_{l=1}^{K} \exp(g_l(x))} \quad k = 1, \cdots, K, \tag{2.3}$$

where $g_k(x) = \beta_{k0} + x\beta_k$, $\beta_k = (\beta_{k1}, \cdots, \beta_{kP})^T \in \mathbf{R}^P$. The model parameters are represented by a pair (β_0, β) with $\beta = (\beta_{11}, \cdots, \beta_{1P}; \cdots; \beta_{K1}, \cdots, \beta_{KP}) \in \mathbf{R}^{KP}$, $\beta_0 = (\beta_{10}, \cdots, \beta_{K0})$. We say β_k is the k-th vector component of β.

For n samples $(x_1, y_1), \cdots, (x_n, y_n)$, the output $y_i = k$ is encoded as $(y_{i1}, \cdots, y_{iK})$ with $y_{ik} = 1$ and $y_{ih} = 0$ for $h \neq k$, and we write $p_{ik} = P(y_i = k|x_i)$. The (scaled) negative $\log$-likelihood function is:

$$\begin{aligned} L(\beta_0, \beta) &= -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \cdot \ln p_{ik} \\ &= -\frac{1}{n} \sum_{i=1}^{n} \Big[\sum_{k=1}^{K} y_{ik}(\beta_{k0} + x_i\beta_k) - \ln \sum_{l=1}^{K} \exp(\beta_{l0} + x_i\beta_l) \Big]. \end{aligned} \tag{2.4}$$

In high dimension problems ($P \gg n$), the (unregularized) maximum likelihood approach can lead to severe overfitting. Regularization of features is a popular choice to identify a small number of significant features for model interpretation and improvement of prediction stability. When the feature grouping information is available, along with the consideration that features and groups of features may vary across response classes, we formulate a new sparsity-pursuit penalty “class-conditional sparse overlapping group lasso" (CCSOGL).

Suppose that $G = \{G_1, \cdots, G_J\}$ is the group structure among features $\{f_1, \cdots, f_P\}$, the coefficient vector β is, based on G, decomposed as a sum of class-dependent group support vectors $\omega_\beta = \{\omega_k^j \in \mathbf{R}^{KP} : 1 \leq j \leq J, 1 \leq k \leq K, \beta = \sum_{j=1}^{J} \omega_k^j\}$. Each $\omega_k^j = (\omega_{k,1}^j; \cdots; \omega_{k,K}^j)$, where $\omega_{k,h}^j \in \mathbf{R}^P$ for $1 \leq h \leq K$ is the h-th vector component of ω_k^j, has the following property: (i) For $h \neq k$, $\omega_{k,h}^j = 0$; (ii) The k-th vector component $\omega_{k,k}^j$ of ω_k^j is a support vector of β_k for group G_j as in SOGL.

In other words, for the k-th class,

$$\beta_k = \sum_{j=1}^{J} \omega_{k,k}^j, \ 1 \leq k \leq K. \tag{2.5}$$

Based on this decomposition, CCSOGL is defined as:

$$h(\beta) = \inf_{\omega_\beta} \sum_{j=1}^{J} \sum_{k=1}^{K} (\alpha ||\omega_k^j||_1 + (1-\alpha)\sqrt{d_j}||\omega_k^j||_2), \tag{2.6}$$

where $0 \leq \alpha < 1$ controls tradeoff between l_1 and l_2 norm, d_j is the size of the j-th group G_j.

Lemma 2.1. *$h(\beta)$ is a norm. In particular, $h(\beta)$ is convex.*

Proof. See supplemental material. □

The CCSOGL estimator is given by the convex optimization problem:

$$\min_{\beta_0, \beta} L(\beta_0, \beta) + \lambda h(\beta). \tag{2.7}$$

2.3.3 Optimization Algorithm for Solving CCSOGL

Instead of solving the optimization problem (2.7), we use “feature duplication" method as in [59], [120] to reduce it to a non-overlapping convex problem in the expanded feature space:

$$\min_{\beta_0,\omega_\beta} S(\beta_0,\omega_\beta) := L(\beta_0,\omega_\beta) + \lambda \sum_{j=1}^{J}\sum_{k=1}^{K} \big(\alpha||\omega_k^j||_1 + (1-\alpha)\sqrt{d_j}||\omega_k^j||_2\big), \tag{2.8}$$

where, with a slight abuse of notation, $L(\beta_0,\omega_\beta)$ represents $L(\beta_0,\beta)$ with β being substituted by ω_β according to $\beta = \sum_{k,j}\omega_k^j$. Feature duplication can be seen by noticing for each vector component β_k of β, $X\beta_k = X\sum_{j=1}^{J}\omega_{k,k}^j$. For notational convenience, in the following section, we drop out those zero components in the support vector ω_k^j as they don't affect the objective function (and hence $\omega_k^j \in \mathbf{R}^{d_j}$). We also assume that overlapping features of each sample x_i have already been duplicated. Furthermore, we use $x_i^j \in \mathbf{R}^{d_j}$ to represent the sub-vector of the i-th sample x_i such that each component of x_i^j corresponds to the feature in th j-th group G_j respectively.

Problem (2.8) is a convex program, and the penalty term is block separable [145]:

$$\Omega(\omega_\beta) = \sum_{j=1}^{J}\sum_{k=1}^{K}(\alpha||\omega_k^j||_1 + (1-\alpha)\sqrt{d_j}||\omega_k^j||_2) = \sum_{j=1}^{J}\sum_{k=1}^{K}\Omega_k^j(\omega_k^j), \tag{2.9}$$

where $\Omega_k^j(\omega_k^j) = \alpha||\omega_k^j||_1 + (1-\alpha)\sqrt{d_j}||\omega_k^j||_2$. This implies that the block coordinate descent algorithm ([145], [156]) is well suited for this problem. In the algorithm, we cycle through the parameter blocks and each iteration minimizes a subproblem keeping all but the currently chosen parameter block fixed. In the following description of the algorithm,

$(\tilde{\beta}_0, \tilde{\omega}_\beta) = \{\tilde{\beta}_{k0}, \tilde{\omega}_k^j : 1 \leq k \leq K, 1 \leq j \leq J\}$ represents the numeric values learned in the previous update; $L(\beta_{k0})$ represents $L(\beta_0, \omega_\beta)$ as a function of β_{k0} with all coefficients being assigned with the current values except β_{k0}; $L(\omega_k^j)$, $S(\beta_{k0})$ and $S(\omega_k^j)$ are similar.

In the minimization for β_{k0} $(1 \leq k \leq K)$, as it is not penalized, we update β_{k0} using the Newton-Raphson formula:

$$\tilde{\beta}_{k0} \leftarrow \tilde{\beta}_{k0} - \frac{L'_{k0}}{L''_{k0}}, \tag{2.10}$$

where $L'_{k0} = \frac{1}{n}\sum_{i=1}^n(\tilde{p}_{ik} - y_{ik})$, $L''_{k0} = \frac{1}{n}\sum_{i=1}^n \tilde{p}_{ik}(1 - \tilde{p}_{ik})$, $\tilde{p}_{ik}$ is the probability calculated by (2.3) at the current value $(\tilde{\beta}_0, \tilde{\omega}_\beta)$, L'_{k0} and L''_{k0} denote the derivative of 1-st and 2-nd order of $L(\beta_{k0})$ respectively. With a proper initialization, the Newton method converges.

In updating the block ω_k^j $(1 \leq k \leq K, 1 \leq j \leq J)$ with other blocks holding fixed, an optimization subproblem is constructed in which the objective function $Q(\omega_k^j) : \mathbf{R}^{d_j} \to \mathbf{R}$ is a sum of what is called the "majorizing function" $M(\omega_k^j)$ of $L(\omega_k^j)$ and the corresponding penalty block $\lambda\Omega_k^j$ plus a constant.

More specifically,

$$M(\omega_k^j) = L(\tilde{\beta}_0, \tilde{\omega}_\beta) + (\omega_k^j - \tilde{\omega}_k^j)^T \nabla_k^j L(\tilde{\beta}_0, \tilde{\omega}_\beta) + \frac{1}{2t}||\omega_k^j - \tilde{\omega}_k^j||_2^2, \tag{2.11}$$

$$Q(\omega_k^j) = M(\omega_k^j) + \lambda\Omega_k^j(\omega_k^j) + \lambda C, \tag{2.12}$$

$$\tilde{\omega}_k^j \leftarrow \arg\min_{\omega_k^j} Q(\omega_k^j), \tag{2.13}$$

where $\nabla_k^j L = \frac{\partial L}{\partial \omega_k^j} = \frac{1}{n}\sum_{i=1}^n (p_{ik} - y_{ik}) x_i^{jT}$, t is a properly selected constant, $C = \Omega(\tilde{\omega}_\beta) - \Omega_k^j(\tilde{\omega}_k^j)$. The majorizing function $M(\omega_k^j)$ comes from the "majorize-minimization" algorithm [56] with a nice property that if t is chosen small enough, the third term $\frac{1}{2t}||\omega_k^j - \tilde{\omega}_k^j||_2^2$

will dominate the Hessian term in the Taylor expansion of $L(\omega_k^j)$. As a consequence, the following inequality holds for all $\omega_k^j \in \mathbf{R}^{d_j}$:

$$L(\omega_k^j) \leq M(\omega_k^j). \tag{2.14}$$

Adding the penalty term on $M(\omega_k^j)$ leads to a majorizing function $Q(\omega_k^j)$ for the objective $S(\omega_k^j)$ in (2.8). That is, for all $\omega_k^j \in \mathbf{R}^{d_j}$,

$$S(\omega_k^j) \leq Q(\omega_k^j). \tag{2.15}$$

Lemma 2.2. *Cyclic block coordinate descent algorithm for CCSOGL converges.*

Proof. See supplemental material. □

By completing the square in $M(\omega_k^j)$, minimizing $Q(\omega_k^j)$ is equivalent to minimizing:

$$R(\omega_k^j) = \frac{1}{2\lambda t}||\omega_k^j - [\tilde{\omega}_k^j - t\nabla_k^j L(\tilde{\beta}_0, \tilde{\omega}_\beta)]||_2^2 + \Omega_k^j(\omega_k^j). \tag{2.16}$$

Note that $R(\omega_k^j)$ is strictly convex, so the optimal minimizer is characterized by the first order condition [100]. This results in the following lemma:

Lemma 2.3. *The minimizer $\omega_k^{j\star}$ for (2.16) is given by the following update rule:*
if $||T_\alpha(\frac{\tilde{\omega}_k^j - t\nabla_k^j L(\tilde{\beta}_0, \tilde{\omega}_\beta)}{\lambda t})||_2 \leq (1-\alpha)\sqrt{d_j}$, $\omega_k^{j\star} = 0$,
if $||T_\alpha(\frac{\tilde{\omega}_k^j - t\nabla_k^j L(\tilde{\beta}_0, \tilde{\omega}_\beta)}{\lambda t})||_2 > (1-\alpha)\sqrt{d_j}$,

$$\omega_k^{j\star} = \Big[1 - \frac{(1-\alpha)\sqrt{d_j}\lambda t}{||T_{\alpha\lambda t}(\tilde{\omega}_k^j - t\nabla_k^j L(\tilde{\beta}_0, \tilde{\omega}_\beta))||_2}\Big] \cdot T_{\alpha\lambda t}(\tilde{\omega}_k^j - t\nabla_k^j L(\tilde{\beta}_0, \tilde{\omega}_\beta)), \tag{2.17}$$

Algorithm 1 Cyclic Block Coordinate Descent for CCSOGL

Initialize (β_0, ω_β)
repeat
  for $k = 1$ to K
    $\beta_{k0} \leftarrow \arg\min L(\beta_{k0})$, using Newton-Raphson formula (2.10)
    for $j = 1$ to J
      if $||T_\alpha(\frac{\tilde{\omega}_k^j - t\nabla_k^j L(\tilde{\beta}_0, \tilde{\omega}_\beta)}{\lambda t})||_2 \leq (1-\alpha)\sqrt{d_j}$
        $\tilde{\omega}_k^j \leftarrow 0$
      else
        $\tilde{\omega}_k^j \leftarrow \Big[1 - \frac{(1-\alpha)\sqrt{d_j}\lambda t}{||T_{\alpha\lambda t}(\tilde{\omega}_k^j - t\nabla_k^j L(\tilde{\beta}_0, \tilde{\omega}_\beta))||_2}\Big] \cdot T_{\alpha\lambda t}(\tilde{\omega}_k^j - t\nabla_k^j L(\tilde{\beta}_0, \tilde{\omega}_\beta))$
until converge

where $T_v(x) = \big(S(x_1, v), \cdots, S(x_{d_j}, v)\big)$ $(x \in \boldsymbol{R}^{d_j})$ *and* $S(u, v) = \textit{sign}(u)\max\{|u| - v, 0\}$ $(u \in \boldsymbol{R}, v \geq 0)$ *is the soft-thresholding operator.*

Proof. See supplemental material. □

From Lemma 2.3, we see that our CCSOGL model first select feature groups; within the selected feature groups, CCSOGL performs feature selection. As feature groups are selected class-wise, CCSOGL can indeed achieve the sparsity pattern shown in Panel (e) of Figure 1.

Integrating all of above leads to Algorithm 1 for solving our CCSOGL penalized multinomial logistic model.

2.4 Application

In this section, we evaluate and validate our method and compare its performance with selected competing classification methods using several publicly available datasets. Tested methods include Lasso, GLasso, $l1$-regularized $l2$-loss SVM and CCSOGL. The Lasso and GLasso were implemented using the R package *glmnet* [31]. L1-regularized SVM (L2-loss) was implemented in R package *LiblineaR*. The CCSOGL was implemented in C++ in house

interfacing with R through the R package *Rcpp* and *RcppArmadillo* [27].

2.4.1 Data Description.

We first used three gene expression datasets to evaluate CCSOGL. The details of datasets are as follows:

- *NCI-60* contains gene expression levels from 60 cell lines with 9 different types of cancer: 6 leukemia, 8 melanoma, 9 non-small-cell lung carcinoma, 7 colon, 6 central nervous system, 8 renal, 8 breast, 2 prostate and 6 ovarian. We removed samples of prostate cancer due to very small class size, resulting in 58 samples from 8 classes remain in analysis. More details for NCI60 can be found in [133]. The dataset is available from *http://www.broadinstitute.org/mpr/NCI60/*.

- *Brain Cancer* consists of gene expression profiles from 42 patients with different brain cancer types of the central nervous system. The samples are divided into 5 classes: 8 primitive neuroectodermal tumors (PNET), 10 atypical teratoid/rhabdoid tumors (AT/RT), 10 medulloblastomas, 10 malignant gliomas and 4 human cerebella. See [113] for more information. The data can be downloaded from *http://www.broadinstitute.org /mpr/publications/pro- jects/CNS/*.

- *Breast Cancer Subclass* contains gene expression values for 198 samples in 5 breast cancer subclasses: 30 basal-like, 11 HER2, 64 luminal A, 90 luminal B and 3 normal breast-like. 3 samples with normal breast-like were removed from analysis. More background information is available in [21]. This dataset can be downloaded form the Gene Expression Omnibus with accession number GSE7390.

Datasets were preprocessed before analysis. We used gene set collections from MSigDB

Table 1: Summary of datasets used in our analysis. In the table, K, n, p and J represent number of classes, number of samples, number of features and number of feature sets, respectively; “Collection" represents the gene set collection used from MSigDB.

Dataset	K	n	p	J	Collection
NCI60	8	58	2654	103	C4 CM
Brain	5	42	2035	111	C5
Breast	4	195	3582	43	C6

database [135] as our predefined feature groups in CCSOGL. Not all gene sets in one collection were used in our analysis. We first applied Gene Set Enrichment Analysis (GSEA) [135] to filter out irrelevant gene sets using a cutoff of *p*-value $\geq 5\%$. Using irrelevant gene sets will introduce too much noise in our model. Further, genes in the raw data that are not present in the selected gene sets were removed from our analysis. Expression values for each gene were normalized using $x' = (x - \min(x))/(\max(x) - \min(x))$ for numerical convenience. Table 1 provides details of datasets used in our experiments.

2.4.2 Performance Evaluation.

Performances of different sparsity-induced methods were evaluated following the conventional way of performing external cross-validation as done in the closely related works in literature, such as [119, 148]. Since classes in the used datasets are unbalanced, solely estimating prediction errors favors models with better predictive performances on the dominant classes and may obscure model predictive behaviors. Hence, in addition to estimating prediction errors, macro F1-score, treating each class equally regardless of sample size, was also used. To obtain stable estimated prediction errors as well as calculate macro F1-scores, we used (stratified) 4-fold cross-validation and repeated this procedure 10 times.

We first estimated prediction error with cross-validation error (CVE). Table 2 reports the

4-fold cross validation errors for different models. We see that the CCSOGL outperforms Lasso and group Lasso (GLasso) due to incorporation of feature group information. In the multi-cancer classification problem, it is translated into the assumption that different cancers are regulated by different genes and pathways. To further validate performance of CCSOGL, we performed one-sided two sample t-test on the cross-validation error between CCSOGL and each of Lasso, GLasso and SVM: $H_0 : \text{CVE}_{\text{CCSOGL}} = \text{CVE}_{\text{other}}$ *vs.* $H_1 : \text{CVE}_{\text{CCSOGL}} < \text{CVE}_{\text{other}}$. Table 3 provides the resultant statistics for three datasets, showing significantly that CCSOGL performs better than other models. Incorporating the prior knowledge about feature groups and allowing class-specific dependencies of features and feature groups in building the models indeed improve the classifier performance as demonstrated by the experiments.

Table 2: Average of 4-fold cross validation error (CVE) for different methods over 10 runs (along with their standard deviations). The best performance is bold faced.

Dataset	Lasso	GLasso	SVM	CCSOGL
NCI60	0.448 (0.042)	0.398 (0.027)	0.461 (0.024)	**0.384** (0.031)
Brain	0.262 (0.036)	0.246 (0.043)	0.217 (0.046)	**0.175** (0.033)
Breast	0.243 (0.021)	0.236 (0.019)	0.240 (0.018)	**0.205** (0.013)

Table 3: p-values (along with t statistics) on cross-validation error (CVE) for one-sided two sample t-test H_0: CVE of CCSOGL = CVE of other method *vs.* H_1: CVE of CCSOGL $<$ CVE of other method.

CCSOGL *vs.*	Lasso	GLasso	SVM
NCI60	$\leq 0.001^*$	0.138	$\leq 0.001^*$
Brain	$\leq 0.001^*$	$\leq 0.001^*$	0.016^*
Breast	$\leq 0.001^*$	$\leq 0.001^*$	$\leq 0.001^*$

We also used the 4-fold macro F1-score as an evaluation metric for each method. In multi-class classification, macro F1-score [147] is a performance measurement for classi-

fiers that is calculated the average of per-class F1-scores. F1-score is the harmonic mean of *precision* and *recall* for each classification, i.e., $F_1 = 2 \times \frac{precision \times recall}{precision + recall}$, where *precision* is the ratio of samples which are classified as positive are correct and *recall* is the ratio of positive samples that are correctly classified. Here, we calculated the 4-fold macro F1-score as the average of the macro F1-scores in the 4-fold cross validation (one F1-score for each fold). Again, we reported the average of cross validation macro F1-scores over the 10 runs.

Macro F1-score weighs prediction performance for each class equally. Hence, macro F1-score will not be dominated by performances of classifiers on classes with larger sample size, yet be sensitive for performances on classes with smaller sample size. This implies that macro F1-score provides more insights for model evaluation, especially for applications in unbalanced data. As we report average of macro F1-score on multiple runs, models with less variance of macro F1-score indicate more consistency and robustness.

Table 4 presents average of 4-fold cross validation macro F1-scores (along with standard deviations) for different methods on 10 repetitions. As shown in the table, CCSOGL enjoys the smallest variance for three datasets without compromising macro F1-scores, indicating a more robust performance over the competing methods.

Table 4: Average of 4-fold macro F1-score for different methods over 10 runs (along with their standard deviations). Smallest standard deviation is starred and highest macro F1-score is bold faced.

Dataset	Lasso	GLasso	SVM	CCSOGL
NCI60	0.508 (0.042)	0.558 (0.038)	0.485 (0.035)	**0.565** $(0.030)^*$
Brain	0.710 (0.050)	0.699 (0.051)	0.749 (0.037)	**0.782** $(0.027)^*$
Breast	0.622 (0.040)	0.622 (0.029)	**0.639** (0.031)	0.623 $(0.009)^*$

Table 5: Results for class-specific feature group selection: number of selected groups for each class, number of selected groups unique to each class and proportion of the selected groups among all groups.

Dataset	NCI60	Brain	Breast
# Group	{8,7,4,6,3,6,4,7}	{6,6,5,1,8}	{3,10,3,8}
# Unique	{7,3,1,4,1,4,3,4}	{2,2,3,1,3}	{2,4,0,3}
Proportion	0.35	0.16	0.37

2.4.3 Class-specific Feature Group Selection

CCSOGL incorporates heterogeneous structures of feature groups across classes as described in the motivation. The incorporation brings CCSOGL an advantage over other competing method that CCSOGL is able to achieve class-specific sparsity at group level.

Table 5 provides results of CCSOGL for class-wise selecting feature groups. We chose tuning parameters from 4-fold cross-validation and then trained model on the entire datasets. Selected feature groups are obtained form the trained CCSOGL model. From the table, we see that CCSOGL not only achieves group selection, but also uncovers different sparsity pattern across classes: some feature groups are identified that are unique to each class while some feature groups are shared by several classes. This flexibility in group selection for CCSOGL makes the selected model much easier to interpret.

2.5 Conclusion

In this chapter, we have proposed a regularized method so-called CCSOGL for multinomial logistic regression and compared the performance of CCSOGL with other state-of-art sparsity-inducing methods on several benchmark datasets. The CCSOGL method can perform class-dependent selection of feature groups and features to achieve a superior performance in multinomial classification. This flexibility of CCSOGL not only increases predictive accuracy and model robustness, but also potentially discovers the functional feature

groups in the classification, which may provides insight on the related field. Our model is efficiently and accurately solved by the cyclic block coordinate descent algorithm. Future development includes generalization of our methods to accommodate multiple types of features and extension to classification problems with ordinal class labels.

CHAPTER 3 FEATURE SELECTION FOR FINITE MIXTURE OF REGRESSION

Finite mixture of Gaussian regression (FMR) is a widely-used modeling technique in supervised learning problems. In cases where the number of features is large, feature selection is desirable to enhance model interpretability and to avoid overfitting. In this chapter, we propose a robust feature selection method via $l_{2,1}$-norm penalized maximum likelihood estimation (MLE) in FMR, with extension to sparse $l_{2,1}$ penalty by combining l_1-norm with $l_{2,1}$-norm for increasing flexibility. To solve the non-convex and non-smooth problem of (sparse) penalized MLE in FMR, we develop an new EM-based algorithm for numerical optimization, with combination of block coordinate descent and majorizing-minimization scheme in M-step. We finally apply our method in six simulations and one real dataset to demonstrate its superior performance.

3.1 Introduction

Finite mixture of regression (FMR) is a flexible modeling technique for supervised learning problems. It extends uni-modal assumption of Generalized Linear Model (GLM) to multi-modal cases. This method is widely used in various applications such as biology, economics, and engineering [25, 50, 66]. Among these applications, many problems arise with high dimensionality yet the sample size is relatively small. Fitting FMR directly with maximum likelihood approach not only results in severe overfitting and poor generalization performance, but also makes the model hard to interpret. One viable approach to address those two issues, i.e., variance reduction and feature selection, is to fit FMR with sparsity-inducing penalty.

Various sparsity-inducing methods in GLM were extensively developed and successfully applied to classification and uni-modal regression problems. [144, 160, 96, 59, 120]. How-

ever, it is not the case for multi-modal regression problems, especially in high dimension. Initial works by [65, 132] applied Lasso-typed (l_1-norm of model parameters) methods to finite mixture of Gaussian regression. l_1 penalty is capable of finding heterogeneous feature structure across mixture components. In FMR models, parameters of features are naturally structured, i.e., multiple parameters (one parameter from each mixture component) corresponding to one feature are grouped. The l_1 penalty, however, performs feature selection for each mixture component individually, hence it misses the similarity and relatedness among different mixture components. As we may expect that different mixture components of FMR share some common features, incorporating the group structure in modeling is beneficial. To this end, we seek a method that utilizes the grouping information in FMR as well as enjoy the flexibility of allowing features to be component-dependent.

We propose a novel penalized finite mixture of Gaussian regression model with structured feature selection that explicitly incorporates the information of parameter grouping via $l_{2,1}$-norm. $l_{2,1}$ penalty has one appealing property that it performs feature selection at the (parameter) group level, encouraging the same sparsity pattern across all mixture components. Our approach is more robust to noisy features and model noise as demonstrated in simulation studies. Moreover, we further extend $l_{2,1}$ to flexible sparse $l_{2,1}$ penalty by combing with l_1-norm. With incorporation of l_1-norm, sparse $l_{2,1}$ penalized FMR allows heterogeneous feature structures across mixture components while possesses robustness of $l_{2,1}$ penalty.

The resultant penalized MLE in FMR is formulated as a non-convex optimization problem. The standard approach for penalized FMR is EM-type algorithm; the non-smoothness of sparse $l_{2,1}$ penalty, however, poses substantial challenges in the M-step of EM algorithm.

Inspired by a re-parametrization trick in [132], we combine block coordinate descent algorithm and majorizing-minimization scheme for numerical optimization in the M-step, with efficient closed-form updates in each iteration. Finally, we apply our method to evaluate its performance using simulation and real data sets.

3.2 Related Work

Finite mixture of regression models can be viewed as an extension of Generalized Linear Model (GLM) and have been extensively studied [118, 57, 95, 8, 138]. In applications of FMR, “mixtures of experts" and its extension “hierarchical mixtures of experts", are widely used and have achieved great success in various applications [60, 63, 62]. See [95] for a comprehensive review for finite mixture models. While original FMR was developed mainly for low-dimensional data, as high dimensional data is common in recent years, fitting FMR directly is ill-suited due to the curse of dimensionality.

In terms of penalized methods, various sparsity-inducing penalties have been proposed for high dimensional data. They often improve model performances due to bias-variance trade-off. In GLM, the Lasso [144] is a penalized method by adding l_1 penalty in the maximum likelihood, for simultaneous parameter estimation and feature selection. In applications where features can be grouped by prior knowledge (for example, genes (features) are grouped into pathways), Group Lasso [160, 96] and overlapping Group Lasso [59] extend Lasso for feature selection at the group level; Sparse Group Lasso [129, 120] improves Group Lasso by further inducing within-group sparsity.

In the multi-task learning and multi-label classification, $l_{2,1}$ norm that is of the same functional form with Group Lasso is widely used as penalty [86, 105, 42, 128, 148]. $l_{2,1}$ norm is applied mainly for capturing the similarity and relatedness in feature structures

among different tasks or labels.

In the context of FMR, Khalili et al. [65] and Städler et al. [132] are the pioneer works that combine FMR and feature selection via sparsity-inducing penalty. Both works consider l_1-norm penalized FMR and solve the penalized MLE using EM-algorithm, with challenges in the M-step due to the non-smoothness of l_1 penalty. In Khalili et al. [65], a differentiable approximation of l_1 penalty is used and M-step is numerically optimized by solving the system of equations given by the first order conditions. Städler et al. [132] differs from Khalili et al. [65] that a re-parameterization strategy is used, resulting in a convex problem in the M-step. This re-parameterization is beneficial as efficient algorithms such as coordinate descent in convex optimization can be applied. As mentioned in introduction, l_1 penalized FMR misses the grouping information of parameters in FMR, our approach of sparse $l_{2,1}$ penalized FMR improves this limit and extends l_1 penalized FMR, leading to group structured sparsity while keeping the ability of selecting component-dependent features.

3.3 Method

Notations: Scalars are denoted as lower case letters, vectors and matrices as bold letters with vectors viewed as one-column matrices. $||\cdot||_1$ represents the l_1-norm of a vector or a matrix, $||\cdot||_2$ the l_2-norm of a vector, $||\cdot||_F$ the Frobinius norm of a matrix, $||\cdot||_{2,1}$ the $l_{2,1}$-norm of a matrix. $\mathbf{R}_+$ is the set of positive reals. $\boldsymbol{v}'$ is the transpose of a vector $\boldsymbol{v}$ (or matrix). For a matrix $\boldsymbol{M}$, $\boldsymbol{M}_{i\cdot}$ and $\boldsymbol{M}_{\cdot j}$ are used to represent the i-th row and the j-th column respectively.

3.3.1 Finite Mixture of Regression

Let $\{(\boldsymbol{x}_i, y_i), i = 1, \cdots, n\}$ be the set of independent observations, where $\boldsymbol{x}_i = (x_{i1}, \cdots, x_{ip})' \in \mathbf{R}^p$ is the p-dimensional vector of features for the i-th observation, and $y_i \in \mathbf{R}$ is the response. $\boldsymbol{X}$ represents the $n \times p$ design matrix and $\boldsymbol{Y}$ the vector of observation responses. The finite mixture of regression model (with k mixture components) is given as follows:

$$y|x \sim \pi_1 \mathcal{N}(\beta_{01} + \boldsymbol{x}'\boldsymbol{\beta}_1, \sigma_1^2) + \cdots + \pi_k \mathcal{N}(\beta_{0k} + \boldsymbol{x}'\boldsymbol{\beta}_k, \sigma_k^2),$$

where π_j represents the mixture probability, σ_j ($\sigma_j > 0$) the standard deviation, $(\beta_{0j}, \boldsymbol{\beta_j})$ the linear coefficients for the j-th Gaussian component $\mathcal{N}(\beta_{0j} + \boldsymbol{x}'\boldsymbol{\beta}_j, \sigma_j^2)$ ($1 \leq j \leq k$). π_j's satisfy $\pi_j > 0$ and $\sum_{j=1}^k \pi_j = 1$.

The finite mixture of regression model is studied from the maximum likelihood approach. The parameter of FMR,

$$\Theta = (\pi_1, \cdots, \pi_k, \beta_{01}, \cdots, \beta_{0k}, \boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_k, \sigma_1, \cdots, \sigma_k),$$

is estimated by minimizing the (scaled) negative log-likelihood:

$$L(\Theta) = -\frac{1}{n}\sum_{i=1}^{n} \log\Big(\sum_{j=1}^{k} \frac{\pi_j}{\sqrt{2\pi}\sigma_j} \exp\big(-\frac{(y_i - \beta_{0j} - \boldsymbol{x}_i'\boldsymbol{\beta}_j)^2}{2\sigma_j^2}\big)\Big), \tag{3.1}$$

with the constraint $\sum_{j=1}^k \pi_j = 1$, $\sigma_j > 0$.

3.3.2 Feature Selection in $l_{2,1}$-norm Penalized FMR

Assume that for the j-th ($1 \leq j \leq k$) mixture component $\boldsymbol{\beta}_j = (\beta_{1j}, \cdots, \beta_{pj})'$, and write $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_k)$ as a $(p \times k)$ matrix of parameters. Let $\boldsymbol{\beta}_{l\cdot} = (\beta_{l1}, \cdots, \beta_{lk}) \in \mathbf{R}^k$ is the

m-th row of $\boldsymbol{\beta}$, corresponding to the l-th feature in the finite mixture model. With this correspondence, $\boldsymbol{\beta}$ can be decomposed into p such parameter groups.

The $l_{2,1}$-norm of $\boldsymbol{\beta}$ is defined as:

$$||\boldsymbol{\beta}||_{2,1} = \sum_{l=1}^{p} \sqrt{\sum_{j=1}^{k} \beta_{lj}^2} = \sum_{l=1}^{p} ||\boldsymbol{\beta}_{l\cdot}||_2. \tag{3.2}$$

Using $l_{2,1}$-norm as a penalty, the penalized MLE for FMR is obtained by solving:

$$\arg\min_{\Theta} L(\Theta) + \lambda||\boldsymbol{\beta}||_{2,1}, \tag{3.3}$$

where $\lambda > 0$ is a tuning parameter.

The $l_{2,1}$ penalty selects features at the group level: with a large λ, one feature is unselected by dropping out the corresponding whole parameter group. For the selected parameter group in $l_{2,1}$, every element is non-zero. However, this property seems restrictive when the component-dependent features in FMR may exist. To this end, we further propose the sparse $l_{2,1}$ penalty in FMR by incorporating l_1-norm:

$$P(\boldsymbol{\beta}) = (1-\alpha)\sqrt{k}||\boldsymbol{\beta}||_{2,1} + \alpha||\boldsymbol{\beta}||_1 \tag{3.4}$$

where $\alpha \in [0, 1]$ is a trade-off between l_1 and l_2 norm. $\sqrt{k}$ is used for a balance between l_1 and l_2 penalty. Notice that if $\alpha = 1$, (3.4) is exactly the l_1 penalty; while $\alpha = 0$, (3.4) is deduced to $l_{2,1}$ penalty.

With sparse $l_{2,1}$ penalty, (3.3) can be extended to

$$\arg\min_{\Theta} L(\Theta) + \lambda P(\boldsymbol{\beta}), \tag{3.5}$$

with constraints $\Theta \in \mathbf{R}_+^k \times \mathbf{R}^{k(p+1)} \times \mathbf{R}_+^k$, $\sum_{j=1}^k \pi_j = 1$.

3.3.3 Re-parameterization

The general framework for maximum likelihood approach in unpenalized FMR is to use EM-algorithm. In the penalized cases, existing methods [132, 65] adopt EM approach with optimizing the expectation of penalized complete log-likelihood. With the conventional parameterization Θ, the subproblem of optimization in the M-step is a challenging task due to non-convexity of the negative expected complete log-likelihood and non-smoothness of the sparse $l_{2,1}$ penalty. However, the non-convexity can be tackled by a re-parameterization of model parameters [132], resulting in a convex optimization problem in the M-step. The same trick can be applied in the sparse $l_{2,1}$ penalized FMR. Although the non-smoothness induced by $l_{2,1}$- and l_1-norm still exists, as we show in the next section, re-parametrization is sufficient for an effective optimization scheme in the M-step of EM algorithm.

In FMR, the negative maximum likelihood function (3.1) can be re-parameterized, for $j = 1, \cdots, k$, as follows:

$$\tau_j = \sigma_j^{-1}, \quad \eta_{0j} = \beta_{0j}/\sigma_j, \quad \boldsymbol{\eta}_j = \boldsymbol{\beta}_j/\sigma_j.$$

Let $\Phi = (\pi_1, \cdots, \pi_k, \eta_{01}, \cdots, \eta_{0k}, \boldsymbol{\eta}_1, \cdots, \boldsymbol{\eta}_k, \tau_1, \cdots, \tau_k)$. The re-parameterization yields an

one-to-one mapping from Θ to Φ. Hence, (3.1) can be rewritten as:

$$L(\Phi) = -\frac{1}{n}\sum_{i=1}^{n}\log\Big(\sum_{j=1}^{k}\pi_j\frac{\tau_j}{\sqrt{2\pi}}\exp\big(-\frac{(\tau_j y_i - \eta_{0j} - \boldsymbol{x}_i'\boldsymbol{\eta}_j)^2}{2}\big)\Big). \tag{3.6}$$

Under the re-parametrization, the sparse $l_{2,1}$ penalized MLE Φ^* for FMR is given by the following optimization problem:

$$\Phi^* = \arg\min_{\Phi} L(\Phi) + \lambda P(\boldsymbol{\eta}), \tag{3.7}$$

where $\Phi \in \mathbf{R}_+^k \times \mathbf{R}^{k(p+1)} \times \mathbf{R}_+^k$, $\sum_{j=1}^k \pi_j = 1$. λ and α are the tuning parameters. Notice that η_{0j}'s are not penalized.

3.4 Non-convex Optimization

In this section, we present a EM-based algorithm for numerically optimizing sparse $l_{2,1}$ penalized FMR.

In the following sections, for notational ease, we have suppressed η_{0j} into $\boldsymbol{x}'\boldsymbol{\eta}_j$ except in (2b) and (2c) of M-step. $\langle\cdot,\cdot\rangle$ represents dot product and $\odot$ element-wise product for two matrices of the same dimension.

Let $l_c(\Phi)$ be the complete log-likelihood function:

$$l_c(\Phi) = \sum_{i=1}^{n}\sum_{j=1}^{k}\Big[z_{ij}\log\Big(\frac{\tau_j}{\sqrt{2\pi}}\exp\big(-\frac{(\tau_j y_i - \boldsymbol{x}_i'\boldsymbol{\eta}_j)^2}{2}\big)\Big) + z_{ij}\log\pi_j\Big],$$

where $\{z_{ij} : j = 1, \cdots, k\}$ is the latent membership vector for the i-th observation belonging to which mixture component: $z_{ij} = 1$ and $z_{ir} = 0$ for $r \neq j$ indicating the i-th observation belongs to the j-th component. The sparse $l_{2,1}$ penalized (scaled) negative

complete log-likelihood is then

$$L_c(\Phi) = -\frac{1}{n} l_c(\Phi) + \lambda P(\boldsymbol{\eta}).$$

The EM algorithm minimizes $L_c(\Phi)$ by iterating between the E- and M-step as follows. Assume that $\Phi^{(m)}$ is the current parameter estimate:

E step. Given the current estimate $\Phi^{(m)}$, the E step computes the conditional expectation $Q(\Phi)$ of $L_c(\Phi)$ with respect to the latent variables z_{ij}'s. This is equivalent to compute the expectation $w_{ij}^{(m)}$ of z_{ij} for $i = 1, \cdots, n$ and $j = 1, \cdots, k$:

$$w_{ij}^{(m)} = \frac{\pi_j^{(m)} \tau_j^{(m)} \exp\big(-\frac{1}{2}(\tau_j^{(m)} y_i - \boldsymbol{x}_i' \boldsymbol{\eta}_j^{(m)})^2\big)}{\sum_{r=1}^{k} \pi_r^{(m)} \tau_r^{(m)} \exp\big(-\frac{1}{2}(\tau_r^{(m)} y_i - \boldsymbol{x}_i' \eta_r^{(m)})^2\big)}. \tag{3.8}$$

$$\begin{aligned} Q(\Phi) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij}^{(m)} \Big[\log\Big(\frac{\tau_j}{\sqrt{2\pi}} \exp\big(-\frac{(\tau_j y_i - \boldsymbol{x}_i' \boldsymbol{\eta}_j)^2}{2}\big)\Big) \\ + \log \pi_j \Big] + \lambda\big((1-\alpha) \sum_{l=1}^{p} \sqrt{k} ||\boldsymbol{\eta}_{l\cdot}||_2 + \alpha \sum_{l=1}^{p} ||\boldsymbol{\eta}_{l\cdot}||_1\big). \end{aligned}$$

M step. The update $\Phi^{(m+1)}$ is obtained by minimizing $Q(\Phi)$ with respect to Φ, which is further equivalent to solve two independent minimization problems with respect to $(\pi_1, \cdots, \pi_k)$ and $(\eta_{01}, \cdots, \eta_{0k}, \boldsymbol{\eta}_1, \cdots, \boldsymbol{\eta}_k, \tau_1, \cdots, \tau_k)$ respectively.

(1) Minimization with respect to $(\pi_1, \cdots, \pi_k)$. This is the same with mixture probability update in the usual EM algorithm:

$$\pi_j^{(m+1)} = \frac{1}{n} \sum_{i=1}^{n} w_{ij}^{(m)}, \quad j = 1, \cdots, k. \tag{3.9}$$

(2) Update $\Phi_{-\pi} = (\eta_{01}, \cdots, \eta_{0k}, \boldsymbol{\eta}_1, \cdots, \boldsymbol{\eta}_k, \tau_1, \cdots, \tau_k)$. After simplification of $Q(\Phi)$, we equivalently solve the following convex problem (each summand is convex):

$$\begin{aligned}
\Phi_{-\pi}^{(m+1)} &= \min_{\Phi_{-\pi}} S(\Phi_{-\pi}) \\
S(\Phi_{-\pi}) &= -\frac{1}{n}\sum_{j=1}^{k}\big(\sum_{i=1}^{n} w_{ij}^{(m)}\big)\log\tau_j \\
&\quad + \frac{1}{n}\sum_{j=1}^{k}\sum_{i=1}^{n}\frac{w_{ij}^{(m)}}{2}(\tau_j y_i - \boldsymbol{x}_i'\boldsymbol{\eta}_j)^2 \\
&\quad + \lambda\big((1-\alpha)\sum_{l=1}^{p}\sqrt{k}||\boldsymbol{\eta}_{l\cdot}||_2 + \alpha\sum_{l=1}^{p}||\boldsymbol{\eta}_{l\cdot}||_1\big).
\end{aligned} \tag{3.10}$$

The sparse $l_{2,1}$ penalty is separable between blocks of $\eta_{m\cdot}$'s [146], implying block coordinate algorithm is well suited for minimizing $S(\Phi_{-\pi})$.

In the block coordinate descent, we cyclically update each parameter (or parameter blocks) by approximately minimizing $S(\Phi_{-\pi})$, holding all except the current parameter fixed until convergence. This update strategy leads to the updates as follows. As a side note, Eq. (3.10) can be initialized by the current estimate $\Phi_{-\pi}^{(m)}$. For notational simplicity, we drop the iteration index of $\Phi_{-\pi}$ and use $\tilde{\Phi}_{-\pi} = (\tilde{\eta}_{01}, \cdots, \tilde{\eta}_{0k}, \tilde{\boldsymbol{\eta}}_1, \cdots, \tilde{\boldsymbol{\eta}}_k, \tilde{\tau}_1, \cdots, \tilde{\tau}_k)$ to represent the current value for $\Phi_{-\pi}$ in optimizing Eq. (3.10).

(2a) Update τ_j.

Since τ_j is not penalized and $S(\Phi_{-\pi})$ be differentiable with respect to τ_j, first order condition $\partial S(\Phi_{-\pi})/\partial\tau_j = 0$ results in the update for $j = 1, \cdots, k$:

$$\tilde{\tau}_j \leftarrow \frac{\langle \boldsymbol{W}_{\cdot j}^{(m)}, \boldsymbol{Y}\odot\hat{\boldsymbol{Y}}\rangle + \sqrt{\langle \boldsymbol{W}_{\cdot j}^{(m)}, \boldsymbol{Y}\odot\hat{\boldsymbol{Y}}\rangle^2 + 4s_j\langle \boldsymbol{W}_{\cdot j}^{(m)}, \boldsymbol{Y}\odot\boldsymbol{Y}\rangle}}{2\langle \boldsymbol{W}_{\cdot j}^{(m)}, \boldsymbol{Y}\odot\boldsymbol{Y}\rangle}, \tag{3.11}$$

where $\boldsymbol{W}_{\cdot j}^{(m)} = (w_{1j}^{(m)}, \cdots, w_{nj}^{(m)})'$, $\boldsymbol{Y} = (y_1, \cdots, y_n)'$, $\hat{\boldsymbol{Y}} = (\boldsymbol{x}_1'\tilde{\boldsymbol{\eta}}_j, \cdots, \boldsymbol{x}_n'\tilde{\boldsymbol{\eta}}_j)'$ and $s_j = \sum_{i=1}^n w_{ij}^{(m)}$.

(2b) Update η_{0j}.

Due to differentiability of $S(\Phi_{-\pi})$ respect to η_{0j}, the first order condition $\partial S(\Phi_{-\pi})/\partial\eta_{0j} = 0$ gives the minimizer

$$\tilde{\eta}_{0j} \leftarrow \frac{\langle \boldsymbol{W}_{\cdot j}^{(m)}, \hat{\boldsymbol{H}}_{0j} \rangle}{s_j}, \quad j = 1, \cdots, k, \tag{3.12}$$

where $\hat{\boldsymbol{H}}_{0j} = (\tilde{\tau}_j y_1 - \boldsymbol{x}_1'\tilde{\boldsymbol{\eta}}_j, \cdots, \tilde{\tau}_j y_n - \boldsymbol{x}_n'\tilde{\boldsymbol{\eta}}_j)$.

(2c) Update parameter block $\boldsymbol{\eta}_{l\cdot} = (\eta_{l1}, \cdots, \eta_{lk})$.

Minimizing $S(\Phi_{-\pi})$ with respect to $\boldsymbol{\eta}_{l\cdot}$ is equivalent to minimizing (with matrix notation)

$$\min_{\boldsymbol{\eta}_{l\cdot}} \frac{1}{2n} || \sqrt{\boldsymbol{W}^{(m)}} \odot (\tilde{\boldsymbol{R}} - \sum_{h \neq l} \boldsymbol{X}_{\cdot h}\tilde{\boldsymbol{\eta}}_{h\cdot} - \boldsymbol{X}_{\cdot l}\boldsymbol{\eta}_{l\cdot}) ||_F^2 + \lambda_1 ||\boldsymbol{\eta}_{l\cdot}||_1 + \lambda_2 ||\boldsymbol{\eta}_{l\cdot}||_2, \tag{3.13}$$

where $\sqrt{\boldsymbol{W}^{(m)}} = (\sqrt{w_{ij}^{(m)}})_{n\times k}$, $\tilde{\boldsymbol{R}} = (\tilde{r}_{ij})_{n\times k}$ with $\tilde{r}_{ij} = \tilde{\tau}_j y_i - \tilde{\eta}_{0j}$, $\lambda_1 = \lambda\alpha$, $\lambda_2 = \lambda(1-\alpha)\sqrt{k}$. Since the problem in Eq. (3.13) is convex, an efficient gradient decent type algorithm can be used that combines majorizing-minimization scheme with the first order condition (in terms of subgradient).

Details of Step (2c) We denote the quadratic term in Problem (3.13) (ignoring $1/2n$) as

$$M(\boldsymbol{\eta}_{l\cdot}) = ||\boldsymbol{A} - \sqrt{\boldsymbol{W}^{(m)}} \odot \boldsymbol{X}_{\cdot l}\boldsymbol{\eta}_{l\cdot}||_F^2,$$

where $\boldsymbol{A} = \sqrt{\boldsymbol{W}^{(m)}} \odot (\tilde{\boldsymbol{R}} - \sum_{h\neq l} \boldsymbol{X}_{\cdot h}\tilde{\boldsymbol{\eta}}_{h\cdot})$. Simple calculation gives us that for $j = 1, \cdots, k$:

$$\frac{\partial M}{\partial \eta_{lj}} = 2\big[|| \sqrt{\boldsymbol{W}^{(m)}}_{\cdot j} \odot \boldsymbol{X}_{\cdot l} ||_2^2 \eta_{lj} - (\sqrt{\boldsymbol{W}^{(m)}}_{\cdot j} \odot \boldsymbol{X}_{\cdot l})' \boldsymbol{A}_{\cdot j} \big]$$

$$\frac{\partial^2 M}{\partial \eta_{lj} \partial \eta_{lr}} = \begin{cases} 2||\sqrt{\boldsymbol{W}^{(m)}}_{\cdot j} \odot \boldsymbol{X}_{\cdot l}||_2^2 & \text{if } j = r \\ 0 & \text{if } j \neq r, \end{cases}$$

implying the Hessian matrix $\boldsymbol{H}_M$ of M is diagonal. If we let

$$t = 2 \max_{1 \leq j \leq k} ||\sqrt{\boldsymbol{W}^{(m)}}_{\cdot j} \odot \boldsymbol{X}_{\cdot l}||_2^2,$$

we see that $t\boldsymbol{I}$ will dominate Hessian $\boldsymbol{H}_M$ of M. That is, $t\boldsymbol{I} - \boldsymbol{H}_M$ is positive semi-definite. Consequently, we have a majorization function M_m by replacing $\boldsymbol{H}_M$ with $t\boldsymbol{I}$ in the second order Taylor expansion of M (which is M itself) at the current value $\tilde{\boldsymbol{\eta}}_{l\cdot}$ of $\boldsymbol{\eta}_{l\cdot}$:

$$M_m(\boldsymbol{\eta}_{l\cdot}) = M(\tilde{\boldsymbol{\eta}}_{l\cdot}) + (\boldsymbol{\eta}_{l\cdot} - \tilde{\boldsymbol{\eta}}_{l\cdot})\nabla M(\tilde{\boldsymbol{\eta}}_{l\cdot}) + \frac{t}{2}||\boldsymbol{\eta}_{l\cdot} - \tilde{\boldsymbol{\eta}}_{l\cdot}||_2^2,$$

$$M_m(\boldsymbol{\eta}_{l\cdot}) \geq M(\boldsymbol{\eta}_{l\cdot}),$$

where ∇M is the Jacobian of M.

With the help of the majorizing function $M_m(\boldsymbol{\eta}_{l\cdot})$, we approximately solve Problem (3.10) by solving:

$$\min_{\boldsymbol{\eta}_{l\cdot}} \frac{1}{2n} M_m(\boldsymbol{\eta}_{l\cdot}) + \lambda_1 ||\boldsymbol{\eta}_{l\cdot}||_1 + \lambda_2 ||\boldsymbol{\eta}_{l\cdot}||_2,$$

which is further equivalent to

$$\min_{\boldsymbol{\eta}_{l\cdot}} \frac{t}{4n} ||\boldsymbol{\eta}_{l\cdot} - (\tilde{\boldsymbol{\eta}}_{l\cdot} - \nabla M(\tilde{\boldsymbol{\eta}}_{l\cdot})/t)||_2^2 + \lambda_1 ||\boldsymbol{\eta}_{l\cdot}||_1 + \lambda_2 ||\boldsymbol{\eta}_{l\cdot}||_2 \tag{3.14}$$

For Problem (3.14), by the first order condition given by subgradient, the closed form

Table 6: Model parameter specification.

	M1	M2	M3	M4	M5	M6
n	300	300	450	300	300	300
p	50, 100	50, 100	50, 100	50, 100	50, 100	50,100
k	2	2	3	2	2	2
π	(0.5,0.5)	(0.5,0.5)	(1/3,1/3,1/3)	(0.5,0.5)	(0.5,0.5)	(0.5,0.5)
σ	(0.5,0.5)	(1.5,1.5)	(0.5,0.5)	(0.5,0.5)	(1,1)	(0.3,0.3)
β_1	(4,4,4,4,4)	(4,4,4,4,4)	(10,10,10,10,10,)	(4,4,4,4,0,0,0,0)	(4,4,4,4,4,4,0,0,0)	(4,4,4,4,4,4,0,0,0)
β_2	(-1,-1,-1,-1,-1)	(-1,-1,-1,-1,-1)	(3,3,3,3,3)	(0,0,0,0,-1,-1,-1,-1)	(-1,-1,-1,0,0,0,-1,-1,-1)	(-1,-1,-1,0,0,0,-1,-1,-1)
β_3	-	-	(-1,-1,-1,-1,-1)	-	-	-

of exact solution can be calculated and given as follows:

$$\boldsymbol{\eta}_{l\cdot}^{\star} = \begin{cases} 0, & ||T_{\lambda_1}(\frac{\boldsymbol{\mu}}{u})||_2 \leq \lambda_2 \\ \Big[1 - \dfrac{\lambda_2 u}{||T_{\lambda_1 u}(\boldsymbol{\mu})||_2}\Big] \cdot T_{\lambda_1 u}(\boldsymbol{\mu}), & ||T_{\lambda_1}(\frac{\boldsymbol{\mu}}{u})||_2 > \lambda_2, \end{cases} \tag{3.15}$$

where $u = 2n/t$, $\boldsymbol{\mu} = \tilde{\boldsymbol{\eta}}_{l\cdot} - \nabla M(\tilde{\boldsymbol{\eta}}_{l\cdot})/t$, $S(u,v) = \textsf{sign}(u)\max\{|u| - v, 0\}$ $(u \in \mathbf{R}, v \geq 0)$ is the soft-thresholding operator and $T_v(\boldsymbol{\mu}) = \big(S(\mu_1, v), \cdots, S(\mu_d, v)\big)$ is the element-wise soft-thresholding. Iteratively applying this update solves Problem (3.13).

Remark: The convergence of block coordinate descent for Problem (3.10) in non-trivial and well established in [146]. In our implementation, we approximately optimize M-step by making one-iteration progress. We also adopt random initialization for EM algorithm in the implementation as it can be trapped in local optima. In our practice, both strategies work well.

3.5 Experiments

3.5.1 Simulation

In this section, we set up six models (M1-M6) for simulation to investigate performances of (sparse) $l_{2,1}$ penalized FMR. M1, M2, M3 and M4 are designed to evaluate $l_{2,1}$

penalized FMR ($\alpha = 0$), with comparison to l_1 penalized FMR[1] ($\alpha = 1$). In M5 and M6, we evaluate the sparse $l_{2,1}$ penalized FMR ($0 < \alpha < 1$). For model evaluation, we report average results on 50 independent runs.

In all models, design matrix $\boldsymbol{X}$ is generated from multivariate normal distribution with zero mean and a diagonal covariance matrix. Response $\boldsymbol{Y}$ is then generated from finite mixture of k Gaussians. Details of model setup are specified in Table 6.

In each run of simulations, we partitioned simulated data (n observations) to three equal subsets: training, validation and testing sets. A sequence of models corresponding to a λ-sequence is trained on training data; the optimal value of tuning parameter λ along with its corresponding trained model is selected as the one minimizing NLogloss (see below) on validation data. Finally, predictive NLogLoss are evaluated and reported on testing data.

Performance metrics We assess the performance of penalized FMR from different perspectives. For predictive performance, we used negative log-likelihood loss (NLogLoss) in trained model:

$$-\sum_{i=1}^{n} \log\Big(\sum_{j=1}^{k} \hat{\pi}_j \frac{1}{\sqrt{2\pi}\hat{\sigma}_j} \exp\big(-\frac{(y_i - \hat{\beta}_{0j} - \boldsymbol{x}_i'\hat{\boldsymbol{\beta}}_j)^2}{2\hat{\sigma}_j^2}\big)\Big).$$

To measure sparsity, we first reported “true positive rate" (TPR) and “false positive rate"

[1] l_1 FMR fitted with R package "fmrlasso".

Table 7: M1-M4: predictive negative log-likelihood (smaller is better) with standard deviation for $l_{2,1}$ and l_1 penalized FMR.

p		M1	M2	M3	M4
50	$l_{2,1}$	**186.35** (10.80)	**267.94** (15.48)	**399.06** (24.86)	192.01 (17.35)
	l_1	190.21 (16.47)	277.30 (20.45)	402.28 (57.28)	**180.09** (16.41)
100	$l_{2,1}$	**201.36** (10.30)	**282.05** (27.33)	**429.35** (60.04)	219.65 (24.32)
	l_1	229.42 (31.94)	302.63 (23.87)	493.83 (54.44)	**198.43** (18.97)

(FPR):

$$\text{TPR} = \frac{\#\text{active parameters selected}}{\#\text{active parameters}},$$
$$\text{FPR} = \frac{\#\text{inactive parameters selected}}{\#\text{inactive parameters}}.$$

To further quantify accuracy of feature selection, we used root mean squared error (RMSE) of estimators:

$$\text{RMSE} = \sqrt{\frac{1}{kp}||\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}||_F^2},$$

where β is the matrix of true values and $\hat{\beta}$ is the estimation matrix.

Results on M1-M4 In M1-M4, we specifically investigate performances of $l_{2,1}$ FMR ($\alpha = 0$), in comparison with l_1 FMR. M1 and M2 have the same model setups with five active parameters, except that M2 is noisier than M1: true standard deviation is 0.5 vs 1.5. M3 is slightly more complicated with three mixture components. M4 has similar setups with M1 but differs in the sparsity pattern of active parameters. There are two different sparsity patterns in four models: M1, M2 and M3 are of the same sparsity pattern that all mixture components have active parameters corresponding to a same set of features; on the contrary, mixture components in M4 are of heterogeneous feature structure: none of features are active in both two components.

We fitted the models with $p = 50$ and $p = 100$, keeping true parameters unchanged

while doubling noisy parameters. Since $l_{2,1}$ penalty performs parameter selection at the group level, it implies an homogeneous assumption of active features for all mixture components. Therefore, $l_{2,1}$ FMR is expected to perform better in M1, M2 and M3, when the underlying assumption is satisfied, but worse than l_1 in heterogeneous case of M4. Simulation results confirm our expectations on $l_{2,1}$ FMR.

Table 7 presents the predictive negative log-likelihood. In either case of $p = 50$ and $p = 100$, $l_{2,1}$ FMR has better predictive power than l_1 FMR in M1, M2 and M3, but less in M4. In evaluation of feature selection, Figure 2 shows the results. We found that with respect to estimation accuracy and active parameter selection, RMSE and TPR display the same trend: $l_{2,1}$ is overall better than l_1 in M1, M2 and M3, but worse in M4. Interestingly in terms of FPR, l_1 penalty performs overall better than $l_{2,1}$ in M1-M4. This is possibly due to the effect of group selection in $l_{2,1}$ penalty. However, even with a larger FPR in M1, M2 and M3, the smaller RMSE of $l_{2,1}$ along with a larger TPR indicates that the false positives (i.e., inactive parameters selected) in $l_{2,1}$ are estimated accurately as insignificant numbers. It is particularly impressive that in M2 and M3, adding more noisy features, $l_{2,1}$ remains stable and robust while l_1 has a significant loss in TPR and RMSE.

Results on M5-M6 As we see the experimental results in M1-M4, the homogeneous assumption of active features is critical for $l_{2,1}$ penalty. In real cases, we believe this assumption is too restrictive to be satisfied. To alleviate the limits of $l_{2,1}$, we introduce sparse $l_{2,1}$ penalized FMR that is a weighted combination of l_1 norm and $l_{2,1}$ norm. As a result, the sparse $l_{2,1}$ penalty has a property of further selecting parameters within the selected parameter groups, which could potentially enhance the performance of $l_{2,1}$ penalty only.

With this purpose in mind, we designed M5 and M6 to show that sparse $l_{2,1}$ indeed

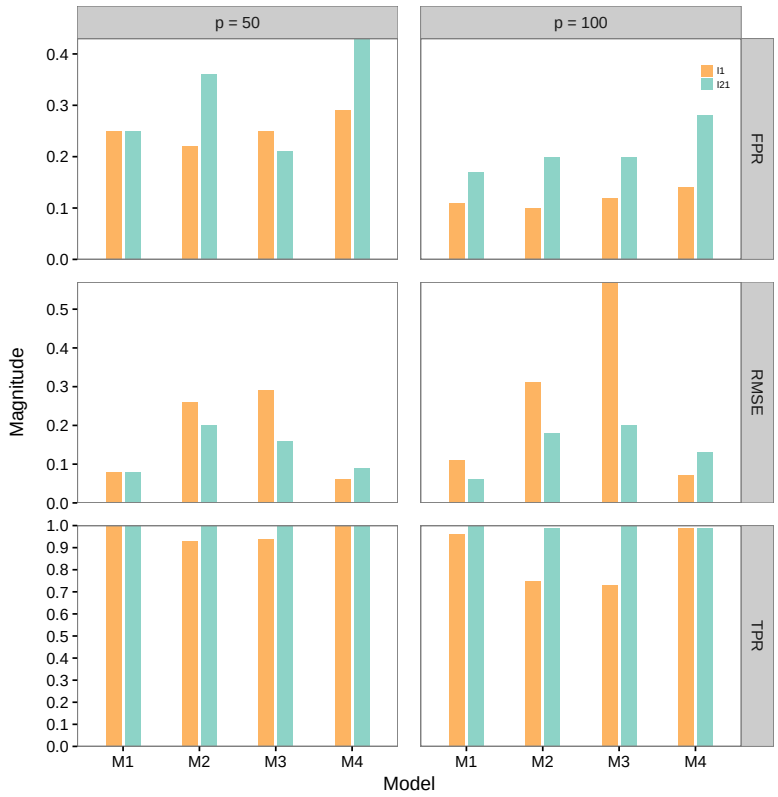

Figure 2: RMSE, TPR and FPR for $l_{2,1}$- and l_1 penalty on M1-M4. For RMSE and FPR, smaller is better; for TPR, larger is better.

Table 8: M5-M6: predictive negative log-likelihood (smaller is better) with standard deviation for sparse $l_{2,1}$ FMR, fitted with different α's.

p	α	M5	M6
50	0 ($l_{2,1}$)	258.45 (21.63)	188.35 (17.44)
	0.25	256.26 (21.56)	183.75 (17.83)
	0.50	**255.37** (22.06)	182.99 (18.21)
	0.75	256.22 (26.43)	181.72 (19.87)
	1 (l_1)	256.49 (25.96)	**166.82** (20.61)
100	0 ($l_{2,1}$)	281.24 (25.89)	224.08 (33.43)
	0.25	277.89 (26.87)	**216.77** (27.20)
	0.50	**277.56** (26.87)	218.72 (35.07)
	0.75	279.07 (26.92)	231.14 (55.18)
	1 (l_1)	289.84 (35.66)	236.09 (51.30)

benefits from incorporating within-group sparsity in $l_{2,1}$. M5 and M6 have the same setups except that M5 is noisier than M6. Notice that in M5 and M6, feature structures for mixture components are different yet with three active features in common. In these cases, we fit sparse $l_{2,1}$ penalized FMR with $\alpha \in (0, 1)$ and compare it with models of $l_{2,1}$ only and l_1

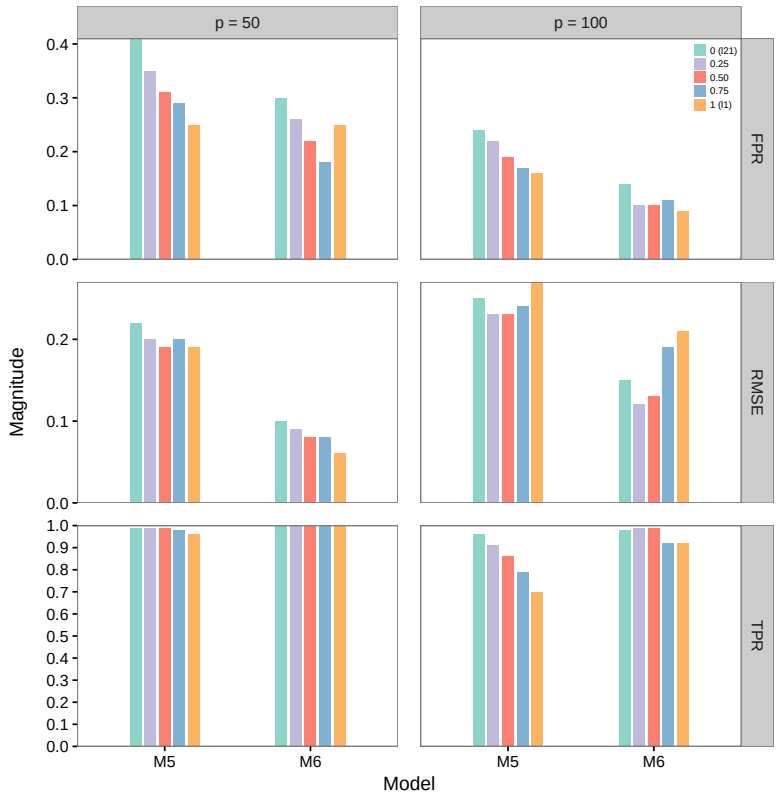

Figure 3: RMSE, TPR and FPR for sparse $l_{2,1}$ penalty FMR fitted with different α values $\{0, 0.25, 0.50, 0.75, 1\}$ on M5 and M6. Note that $\alpha = 0$ is $l_{2,1}$ penalty, $\alpha = 1$ is l_1 penalty.

only.

In the sparse $l_{2,1}$ penalty, α could be treated as a tuning parameter and the optimal value can be selected by validation set or cross validation. In our experiment, to keep computational cost moderate, we pre-fix α on a grid of values $\{0, 0.25, 0.5, 0.75, 1\}$, where $\alpha = 0$ corresponds to $l_{2,1}$ penalty and $\alpha = 1$ to l_1 penalty. Predictive negative log-likelihoods are shown in Table 8. We see that in four cases, the sparse $l_{2,1}$ penalty (that is, $\alpha \neq 0$) indeed provides improvement over $l_{2,1}$ penalty only. In the setting of higher dimension $p = 100$ or larger noise $\sigma = 1$, it also performs better than l_1 penalty. Figure 3 shows sparsity performance. One observation is that, due to group selection effect, $l_{2,1}$ overall has larger both the TPR (larger is better) and FPR (smaller is better); incorporating l_1 into

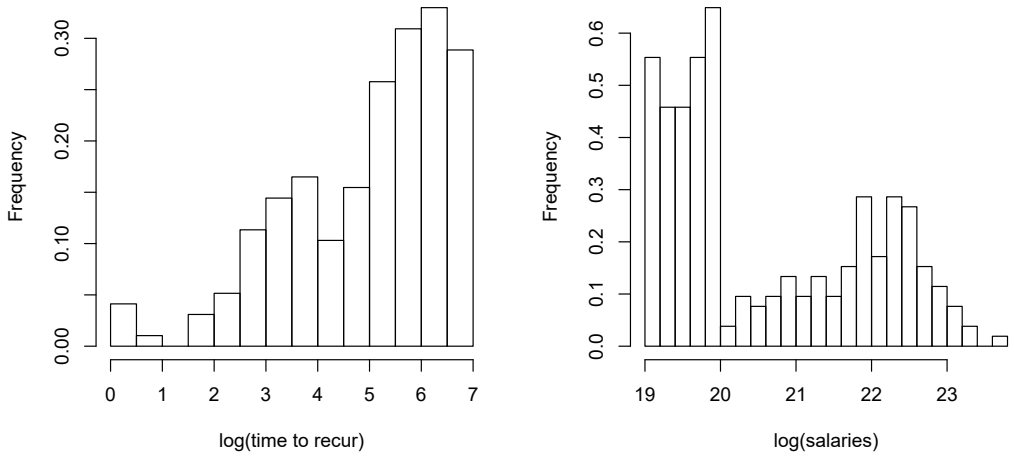

Figure 4: Histogram of log(time to recur).

$l_{2,1}$ alleviates this trade-off, leading that the sparse $l_{2,1}$ has better estimation RMSE than the single $l_{2,1}$ penalty. Similarly as predictive performance, it also has better accuracy than the single l_1 penalty in the noisier models.

3.5.2 Real Data Application

We apply the sparse $l_{2,1}$ penalized FMR in (1) Wisconsin breast cancer dataset (WBC) that is publicly available at UCI machine learning repository[2]; (2) NHL salaries dataset [3]. WBC contain 194 records of “time to recur" for patients with breast cancer (after removing 4 cases with missing values) with 32 features describing characteristics of the cell nuclei in the digitized image of breast cancer. NHL dataset used in experiments has 262 records of NHL players with 122 features measuring players' performances. In our experiment, we used the logarithm of the original targets as the targets due to the right-skewness of original scale. Figure 4 shows the histograms of transformed targets.

The histogram in Figure 4 demonstrates a highly unbalanced mixture of Gaussian dis-

[2]http://archive.ics.uci.edu/ml/index.html
[3]https://www.kaggle.com/camnugent/predict-nhl-player-salaries

Table 9: Predictive performance. 10-fold mean predictive negative log-likelihood for unpenalized-, l_1 penalized- and sparse $l_{2,1}$ penalized FMR. Number of components varies from 1 to 3.

	Model/Component(s)	1	2	3
WBC	unpenalized	1.87	3.10	15.70
	l_1	1.70	1.71	1.69
	sparse $l_{2,1}$	-	1.74	**1.66**
NHL	unpenalized	1.85	-	
	l_1	1.41	1.40	
	sparse l_{21}	-	**1.38**	

tribution. Thus we fit FMR with $k = 1, 2, 3$ components for WBC and $k = 1, 2$ for NHL. Three methods, unpenalized-[4], l_1 penalized- and sparse $l_{2,1}$ penalized FMR, were applied in the experiment. For penalized FMR, tuning parameters λ and α are selected by 10-fold cross-validation based on predictive mean negative log-likelihood (CV loss).

Predictive Performance Table 9 shows the results of predictive negative log-likelihood. It turns out that the optimal value (1.66 and 1.38) is achieved with sparse $l_{2,1}$ FMR. An interesting observation for WBC is that for unpenalized FMR, a mixture of two or three components introduces much variance in modeling, resulting in a significant larger CV loss than a single component linear model (3.10 and 15.70 respectively compared with 1.87); for NHL data, unpenalized mixture of two Gaussians fails due to the high dimensionality (122 features). In contrast, l_1 and $l_{2,1}$ FMR are more robust when model complexity increases.

Feature Selection In Table 10, the sparse $l_{2,1}$ FMR selects 8 features for WBC data. The selected features demonstrate heterogeneous effects in different mixture components, possibly due to high correlations among features. However, in terms of predictive performance, the sparse $l_{2,1}$ penalized FMR performs even better than the unpenalized uni-modal

[4]Unpenalized FMR was fitted with R package "flexmix".

Table 10: Parameter estimation for WBC dataset.

Feature	Component 1	Component 2	Component 3
texture	-0.060	-0.010	0.028
area	-0.004	-0.002	- 0.001
area SE	0.012	-0.008	-0.013
worst texture	-0.044	-0.041	0.045
worst perimeter	0	-0.014	0.007
largest area	0	0.002	0.001
tumor size	0.077	-0.027	-0.015
positive lymph nodes	0.047	0.011	-0.097

Table 11: Parameter estimation for NHL dataset.

Feature	Component 1	Component 2
games played	0.057	0.016
shifts	0.716	0.091
setup passes	0.012	0
shots on goal	0.005	0
faceoff wins	1.371	0.070
penalty drawn	0.657	0.042

model (non-mixture) with an 11% improvement ((1.87-1.66)/1.87). Table 11 shows the results for NHL data. The selected features suggest that players who are able to bring more wins or scores (faceoff wins, penalty drawn and shifts) have higher salaries (Component 1 corresponds to the left peak and Component 2 to the right peak of the histogram in Figure 4). This interpretation accords with our intuitions that better players have higher pays.

3.6 Conclusion

In this chapter, we have introduced the sparse $l_{2,1}$ penalized FMR model with feature selection. The $l_{2,1}$ penalty captures relatedness among mixture components. We have shown in the simulation studies $l_{2,1}$ is more robust than l_1 penalized FMR in noisy cases. Sparse $l_{2,1}$ improves $l_{2,1}$ allowing component-dependent feature structure. For model inference, we use EM algorithm and propose an efficient update scheme based on re-parametrization. Finally, sparse $l_{2,1}$ is applied to a real world dataset with an improvement over full linear model as well as feature selection. Future work includes extending the sparse $l_{2,1}$ to

mixtures of logistic or Poisson regressions.

CHAPTER 4 PREDICTIVE MODELING FOR HEALTHCARE INFORMATICS

4.1 Introduction

Predictive modeling is an important task in clinical research, as it can unveil associations between risk factors and asymptomatic disease phenotypes enabling prediction of patients most likely to benefit from early intervention. Traditional methods, such as linear regression, are often used but render predictions based only on the low-level features (e.g., demographics, blood pressure, cholesterol level, glucose level et al.), simplifying the inherent, complex biological mechanism of disease progression as additive effects of low-level features. As a result, nonlinear relations are excluded leading to information losses, and potentially less accurate performance. On the contrary, deep neural networks (DNNs) have made impressive improvements for complex predictive tasks in natural language processing and computer vision. The key factor for their success is that DNNs are capable of learning high-level information from low-level input features. This merit of DNN makes it very promising for predictive modeling in clinical research, enabling capture of non-additive effects among various low-level features. Successful DNNs often require abundant labeled data to avoid overfitting.

However, there exist two challenges for DNN applications in healthcare informatics. The first challenge is the limited availability of large datasets. Collecting labeled data in clinical practice is expensive and time-consuming as the labeling process requires dedicated personnel, explicit definitions, and a formalized mechanism for data compilation. Consequently, we only have limited available labeled data, precluding application of DNN methods to various challenging, but important clinical prediction problems. On the contrary, directly fitting DNNs with small data can result in severe overfitting that is not de-

sirable in healthcare informatics. The second one is that as patient subgroups exhibit differential health outcomes that are potentially associated with different risk factors, a successful model would ideally take these aspects into consideration for patient stratification. However, existing DNN models either rely on the pre-defined patient subgroups or take the “one-size-fits-all" approach and are built without considering patient stratification. Consequently, those models are not able to discover patient subgroups and the risk factors are thereafter identified for the entire patient population, failing to account for potential group differences.

In this chapter, we propose two different DNN models for the challenges mentioned above. For the small data problem, we propose auxiliary-task augmented network (ATAN), a model that predicts the primary target with introducing clinically relevant measures as auxiliary predictive tasks. With auxiliary tasks, ATAN takes advantage of benefits from multi-task learning (MTL) — an approach of regularization and implicit data augmentation. Without assumption of homogeneous feature representation for all tasks in classic multi-task frameworks, ATAN explicitly models the shared feature representation for all tasks, as well as task-specific representation, and combines them together using a weighting mechanism to capture the clinical relevance. By the weighting mechanism, we conceptually quantify the relevance between the primary and auxiliary target. While LVMI was collected as the primary target in our test case, many other variables from CMR that are clinically related with LVMI are also available (see Section IV-A), providing auxiliary targets that can be utilized in predictive modeling. To demonstrate the effectiveness of our method, we apply ATAN to our hypertension dataset and compare its predictive performance with different popular methods (k-nearest neighbor, linear regression, Lasso et al.

[30]). We also interpret our model by analyzing the learned parameters (a.k.a. weights) to identify clinically relevant risk factors.

For patient stratification problem, we introduce a unified DNN model, termed as deep mixture of neural networks (DMNN), that simultaneously predicts clinical outcomes and discover patient subgroups. DMNN consists of an embedding network with gating (ENG) and several local predictive networks (LPNs). ENG embeds raw input features into high-level feature representation which is further used as input for LPNs. Unlike the existing DNN models without subgroup identification, patients will be grouped that share similar functional relations between inputs and clinical outcomes via the gating mechanism in ENG, and each functional relation is modeled by one LPN. The subgroup discoveries enable us to apply existing interpretation techniques to identify subgroup-specific risk factors. By explaining the local input-outcome relations captured by LPN within each patient subgroup, the subgroup-specific sets of risk factors can provide information to account for health disparities. To demonstrate the effectiveness of DMNN, we conduct extensive experiments on a clinic dataset for predicting the diagnosis of acute heart failure. DMNN can achieve state-of-art predictive performance when comparing with other baseline machine learning models. We apply mimic learning technique [9] for interpretation on DMNN and show that DMNN can provide informative risk factors for clinical decision making.

4.2 Related Work

Deep Predictive Model Applications of deep learning models have been flourishing due to the increasing availability of EHR data. Tang et.al[143] and Purushotham et.al[115] benchmark the performance of DNNs with comparison to other machine learning models on MIMIC III data. Che et.al[9] proposed DNN model that incorporates prior knowledge of

medical ontologies as regularization for predicting ICU outcomes. Choi et.al[13] proposed a recurrent neural network with GRU for predictions of heart failure onset. Li et.al[83] and Suo et.al[139] develop multi-task DNN models for predicting disease progression and diagnosis where multiple targets are used as an approach of regularization. Another line of predictive modeling with deep learning is to utilize the power of deep unsupervised learning to learn high-level feature representations in the latent vector space. DNN feature learning significantly reduces the workload of feature engineering especially for complex data like EHR and can be used in the downstream predictive tasks. For example, Miotto et.al[98] and Lasko et.al[69] learn patient representations from EHR, and Choi et.al[15] embeds diagnosis codes and procedure codes into vector space. The learned representations are then used to predict patient health outcomes. However, those methods don't specifically consider the heterogeneity in patients and consequently, model interpretations are for the entire patient population, lacking granularity to account for subgroup differences.

Multi-task learning MTL for predictive modeling is a machine learning paradigm that jointly learns a model for multiple tasks [163]. When these tasks are related to each other, the joint learning can lead to improvement in the generalized predictive performance by leveraging information contained in other tasks. This perspective inspires a learning strategy that many single-task problems often introduce auxiliary tasks and thereafter be transformed into multi-task problems. With this approach for the target (single) task, MTL can be viewed as a method of regularization and implicit data augmentation. See [7, 37, 123, 163] for more details on MTL. There have been many successful applications of MTL in biomedicine. For example, [136, 161] apply MTL in conjunction with linear

model to predict Alzheimer's disease progression; [111] uses MTL for a multi-label task in a clinical text classification problem; [114] propose group Lasso under the MTL paradigm to detect population heterogeneity at the genetic level.

DNN MTL DNN-based MTL (DMTL) has been attracting much interest recently as DNNs can be conveniently adapted to MTL: DNN can have multiple output neurons for multiple tasks. Aside the high-level learning of DNN, another advantage of DMTL over linear-based modeling is that the hierarchical feature learning provides a flexible way of capturing relevance among multiple tasks. These merits lead to many applications for various problems such as speech recognition and computer vision ([89, 125, 157, 164]). Particularly in bioinformatics and health informatics, [117] uses DMTL to predict protein interactions between HIV and human proteins; [137] integrates DNN feature learning into SVM-based MTL for diagnosis of Alzhermer's disease. For general considerations in DMTL, we refer to [7] and [123] for details.

Patient Subgroup Discovery Subgroup identification is one of the most important tasks in medical science and has been studied in various settings. For example, Seymour et.al[126] and Lu et.al[92] use unsupervised clustering techniques to group patients by sepsis and cancer subtypes respectively. In the study of treatment efficacy, subgroups are identified as the patients that have similar treatment responses. In this context, tree methods[81, 134, 90, 26] are developed that patients are grouped in the leaf node that are homogeneous treatment effect. For predictive modeling that is studied in this chapter, finite mixture of (Gaussian and logistic) regression (FMR)[78, 124, 132] and mixture of experts (ME)[60, 61] were used to identify patient subgroups. However, FMR and ME models are based on the conventional machine models which is less capable of extracting

high-level information compared with DNNs. Also, they usually require clean and structured data to train and are not capable of handling complex EHR data (e.g multi-modal and sequential data). To address those challenges in FMR and ME, our proposed DMNN builds a unified DNN architecture that not only exploits the predictive power of DNNs but also be able to discover patient subgroups.

4.3 Proposed ATAN

In this section, we present the proposed method ATAN. We start with the fully connected network for predictive tasks. Then we describe our approach of utilizing auxiliary targets under the multi-task learning framework. Finally, we present a method adapted from [35] to our case for finding the risk factors.

Notations: We use **R** to represent the set of real numbers. Vectors and matrices are denoted as bold letters and scalars as unbold letters.

4.3.1 Basic Feed-forward Network

We implemented fully connected feed-forward neural network (FNN) as the building block of our model. But FNN itself can serve as a predictive model. The mathematical formulation of FNN is described as follows.

Let $(\boldsymbol{x}, y) \in \mathbf{R}^p \times \mathbf{R}$ be a sample, where $\boldsymbol{x}$ is the input p-dimensional feature and y be the target. A FNN for regression task consists of one layer that takes $\boldsymbol{x}$ as input, followed by k hidden layers for learning feature representations:

$$\begin{aligned} \boldsymbol{h_1} &= \sigma(\boldsymbol{W_1}\boldsymbol{x} + \boldsymbol{b_1}), \\ \boldsymbol{h_i} &= \sigma(\boldsymbol{W_i}\boldsymbol{h_{i-1}} + \boldsymbol{b_i}), \quad i = 2, \cdots, k \end{aligned} \tag{4.1}$$

and one output layer that make predictions using the highest level feature representations:

$$\hat{y} = \boldsymbol{W}\boldsymbol{h_k} + b,$$

where $\boldsymbol{W_i}$ and $\boldsymbol{W}$ are the weight matrices with compatible dimensions, $\boldsymbol{b_i}$ and b the bias terms and $\boldsymbol{h_i}$ the hidden state; $\sigma(\cdot)$ is the element-wise activation function, such as sigmoid, ReLU or tanh; $\hat{y}$ is the prediction of the network.

When there are multiple targets, FNN can be modified effortless to accommodate the case. In the output layer,

$$\hat{\boldsymbol{y}} = \boldsymbol{W}\boldsymbol{h_k} + \boldsymbol{b},$$

where $\hat{\boldsymbol{y}} = (\hat{y}_1, \cdots, \hat{y}_T)$ is the predictive vector of T targets. Notice that in this formulation of multi-task learning, an implicit assumption is made that all tasks share the feature representation.

To learn the model parameters, gradient descent based methods along with back propagation are used to minimize the least square loss function.

4.3.2 ATAN Structure

Our central task is to build a predictive model for the primary target that clinicians care about but costly to label (LVMI in our motivation example). We denote the primary target by y^c. In practice, we observe that in addition to the primary measure y^c, there are often other measures available, denoted as y^a, that are clinically relevant to y^c. This relevance provides additional information for the primary task. By incorporating y^a as the secondary target, the predictive model can benefit from multi-task learning as described in Section

II.

However, one pitfall with introducing auxiliary tasks is that the “relevance" is not universally defined. Multi-task methods generally either make assumptions, or define the relevance based on the domain-specific knowledge. ATAN circumvents this issue assuming that the primary target y^c and the secondary target y^a are jointly regulated by the shared and task-specific biological mechanisms. Translating this assumption in modeling, ATAN learns a feature representation that can be decomposed into a weighted sum of the shared and task-specific feature representation. Fig. 5 presents the high-level overview of ATAN structure.

Learning feature representations Let $(\boldsymbol{x}, y^c, y^a) \in \mathbf{R}^p \times \mathbf{R} \times \mathbf{R}$ be a sample. We first learn the shared and task-specific feature representations using feed-forward DNN (FDNN) as follows

$$\begin{aligned} \boldsymbol{h}^s &= \text{FDNN}^s(\boldsymbol{x}), \\ \boldsymbol{h}^c &= \text{FDNN}^c(\boldsymbol{x}), \\ \boldsymbol{h}^a &= \text{FDNN}^a(\boldsymbol{x}), \end{aligned} \tag{4.2}$$

where FDNN$(\cdot)$ is calculated by (4.1) and the activation function therein is the element-wise sigmoid function $1/(1+\exp(-x))$. For notational convenience, we use E^s, E^c and E^a to represent the set of parameters for FDNN^s, FDNN^c and FDNN^a respectively.

Based on our assumption, the final feature representations $\boldsymbol{h}^{fc}$ and $\boldsymbol{h}^{fa}$ for the primary

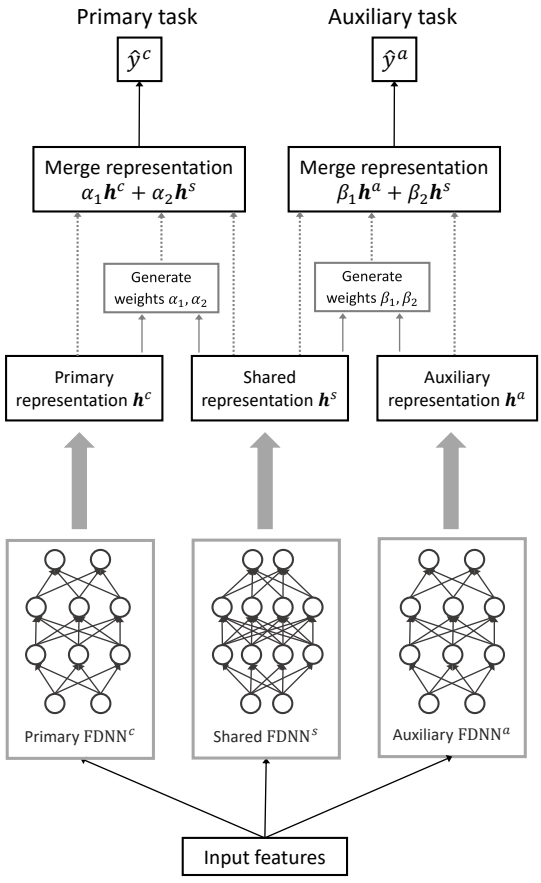

Figure 5: Overview of ATAN with one auxiliary task. ATAN uses the shared network FDNN^s to capture the clinical relevance between the primary and auxiliary task and two independent networks to learn the task-specific feature representations. Then ATAN merges the shared and task-specific feature representations via a weighting scheme. Finally, ATAN makes predictions using the merged representation. Note that FDNN^c, FDNN^s and FDNN^a are not restricted to the same architecture.

and auxiliary task respectively are decomposed as a weighted sum as follows:

$$\boldsymbol{h}^{fc} = \alpha_1 \boldsymbol{h}^c + \alpha_2 \boldsymbol{h}^s, \tag{4.3}$$

$$\boldsymbol{h}^{fa} = \beta_1 \boldsymbol{h}^a + \beta_2 \boldsymbol{h}^s, \tag{4.4}$$

where $\{\alpha_1, \alpha_2\}$ and $\{\beta_1, \beta_2\}$ are the attention weights that conceptually quantify the contributions of the shared and task-specific feature representations. Note that in this formulation, $\boldsymbol{h}^s$, $\boldsymbol{h}^c$ and $\boldsymbol{h}^a$ are of the same dimension. As a side note, another strategy to combine the task-specific and shared feature representations is through vector concatenation $\boldsymbol{h}^{fc} = [\boldsymbol{h}^c, \boldsymbol{h}^s]$. But this approach could introduce more parameters for each $\boldsymbol{h}$ having enough representation power. We hence prefer the weighted sum approach when only limited amount of data is available.

To calculate $\{\alpha_1, \alpha_2\}$ and $\{\beta_1, \beta_2\}$, we adopt a strategy that represents the compatibility of shared and task-specific feature representations:

$$\begin{aligned} \alpha_1 &= \frac{1}{2}\text{cosd}(\boldsymbol{h}^c, \boldsymbol{h}^s), \\ \alpha_2 &= 1 - \alpha_1, \end{aligned} \tag{4.5}$$

for the primary task, and

$$\begin{aligned} \beta_1 &= \frac{1}{2}\text{cosd}(\boldsymbol{h}^a, \boldsymbol{h}^s), \\ \beta_2 &= 1 - \beta_1, \end{aligned} \tag{4.6}$$

for the auxiliary task, where $\text{cosd}(\cdot, \cdot)$ is the cosine-distance between two vectors $\text{cosd}(\boldsymbol{v}_1, \boldsymbol{v}_2)$

$= \boldsymbol{v}_1 \cdot \boldsymbol{v}_2/(||\boldsymbol{v}_1||_2||\boldsymbol{v}_2||_2)$, $||\cdot||_2$ is the euclidean norm of a vector. Since we use sigmoid as the activation function, $\{\alpha_1, \alpha_2\}$ and $\{\beta_1, \beta_2\}$ are positive and hence proper weights. Note that this strategy biases toward the shared feature representation and forces it to makes at least half contribution (i.e., $\alpha_2, \beta_2 \geq 0.5$) to the final feature representation for ATAN enjoying benefits of multi-task learning.

Model Training The predictions for y^c and y^a is calculated using the final feature representations:

$$\begin{aligned} \hat{y}^c &= \boldsymbol{W}^c \boldsymbol{h}^{fc} + b^c, \\ \hat{y}^a &= \boldsymbol{W}^a \boldsymbol{h}^{fa} + b^a, \end{aligned} \tag{4.7}$$

where $\boldsymbol{W}^c$ and $\boldsymbol{W}^a$ are dimension-compatible vectors, b^c and b^a are bias terms.

The objective function is a weighted sum of the least squared loss functions for the primary and auxiliary tasks:

$$\min_{E^s, E^c, E^a, \boldsymbol{W}^c, \boldsymbol{W}^a, b^c, b^a} \sum_{i=1}^{n} (y_i^c - \hat{y}_i^c)^2 + \omega (y_i^a - \hat{y}_i^a)^2, \tag{4.8}$$

where the summation is over n training samples, ω is a parameter controlling the weight of the auxiliary task. We use standard gradient descent algorithm to solve Problem (4.8). Note that ATAN is not restricted to one auxiliary target and multiple auxiliary targets can be incorporated straightforwardly.

4.3.3 Analyzing Weights

Model interpretability and accuracy are both critically important in in clinic practice. While DNNs is generally difficult to interpret, we can still analyze the learned weights to

see how contributions of input features propagate through the network. This approach was initially proposed in [35]. Here we adapt this method to fit the proposed ATAN model.

To keep notations uncluttered, let us take an example, as shown in Fig. 6, to see how contributions of input features to the target can be recursively calculated from the output end of DNNs. Assume the last 3 layers of FDNN are of size 2, 3 and 1 respectively; $\boldsymbol{W_1} = (w_{ij}^1)_{3\times 2}$ and $\boldsymbol{W_2} = (w_{ij}^2)_{1\times 3}$ are the two weight matrices between layers. Let (g_1, g_2) and (h_1, h_2, h_3) be the two hidden layers, y the output layer.

For a hidden neuron h_t $(t = 1, 2, 3)$, its contribution C_t to the target y can be computed as in linear regression:

$$C_{ty} = \frac{|w_{1t}^2|}{\sum_{i=1}^{3} |w_{1i}^2|}.$$

For g_k $(k = 1, 2)$, its contribution C_{kt} to h_t is

$$C_{kt} = \frac{|w_{tk}^1|}{\sum_{i=1}^{2} |w_{ti}^1|}.$$

Then the contribution Q_{kty} of g_k *through* h_t to the target y is defined as

$$Q_{kty} = C_{kt} C_{ty}.$$

Summing over all hidden neurons, the total contribution Q_{ky} for g_k is given by

$$Q_{ky} = \sum_{t=1}^{3} Q_{kty}.$$

For input features x_i's, we can recursively apply the strategy above to calculate the

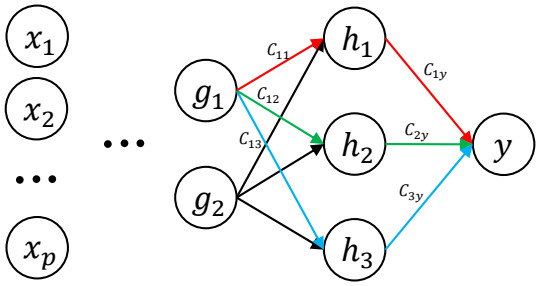

Figure 6: A simple example showing the recursively procedure of calculating the contribution from g_1 to the target y via h_1 (red), h_2 (green) and h_3 (blue) using weight propagation [35]. The total contribution Q_{1y} from g_1 to y is $Q_{1y} = C_{11}C_{1y}+C_{12}C_{2y}+C_{13}C_{3y}$. Recursively applying this strategy can calculate the contributions of input feature x_i's.

contributions.

Within the proposed ATAN model, the contribution of each input features to the primary target y^c can be propagated through the task-specific network FDNN^c and the shared network FDNN^s. Hence if we assume $Q^c_{ky^c}$ and $Q^s_{ky^c}$ are the contributions of feature x_k through FDNN^c and FDNN^s to y^c respectively, the overall contribution Q_{ky^c} for x_k is just the weighted sum given by

$$Q_{ky^c} = \alpha_1 Q^c_{ky^c} + \alpha_2 Q^s_{ky^c},$$

which provides us a heuristic approach for interpreting ATAN, α_1 and α_2 are given by (4.5).

4.3.4 Application

In this section, we apply ATAN to our clinical dataset of African Americans with hypertension at-risk for LVH and compare it with various baseline models to demonstrate its effectiveness. We also analyzed ATAN with its learned parameters to rank risk factors. This provides us some insight for understanding the progression of and potential targeted early intervention for cardiovascular disease.

Data Information and Preprocessing Our study dataset was derived from a cohort of

African American patients who presented to the emergency department of a single center (Detroit Receiving Hospital) between October 2011 and November 2014 with a known history of hypertension and elevated systolic blood pressure ($>$ 160 mm Hg). Previous studies have shown that there are disparities among hypertension patients with some who are at greater risk of LVH, yet no single variable has sufficient power for explaining the disparity. This motivates us to use a DNN model that is capable of capturing complex feature interactions.

As noted, LVH was determined using LVMI as measured on CMR using a cut-point of 89 g/m2 in men and 73 g/m2 in women. Along with LVMI, other measures from CMR are also available, such as wall thickness, left ventricular stroke volume to body surface area (LVSVI) and left ventricular end-diastolic volume indexed to body surface area (LVEDVI). These measures are clinically related to LVMI and could be used as the auxiliary targets in ATAN.

For data preprocessing, we first remove samples with missing LVMI and measures whose missingness is greater than 10%. The remaining dataset consists of 155 samples and 65 measures, where 59 measures are used as features and the remaining 6 including LVMI and other CMR results are used as targets. For missing values, we use “multiple imputation chained equations" (MICE) [155] for imputation. We also standardize feature values to have zero mean and unit variance. Table 12, 13 and Fig. 7 show the detailed information about the used dataset and descriptive statistics of LVMI respectively.

Experiment Implementation We implement ATAN in Pytorch [116]. In ATAN, we select one CMR measure as the auxiliary target. We experiment with posterior wall thickness and LVEDVI separately for our auxiliary task. The architecture of ATAN is of 4 layers, within

Table 12: Details of hypertension dataset. LVMI from CMR is the primary target and other measures in CMR serve as the candidates for auxiliary tasks.

# Sample	155	
# Features	59	• Demographics • Lab results (calcium, eGFR et al.) • Heart functioning (LV ejection rate et al.) • Acoustic and electrocardiography (Cornell product et al.) • Applanation tonometry (Central tension time index et al.)
# CMR results	6	• LVMI • Heart wall thickness (septal, posterior and anterior) • LVSVI • LVEDVI

Table 13: Descriptive statistics of LVMI.

Min	1st Qtl	Median	3rd Qtl	Max	Mean	Size
51.06	80.06	89.72	100.83	155.66	90.81	155

which 2 hidden layers are used to learn the shared and task-specific feature representations. We also restrict the dimension of hidden layers to 80 and 40 for FDNN^c, FDNN^s and FDNN^a for experimental convenience. ATAN is trained using standard gradient descent in conjunction with L2 regularization.

For performance comparison, we also implement various baseline models in scikit-learn library [110]. These baselines include k-nearest neighbors (KNN), random forest (RF), support vector regression (SVR), regularized linear regression (Ridge and Lasso) and also the multi-task Lasso (MTLasso). We also implement a 4-layer FNN model (MLP-4) for predicting LVMI only. The hidden size is chosen to match the feature representation level and dimension of ATAN.

The complete dataset is divided into training and testing sets by a split 125/30. For

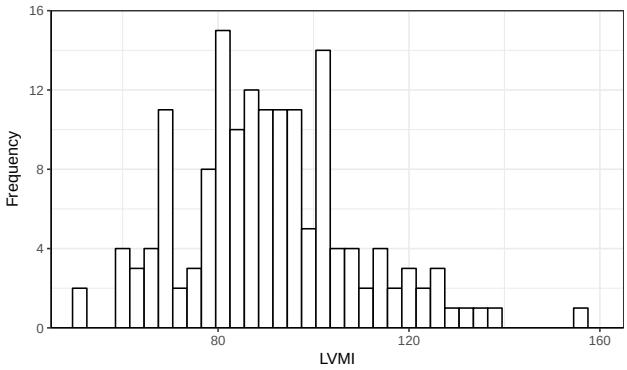

Figure 7: Histogram of LVMI.

ATAN and MLP-4, we split out 90 samples from the training set to actually train the models and the remaining 35 samples is used for validation and parameter selection. For baselines, we use three-fold cross-validation on the training set for parameter selection. The evaluation metrics are finally reported on the testing set. We repeat this procedure 10 times.

For model evaluations, we use the following three metrics. Assume that $\boldsymbol{y}^c = (y_1^c, \cdots, y_n^c)$ is the true vector of the primary target of n samples, $\hat{\boldsymbol{y}}^c = (\hat{y}_1^c, \cdots, \hat{y}_n^c)$ the vector of predicted values:

- Mean squared error (MSE) measures the predictive error without considering the magnitude of target:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i^c - \hat{y}_i^c)^2.$$

- Explained variance score (EVS) computes the explained variation that a model accounts for the data:

$$\text{EVS} = 1 - \frac{\text{Var}(\boldsymbol{y}^c - \hat{\boldsymbol{y}}^c)}{\text{Var}(\boldsymbol{y}^c)},$$

  where $\text{Var}(\cdot)$ is the variance.

- Median absolute error (MAE) is a more robust error than MSE that compute the median of absolute predictive errors:

$$\text{MAE} = \text{Med}(|\boldsymbol{y}^c - \hat{\boldsymbol{y}}^c|),$$

where MED$(\cdot)$ outputs the median of a vector.

Results

Using entire feature set We first evaluate models using all features in the hypertension data. The predictive performance on the test data is reported in Table 14. In the table, ATAN-1 and multi-task lasso (MTLasso) use LVMI and LVEDVI as the tasks and ATAN-2 uses posterior wall thickness as the auxiliary task. From the table, we have following observations.

- Among all models, ATAN with LVEDVI as the auxiliary target (ATAN-1) achieves the best predictive performance and ATAN with posterior wall thickness (ATAN-2) the second best in terms of MSE, EVS and MAE. For example, ATAN-1 and ATAN-2 provide approximately 5% and 4% improvements over Lasso in MSE respectively.

- Compared with Lasso, the vanilla DNN (MLP-4) has almost identical performances. Although DNN is expected to capture the complex relations among features for predicting LVMI, DNN may not efficiently learn feature representation for such small data. However, ATAN-1 and ATAN-2 that introduce auxiliary tasks based on domain knowledge bring some improvements in predictive modeling. This confirms that multi-task framework is an effective regularization approach.

Table 14: Predictive performance along with standard deviations on the testing data. For MTLasso and ATAN, performance on LVMI is reported. ATAN-1 uses LVEDVI as auxiliary target and ATAN-2 uses posterior wall thickness. For MSE and MAE, smaller is better; for EVS, larger is better.

Method	MSE	EVS	MAE
KNN	254.140 (52.329)	0.230 (0.155)	11.071 (1.861)
RF	227.803 (34.387)	0.261 (0.129)	10.778 (2.284)
SVR	294.831 (67.502)	0.083 (0.015)	10.530 (2.735)
Ridge	257.496 (26.513)	0.143 (0.277)	12.464 (1.742)
Lasso	205.983 (28.081)	0.337 (0.109)	11.079 (1.975)
MTLasso	213.398 (31.025)	0.310 (0.136)	11.177 (2.015)
MLP-4	204.104 (20.869)	0.338 (0.126)	10.323 (1.789)
ATAN-1	**195.820** (25.991)	**0.372** (0.090)	**10.149** (1.392)
ATAN-2	198.320 (22.427)	0.356 (0.122)	10.185 (1.812)

- Introducing auxiliary task may not always work. The multi-task Lasso (i.e., MTLasso) has worse performance than Lasso. One reason accountable for this case is that MTLasso assumes that all tasks share the same feature structure. This is however unlikely for LVMI and LVEDVI being regulated by the same set of features.

- ATAN-1 and ATAN-2 performs better than MTLasso, with margins 8% and 7% (MSE), 20% and 15% (EVS), 9% and 8.8% (MAE). This is possibly due to less restrictive assumption of ATAN for defining “relevance" between two targets: ATAN conceptually capture the “relevance" by learning feature representations shared by tasks. This

implies that effectively modeling task relevance is an integral part under the multi-task learning.

- The explained variance score (EVS) are overall low on the testing data. This is because that EVS is very sensitive to bad predictions, and indeed all models fail to make good predictions for some samples. Fig. 7 shows the histogram of LVMI. The histogram implies data might be generated from a multi-modal distribution and methods hence fail to model the local data structure when not enough data is presented.

By examining the predictive performance, we find that predictions often fail at the left and right tails in the sample distribution (results not shown). Previous study ([49, 73]) has discovered the correlation between LVMI and calcium metabolism and shown that patients with LVH (i.e., large LVMI values) have significant higher serum calcium level than those without LVH. For our dataset, we find that in the right tail of data distribution (LVMI value > 120), correlation between calcium level and LVMI is 0.79 and the two-tail correlation test is significant with p-value $= 0.0004$. However, correlation between calcium and LVH is 0.006 in the entire dataset, and -0.100 for LVMI < 120. This implies that LVH prevalence is different among different patient subgroups, and predictive models may fail to capture the disparities when we only have limited amount of data.

Using demographics and lab results only As we see from Table 12, many features are functional measures of the cardiovascular system. However in practice, much previous study has focused on the relations between LVMI and lab results along with demographics, as these relations are more informative on the disease progression and often readily

Table 15: Predictive performance along with standard deviations on the testing data. Only demographics and lab results are used as input features. For MSE and MAE, smaller is better; for EVS, larger is better.

Method	MSE	EVS	MAE
KNN	298.568 (51.941)	-0.011 (0.151)	10.874 (2.007)
RF	255.045 (31.077)	0.123 (0.197)	10.029 (0.826)
SVR	281.800 (44.624)	0.057 (0.014)	10.023 (0.949)
Ridge	288.446 (54.822)	0.001 (0.327)	10.350 (1.488)
Lasso	253.616 (29.377)	0.137 (0.143)	9.322 (1.048)
MTLasso	252.770 (34.322)	0.142 (0.128)	9.231 (1.186)
MLP-4	243.327 (30.834)	0.172 (0.146)	9.290 (1.668)
ATAN-1	238.585 (29.211)	0.191 (0.124)	9.017 (1.638)
ATAN-2	237.482 (31.289)	0.195 (0.127)	9.172 (1.658)

available in clinical practice. In this section we hence exclude features that are functional measures for the cardiovascular system and only use demographics and lab results as the input features, resulting in 34 features remain in the experiment. We follow the same experiment procedure as in the previous section and results are reported on 10 runs.

Table 15 shows the performance only using lab results and demographics. We see that neural network models (MLP-4, ATAN-1 and ATAN-2) have comparable performances and are better than other methods, implying that capturing high-level information would benefit predictive modeling. Comparing with Table 14, predictive performances overall degrade. This is possibly due to that functional measures are expected to be more infor-

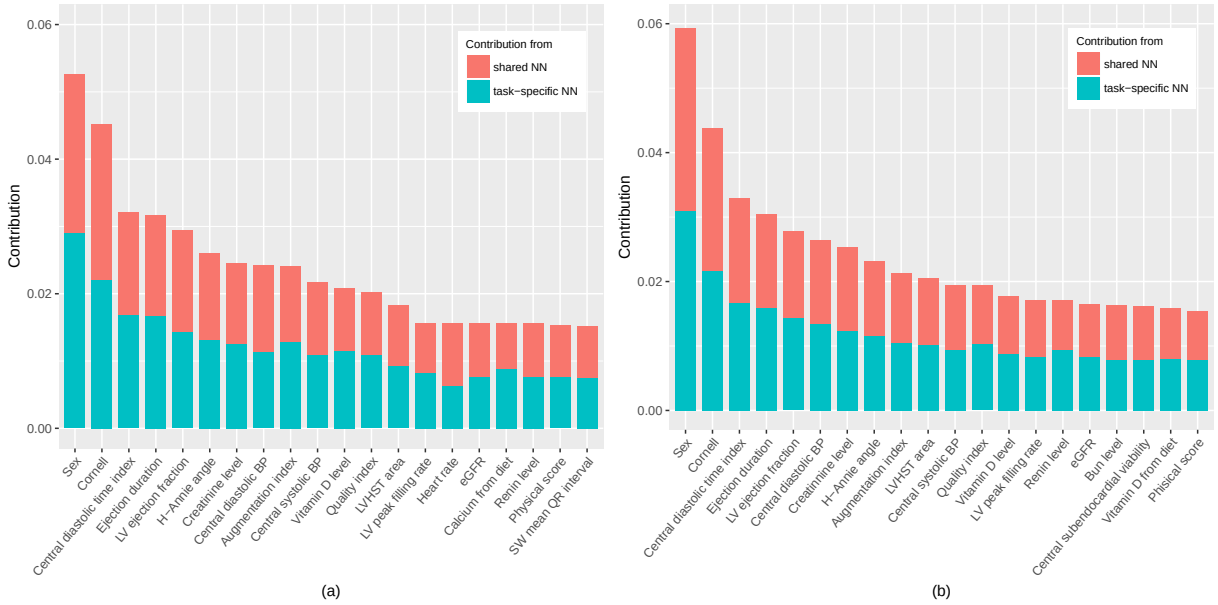

Figure 8: Top-20 important features for the complete set of features. Auxiliary target: (a) LVEDVI (b) posterior wall thickness.

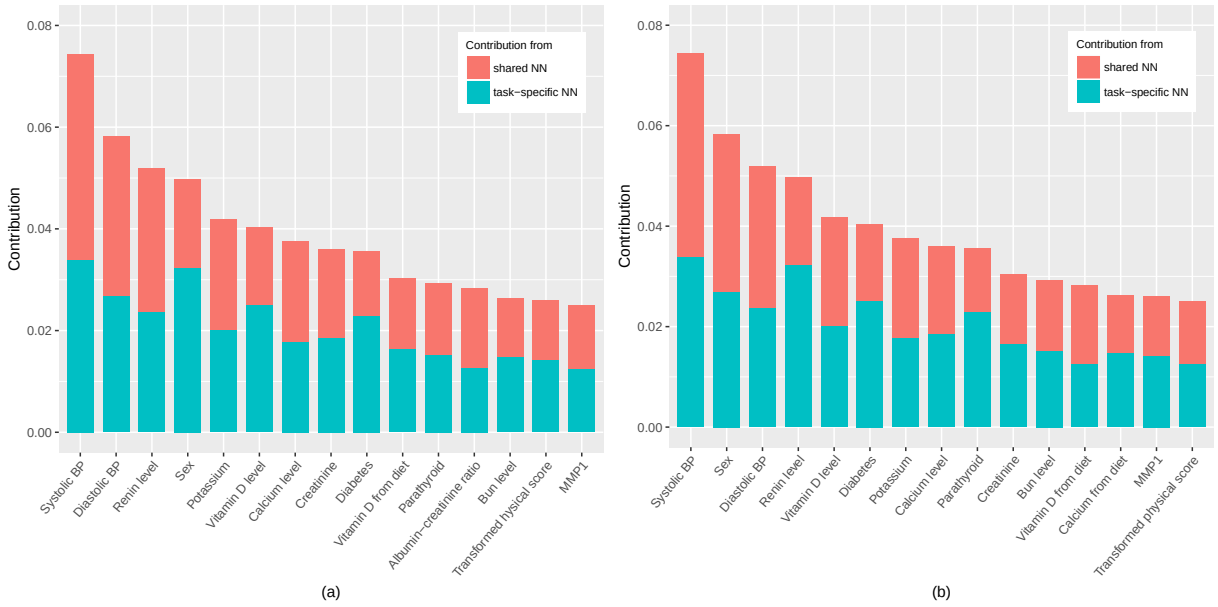

Figure 9: Top-15 important features for only lab results as features. Auxiliary target: (a) LVEDVI (b) posterior wall thickness.

mative for predicting LVMI.

Interpreting ATAN via Analyzing Weights Interpretability is as important as accuracy in clinical research. In this section, we use the heuristic approach proposed in Section 4.3.3

to calculate the contributions of features to targets, from which we can find risk factors for better understanding of disease progression.

Fig. 8 shows the top-20 features using the complete feature set. From the table, sex is indeed an important predictor: sample means of LVMI is 85.21 for female, and 95.78 for male; the two-sample t-test shows that the difference between female and male is significant with p-value less than 0.0001. Aside from sex (which already is known to be a key determinant of LVMI), several features with significant contributions are cardiac measures, such as ejection duration, LV ejection fraction. This is intuitive as heart structure and function are inherently related.

Fig. 9 presents the top-15 features out of demographics and lab results. We see from the figure that both systolic and diastolic blood pressure contributes most for predicting LVMI. The relationship between hypertension and LVH was the basic premise of this work, and the fact that elevated blood pressure corresponds with LVMI is not surprising [22]. However, our interest in this modeling exercise was to see if ATAN, could identify more subtle associations. Indeed, other contributory features from lab results were identified including renin, potassium, vitamin D, calcium, parathyroid hormone, creatinine et al. These top-ranked features accord with previous studies ([28, 49, 112]), supporting that this analysis of feature contribution through weight propagation provides a heuristically reasonable approach for interpreting DNN models.

4.4 Proposed DMNN

4.4.1 Deep Mixture of Neural Networks (DMNN)

DMNN Structure Although existing DNN models achieve state-of-art predictive performance, they don't explicitly take into considerations the heterogeneity in patient health

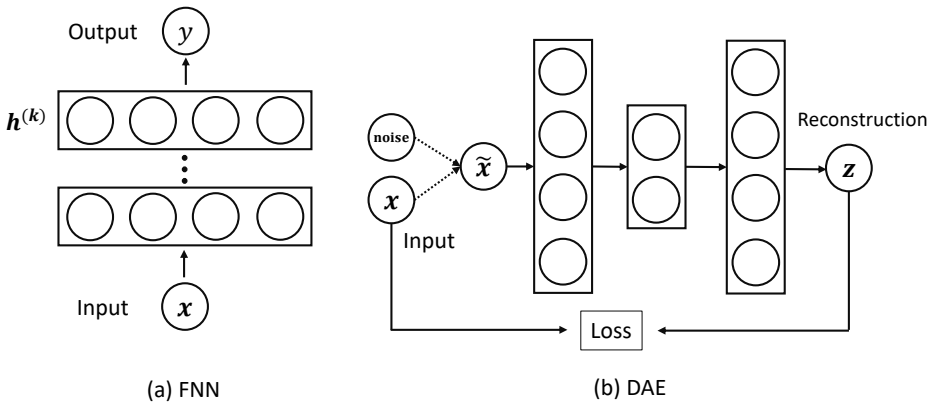

Figure 10: Deep neural network models FNN and denoise autoencoder based on FNN.

conditions (e.g subgroup structure), which could potentially hamper the model interpretation. However, determining patient subgroups usually require domain knowledge in the medical science which may not always be available for complex data. Hence, the goal of DMNN is not only to achieve comparable or better predictive performance but also enable the discovery of patient subgroups. In DMNN, the patient subgroups are defined that share the same functional input-output relations.

One challenge remaining in the partition of patient subgroups is determining which subgroup each patient belongs to. As such membership indicators can be viewed as latent variables in the modeling process, inspired by the classic mixture of experts and the recent deep unsupervised model, we use a DNN to learn feature representations from the input and then predict the membership indicator using softmax (e.g gating) based on the feature representations (termed as embedding network with gating, ENG). The learned features are then fed into multiple FNNs for prediction. However, the importance of those FNNs is gated by the membership indicator in the final loss function (termed as local predictive network, LPN). As such, patients with similar gating patterns are grouped and the FNN can capture the “local" input-outcome functional relation. Figure 11 displays an overview

of DMNN structure.

Mathematically, for each training example $(\boldsymbol{x}, y)$, ENG in DMNN outputs the feature representation $\boldsymbol{h}$ and a vector of gating values $\boldsymbol{g}$:

$$\boldsymbol{h}, \boldsymbol{g} = \text{ENG}(\boldsymbol{x}).$$

The each local predictive network LPN_i makes prediction $\hat{y}_i$ and we obtain its respect loss function L_i:

$$\hat{y}_i = \text{LPN}_i(\boldsymbol{h})$$

$$L_i(\boldsymbol{x}) = f(\hat{y}_i, y), \quad i = 1, \cdots, K$$

where f is the loss function: squared error $f(\hat{y}, y) = (\hat{y} - y)^2$ for regression and cross-entropy $f(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log \hat{y}))$ for classification.

The final loss function for training example $\boldsymbol{x}$ is the sum of loss functions of all K LPNs, weighted by the gating value $\boldsymbol{g} = (g_1, \cdots, g_k)$:

$$L_{final}(\boldsymbol{x}) = \sum_{i=1}^{K} g_i L_i(\boldsymbol{x}). \tag{4.9}$$

By minimizing the final loss, model parameters are learned when the loss function converges to a (local) minimum. Note that K is treated as hyperparameter and the optimal K is selected via cross-validation or validation data.

Subgroup identification in DMNN In DMNN framework, subgroups will be identified as the set of patients that share the similar functional input-output relation. In other

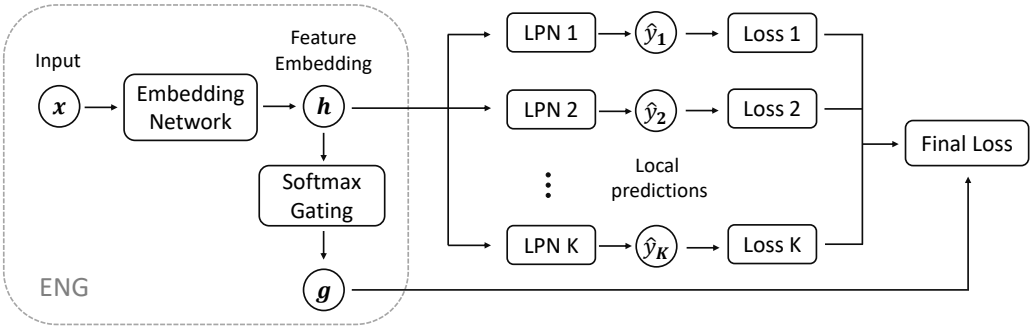

Figure 11: Overview of DMNN with ENG and and K LPNs.

words, clinical outcomes for patients within the same subgroup should be predicted well by some LPN. Intuitively, this implies that we can group patient according to the gating values $\boldsymbol{g}$: patient $\boldsymbol{x}$ belongs to subgroup k where $k = \arg\max_k\{g_i : i = 1, \cdots, g_K\}$, as a better LPN should have a larger gating value. Indeed, we can perform analysis similar to mixture of experts on the final loss function to see the rationale behind DMNN's subgroup identification.

From the final loss function Eq. (4.9), we see that each LPN is encouraged to fit each training sample well, e.g., low error value (though weighted differently). If one LPN captures the input-outcome relation well and gives less error than other LPNs, ENG in DMNN is encouraged to produce larger gating value for that specific LPN (simultaneously reduce other gating values due to the softmax function). Moreover, this will make the ENG embed similar patients closer to each other in the late feature space. Consequently, those patients will exhibit similar gating values. With the clustering effect of ENG, DMNN is capable of discovering patient subgroups.

4.4.2 Model Interpretation by Knowledge Distillation

Model interpretability is as important as accuracy in clinical research. While deep learning models are generally difficult to interpret, recent progress in knowledge distillation[51,

Table 16: Feature statistics (mean and standard deviations for continuous features, percentage for categorical features.). In the table, AA represents African Americans, F female and P positive.

Demo	Stats	Vitals	Stats	Labs	Stats	Hemodynamics	Stats
Age	59.15 (12.51)	Initial SBP	153.27 (33.06)	Troponin (P)	162 (48.4%)	DBP	75.87 (17.51)
Race (AA)	297 (88.7%)	Initial DBP	91.23 (20.73)	NP	3663.42 (6388.41)	SBP	127.86 (29.572)
Gender (F)	174 (51.8%)	Initial HR	91.56 (18.71)	Sodium	138.81 (4.15)	MAP	94.98 (21.01)
Weight	99 (30.92)	RR	20.66 (4.75)	BUN	27.00 (20.46)	dP/dt	602 .90 (258.89)
Height	172.85 (28.37)	OS	96.30 (4.55)	Creatinine	2.04 (2.47)	SVR	1615.43 (752.03)
		Temperature	97.95 (0.67)	eGFR	60.48 (31.20)	SV	66.05 (26.78)
				Hemaglobin	11.78 (2.35)	HR	85.74 (15.01)
						CO	5.55 (2.16)

2] for DNNs enables us to understand what DNNs learn from the data. The main idea is that after an accurate but complex DNN (teacher model) is trained, the knowledge can be transferred to train another simpler model (student model) by predicting the soft labels predicted by the teacher. Training with soft labels is an implicit regularization for the student model with which it can achieve as good performance as the teacher model [2, 10]. If the student model that learns knowledge from the teacher DNN model is interpretable, we can then interpret the DNN through the student model. For our DMNN model, we take the approach developed in Che et.al[10] for interpretation. We first train DMNN as the teacher model and then train the interpretable gradient boosting machine for regression (GBR) as the student model to mimic DMNN's predictive behavior. GBR uses the probability generated by DMNN as target. For each subgroup identified by DMNN, we train GBR and use the feature importance in GBR to identify important risk factors for that subgroup.

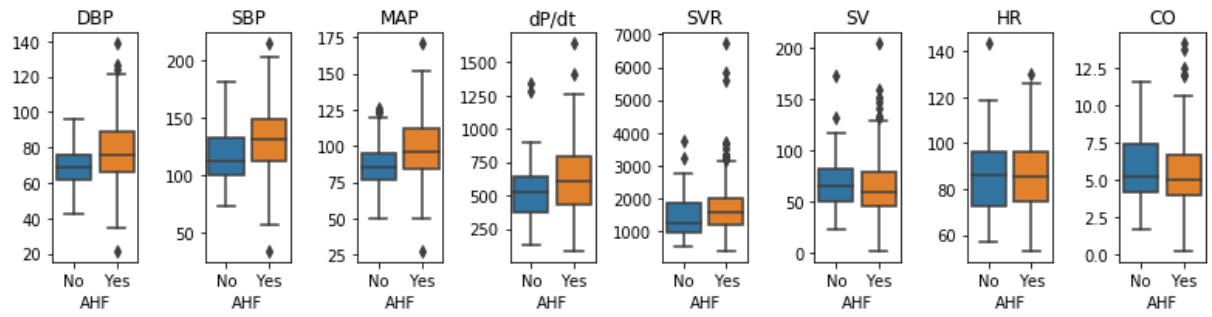

Figure 12: Box plot for AHF *v.s* non-AHF. It can be observed that AHF and non-AHF patients share some similar feature characteristics in hemodynamic features, yet AHF group exhibits larger variance. This implies large heterogeneity in patient health conditions for AHF onset.

4.4.3 Results and Discussions

In this section, we apply DMNN on a clinical dataset collected from patients presenting signs and symptoms of acute heart failure (AHF) in the emergency department (ED) of three urban academic medical centers in Detroit. The task is to predict whether a patient ultimately will be assigned a diagnosis of AHF. Our goal is two-fold: (1) accurately predict the risk of AHF, so that further actions can be effectively taken to avoid adverse outcomes; (2) identify the associated risk factors within the patient subgroups to promote the understanding of health disparities.

Data information and preprocessing The data contain health records for 335 patients with suspected AHF, among which 78% (261/335) of patients actually have AHF onset. There are 26 features in the data, including 5 demographics (age, race, gender, weight and height), 6 vital signs (initial SBP, initial DBP, initial heart rate (HR), respiratory rate (RR), oxygen saturation (OS) and temperature) when presenting in ED, 7 initial lab results (troponin, natriuretic peptide (NP), sodium, BUN, creatinine, eGFR and Hemaglobin), and 8 hemodynamic features measured using a non-invasive device (SBP, DBP, MAP, dP/dt,

SVR, SV, HR and CO). Table 16 shows the details of feature characteristics. Figure 12 is the boxplot of hemodynamic features for comparing AHF against non-AHF patients. From the figure, we can see that while non-AHF and AHF patients share some similar statistics (such as median value, 25th and 75th quantiles), AHF patients show larger variance in hemodynamic features compared with non-AHF patients . This observation implies the existence of large heterogeneity in health conditions among patients in the ED who present with suspected AHF.

For missing values in the dataset (missingness is about 0.6%), we impute them with mean values for continuous features and the majority value for categorical features. Note that the imputation in our experiments is based on training data to prevent possible information leak to testing data (after train/test split). As features have different scales, we also perform data normalization for features to have zero mean and unit variance.

Implementation and evaluation details We implement DMNN using Pytorch. In the experiment, DMNN is of depth 4, consisting of the input layer, two hidden layers of size 40 and 20 respectively which act as the ENG part in DMNN and multiple output layers; the embedding of the 2nd hidden layer will be fed into the softmax layer to obtain gating values and LPNs for predictions which are linear models. Sigmoid function is used in hidden layers for non-linear activation. By initial experiments on the number K of LPNs, we found $K = 2$ work well with this relatively small dataset. Since deep neural network can be easily overfitted and the gating mechanism may degenerate (i.e model may only use one LPN), we apply unsupervised learning techniques to initialize the ENG in DMNN. We first train a 5-layer (dimension 26-40-20-40-26) denoise autoencoder (DAE) and use the encoder to initialize embedding part of ENG; after DAE is trained, we extract the

bottleneck feature representations and use K-means for clustering; we then train the ENG to predict the cluster labels. With ENG initialized, DMNN is trained via stochastic gradient descent in conjunction with L2 regularization.

We test different machine learning models for performance evaluation. Those baseline models include logistic regression (LR), decision trees (DT), K-nearest neighbor (KNN), gradient boosting machine (GBM), feedforward neural network(FNN) of the same hidden dimensions, random forest (RF). We use Python scikit-learn package for model implementation. In the experiment, training data are divided into training/testing by a split 85%/15%. Within the training data, we further split out 10% as validation data for selecting model parameters. The evaluation metrics are the area under receiver operating characteristic curve (AUROC) and area under precision-recall curve (AUPRC). We repeat the train/test procedure 5 times and the average predictive performance on the testing data is reported.

Predictive performance Table 17 shows the predictive performance on the testing data. We see from the table that DMNN performs better than other baseline models. As shown in Figure 13, patients can be clustered into two subgroups. In contrast with baselines that only build a global predictive model for all patients, DMNN builds a local predictive model for each patient subgroup that is able to capture the local functional input-output relations, resulting in performance gain. We also observe that all models achieve good performance in terms of AUPRC and relatively worse performance for AUC. This is due to that the majority of the patients (78%) have AHF onset, and all models can predict AHF well at the cost of misclassifying non-AHF patients as AHF. From Figure 13, we see that there is a large overlap between AHF and non-AHF patients; from Figure 12, non-

Table 17: Average AUC and AUPRC on testing data along with standard deviations.

	LR	GBM	DT	KNN	FNN	RF	DMNN
AUC	0.69 (0.09)	0.70 (0.07)	0.58 (0.08)	0.66 (0.04)	0.69 (0.05)	0.71 (0.06)	**0.74** (0.07)
AUPRC	0.88 (0.05)	0.90 (0.03)	0.81 (0.03)	0.85 (0.04)	0.89 (0.02)	0.90 (0.01)	**0.92** (0.03)

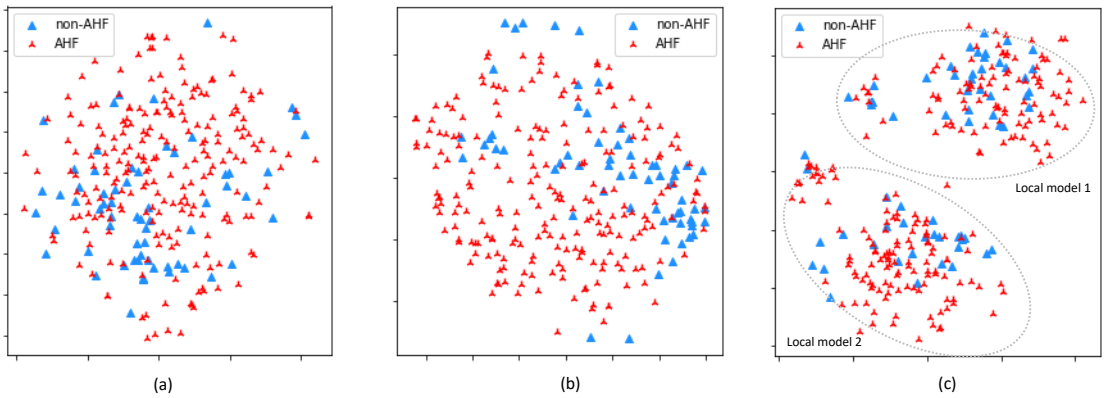

Figure 13: 2D t-SNE plot. (a) Raw input features; (b) feature embedding from FNN; (c) feature embedding from DMNN; DMNN feature embedding exhibits two patient subgroups; a local model is fitted for each subgroup.

AHF and AHF patients have similar characteristics for hemodynamic features. Those two observations imply that patients in ED with possible AHF are similar, yet those with the highest likelihood of diagnosis are rather different. This heterogeneity makes it difficult for models to differentiate AHF and non-AHF patients effectively, leading to a lower AUC score compared with AUPRC score.

Feature analysis We interpret the DMNN model via knowledge distillation as introduced in Section 4.4.2. To do so, the interpretable gradient boosting machine for each patient subgroup is trained to learn the input-output relation captured by DMNN. As GBM mimics the predictive behavior of DMNN, we can then identify risk factors and their dependence relations to the onset of AHF. Figure 14 shows the top8 features that are important for predicting AHF. We see that both DMNN and GBM share some features such as blood

pressure (initial SBP, initial DBP, SBP and DBP), indicating BP is a universal risk factor for AHF. As shown in Table 16, blood pressure level (initial SBP 153.27mm Hg, initial DBP 91.23mm Hg, SBP 127.86mm Hg and DBP 75.87mm Hg) indicates that patients with elevated values are more likely to be diagnosed with AHF. Comparing DMNN with GBM, important risk factors for DMNN are rather different from those of GBM. For example, natriuretic peptide (NP) identified in GBM is the single most important risk factor for the diagnosis of AHF. However in DMNN, NP is only important in Subgroup 2 and not very important in terms of predictive power in Subgroup 1. While NP level is informative in the diagnosis of heart failure, the difference in NP importance in those models possibly implies large disparity in the development of AHF. Within DMNN, both subgroups also share MAP as an important feature yet have subgroup specific features. In Subgroup 1, SV, dP/dt and HR are important hemodynamic features whereas in Subgroup 2, SVR and RR are identified important.

Discussions To further understand the dependence relation between features and AHF in patient subgroups, the partial dependence plots (PDP) for five important features (initial DBP, MAP, RR, SV, SVR) in either Subgroup 1 or 2 are shown in Figure 15. Both subgroups follows a similar dependence relation for initial DBP and MAP: as their level increases, the risk of AHF also increases. But for RR, SV and SVR, the dependence relation differs. In Subgroup 2, RR and SVR have a positive dependence relation on AHF onset while no such relation in Subgroup 1. In Subgroup 1, patients are at high risk of AHF if SV level is too low or too high; whereas in Subgroup 2, SV is not very predictive for AHF. As DMNN performs well in AHF prediction as shown in Table 17, the difference between two subgroups provides useful information for clinicians in disease diagnosis.

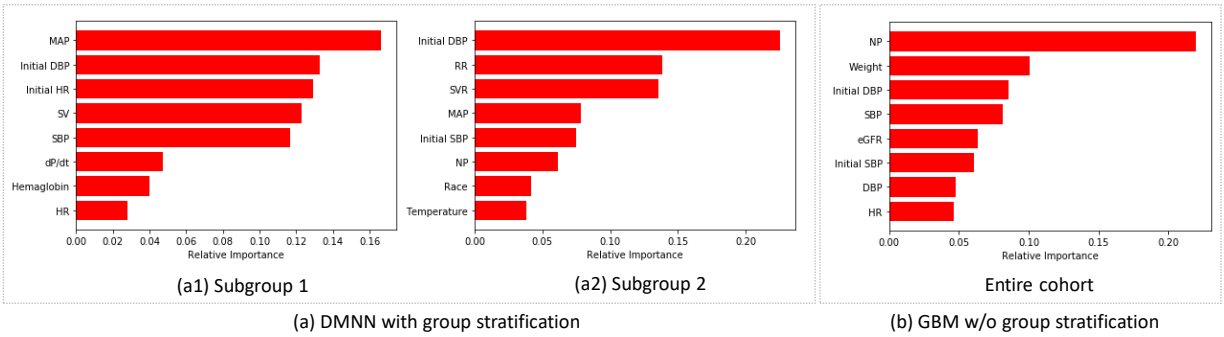

Figure 14: Top 8 important features.

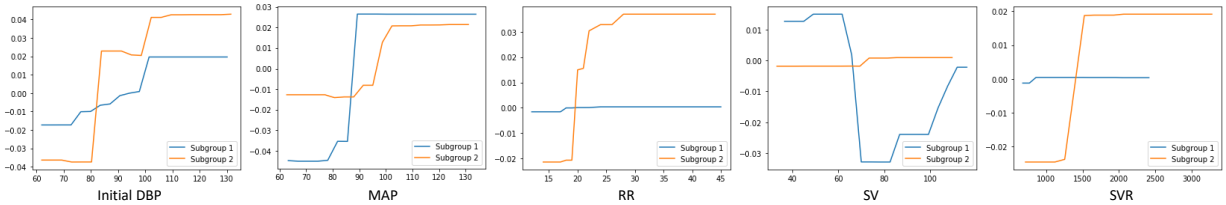

Figure 15: Partial dependence plot for important features for Subgroup 1 or 2.

4.5 Conclusion

In this chapter, we present two novel DNN predictive models, ATAN and DMNN. ATAN introduces multi-task learning as a regularization method. ATAN learns high-level latent information from low-level input features, as well as flexibly leveraging other information contained in the clinically relevant targets. Our experiments with one auxiliary target show that DNNs can offer great improvements for predictive modeling in clinical research when only limited labeled data are available. Moreover, ATAN can be easily extended to multiple auxiliary targets. However, ATAN does not exploit the information from the unlabeled data. Hence, for future work, we plan to develop DNN models combining multi-task learning paradigm with semi-supervised learning, which fully exploits different sources of information for better predictive performance. DMNN identifies patient subgroup via gating mechanism that aims at capturing similar functional input-output relations among

patient population. With subgroup discovery, DMNN can identify subgroup-specific risk factors (in terms of AHF prediction) and such granularity can potentially help clinicians understand subgroup differences, which is an advantage of DMNN compared with traditional “one-size-fits-all" approaches. Experiments on an AHF prediction task show that our proposed method can achieve state-of-art performance and discover risk factors that might be missed by traditional methods. One limitation of this work is that DMNN is not able to characterize patient subgroup (i.e., phenotyping). Hence, for future works, we will expand our work to incorporate information from multimodal data such as medical images, clinical notes and time series data to further improve DMNN for better characterization of patient subgroups.

CHAPTER 5 IN-NEGATIVE-CLASS REWEIGHTED LOGISTIC LOSS

5.1 Introduction

Deep convolutional neural networks (CNNs) trained with logistic or softmax losses (LGL and SML respectively for brevity), e.g., logistic or softmax layer followed by cross-entropy loss, have achieved remarkable success in various visual recognition tasks [70, 68, 47, 130, 141]. The success mainly accredits to CNN's merit of high-level feature learning and loss function's differentiability and simplicity for optimization. When training data exhibit class imbalances, training CNNs with gradient descent is biased towards learning majority classes in the conventional (unweighted) loss, resulting in performance degradation for minority classes. To remedy this issue, the class-wise reweighted loss is often used to emphasize the minority classes that can boost the predictive performance without introducing much additional difficulty in model training [20, 54, 94, 152]. A typical choice of weights for each class is the inverse-class frequency.

A natural question then to ask is **what roles are those class-wise weights playing in CNN training using LGL or SML that lead to performance gain?** Intuitively, those weights make tradeoffs on the predictive performance among different classes. In this chapter, we answer this question quantitatively in a set of equations that tradeoffs are on the model predicted probabilities produced by the CNN models. Surprisingly, effectiveness of the reweighting mechanism for LGL is rather different from SML. Here, we view the conventional (e.g., no reweighting) LGL or SML as a special case where all classes are weighted equally.

As these tradeoffs are related to the logistic and softmax losses, answering the above question actually leads us to answering a more fundamental question about their learning

behavior: **what is the property that the decision boundary must satisfy when models are trained?** To our best knowledge, this question has not been investigated systematically, despite logistic and softmax losses are extensively exploited in deep leaning community.

While SML can be viewed as a multi-class extension of LGL for binary classification, LGL is a different learning objective when used in multi-class classification [5]. From the perspective of learning structure of data manifold as pointed out in [3, 5, 24], SML treats all class labels equally and poses a competition between true and other class labels for each training sample, which may distort data manifold; for LGL, the one-*vs.*-all approach it takes avoids this limitation as it models each target class independently, which may better capture the in-class structure of data. Though LGL enjoys such merits, it is rarely adopted in existing CNN models. The property that LGL and SML decision boundaries must satisfy further reveals the difference between LGL and SML (see Eq. (5.9), (5.10) with analysis). If used for the multi-class classification problem, we can identify two issues for LGL. Compared with SML, LGL may introduce data imbalance, which can degrade model performance as sample size plays an important role in determining decision boundaries. More importantly, since the one-*vs.*-all approach in LGL treats all other classes as the negative class, which is of a multi-modal distribution [78, 80], the averaging effect of the predicted probabilities of LGL can hinder learning discriminative feature representations to other classes that share some similarities with the target class.

5.2 Related Work

With recent explosion in computational power and availability of large scale image datasets, deep learning models have repeatedly made breakthroughs in a wide spectrum of tasks in computer vision [70, 37]. Those advancements include new CNN architectures for

image classification[68, 47, 130, 141], objective detection and segmentation [121, 122], new loss functions [24, 165] and effective training techniques to improve CNN performance [131, 58].

In those supervised learning problems, CNNs are mostly trained with loss functions such as LGL and SML. In practice, class imbalance naturally emerges in real-world data and training CNN models directly on those datasets may lead to poor performance. This phenomenon is referred as the imbalanced learning problem [46]. To tackle this problem, cost-sensitive method [29, 166] is the widely-adopted approach in current training practices as they don't introduce any obstacles in the backpropagation algorithm. One of the most popular methods is class-wise reweighting loss function based on LGL and SML. For example, [54, 152] reweight each class by its inverse-class frequency. In some long-tailed datasets, a smoothed version of weights is adopted [94, 97], which emphasizes less on minority classes, such as the square root of inverse-class frequency. More recently, [20] proposed a weighting strategy based on the calculation of effective sample size. In the context of learning from noisy data, [165] provides analysis on the weighted SGL showing close connection to the mean absolute error (MAE) loss. However, what role class-wise weights play in LGL and SML is not explained in previous works. In this chapter, we provide a theoretical explication on how the weights control the tradeoffs among model predictions.

If we decompose the multi-class classification as multiple binary classification sub-tasks, LGL can also be used as the objective function via one-*vs.*-all approach [45, 5], which is however rarely adopted in existing works of deep learning. Motivated to understand class-wise reweighted LGL and SML, our analysis further leads us to a more profound discovery

in the properties of decision boundaries for LGL and SML. Previous work in [24] showed that the learning objective using LGL is quite different from SML as each class is learned independently. They identified the negative class distraction (NCD) phenomenon that might be detrimental to model performance when using LGL in multi-class classification. From our analysis, the NCD problem can be partially explained that LGL treats the negative class (e.g., non-target classes) as a single class and ignores its multi-modality. If there exists one non-target class that share some similarity with the target class, CNN trained with LGL may make less confident predictions for that non-target class (e.g., probability of belonging to the negative class is small) as its predicted probabilities are averaged out due to other non-target classes with confident predictions. Consequently, samples from that specific non-target class can be misclassified into the target class, resulting in large predictive error.

5.3 Analysis on LGL and SML

In this section, we provide a theoretical explanation for the class-wise weighting mechanism and depict the learning property of LGL and SML losses.

Notation Let $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ be the set of training samples of size N, where $\boldsymbol{x}_i \in \mathbf{R}^p$ is the p-dimensional feature vector and $y_i = k (k = 0, \cdots, K-1)$ is the true class label, and $S_k = \{(\boldsymbol{x}_i, y_i) : y_i = k\}$ the subset of D for the k-th class. The bold $\boldsymbol{y}_i = (y_i^0, \cdots, y_i^{K-1})$ is used to represent the one-hot encoding for y_i: $y_i^k = 1$ if $y_i = k$, 0 otherwise. $N_k = |S_k| (k = 0, \cdots, K-1)$ is used to represent sample size for the k-th class and hence $\sum_k N_k = N$. The maximum size is denoted as $N_{\max} = \max_{k=0,\cdots,K-1}(N_k)$.

5.3.1 Preliminaries

For classification problem, the probability for a sample $\boldsymbol{x}$ belonging to one class is modeled by logistic (e.g., sigmoid) for binary classification

$$p(y=1|\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{1+\exp(-z)},$$

$$p(y=0|\boldsymbol{x};\boldsymbol{\theta}) = 1 - p(y=1|\boldsymbol{x}),$$

and by softmax for multi-class classification

$$p(y=k|\boldsymbol{x};\boldsymbol{\theta}) = \frac{\exp(z_k)}{\sum_{j=0}^{K-1}\exp(z_j)},$$

where all z's are the logits for $\boldsymbol{x}$ modeled by CNN with parameter vector $\boldsymbol{\theta}$. It is worth noting that softmax is equivalent to logistic in binary classification as can be seen from

$$p(y=1|\boldsymbol{x}) = \frac{\exp(z_1)}{\exp(z_0)+\exp(z_1)} = \frac{1}{1+\exp(-(z_1-z_0))}.$$

Hence, without loss of generality, we write class-wise reweighted LGL ($K=2$) and SML ($K \geq 3$) in a unified form as follows

$$L(\boldsymbol{\theta}) = -\sum_{k=0}^{K-1}\lambda_k \sum_{i_k \in S_k} \log f_k(\boldsymbol{\theta};\boldsymbol{x}_{i_k}) = -\sum_{k=0}^{K-1}\lambda_k L_k(\boldsymbol{\theta}), \tag{5.1}$$

where each $f_k(\boldsymbol{\theta};\boldsymbol{x}_i) = p(y_i=k|\boldsymbol{x}_i)$ is the CNN predicted probability of sample $\boldsymbol{x}_i$ belonging to the k-th class; λs are weight parameters to control each class's contribution in the

loss. When all λs are equal, $L(\boldsymbol{\theta})$ is the conventional cross-entropy loss and minimizing it is equivalent to maximizing likelihood. If the training data are imbalanced, a different setup of λs is used, usually classes with smaller sizes are assigned with higher weights. Generally, λs are treated as hyperparameters and selected by cross-validation.

We emphasize here that using logistic function for multi-class ($K \geq 3$) is a different learning objective from softmax in this case as the classification problem is essentially reformulated as K binary classification sub-problems.

5.3.2 Key Equations for Weights λs

Assume that CNN's output layer, after convolutional layers, is a fully connected layer of K neurons with bias terms, then the predicted probability for sample $\boldsymbol{x}$ is given by the softmax activation:

$$f_k(\boldsymbol{x}) = \frac{\exp(\boldsymbol{W_k h_x} + b_k)}{\sum_{j=1}^{K} \exp(\boldsymbol{W}_j \boldsymbol{h_x} + b_j)} \quad (k = 0, \cdots, K-1), \tag{5.2}$$

where $\boldsymbol{h_x}$ is the feature representation of $\boldsymbol{x}$ extracted from convolutional layers, $\boldsymbol{W}_k$ and b_k are parameters of the k-th neuron in the output layer. For notational simplicity, we have dropped $\boldsymbol{\theta}$ in $f_k(\boldsymbol{x})$.

After CNN is trained, we assume that the reweighted SML $L(\boldsymbol{\theta})$ is minimized to local optimum $\boldsymbol{\theta}^*$. By optimization theory, a necessary condition is that the gradient of $L(\boldsymbol{\theta})$ is zero at $\boldsymbol{\theta} = \boldsymbol{\theta}^*$[5]:

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \mathbf{0} \iff \sum_{k=1}^{K} \lambda_k \frac{\partial L_k}{\partial \boldsymbol{\theta}} = \mathbf{0}. \tag{5.3}$$

We specifically consider $L_1(\boldsymbol{\theta})$ for the 1-st class with respect to one component η of $\boldsymbol{\theta}$. Then

[5]More strictly, zero is in the subgradient of $L(\boldsymbol{\theta})$ at $\boldsymbol{\theta}^*$. But this doesn't affect the following analysis.

with chain rule, the necessary condition above gives:

$$\begin{aligned}
&\lambda_1 \frac{\partial L_1}{\partial \eta} + \sum_{k=2}^{K} \lambda_k \frac{\partial L_k}{\partial \eta} = 0 \iff \\
&\lambda_1 \sum_{i_1 \in S_1} \frac{1}{f_{1,i_1}} \frac{\partial f_{1,i_1}}{\partial \eta} + \sum_{k=2}^{K} \lambda_k \sum_{i_k \in S_k} \frac{1}{f_{k,i_k}} \frac{\partial f_{k,i_k}}{\partial \eta} = 0,
\end{aligned} \tag{5.4}$$

where we use $f_{j,i_k} = f_j(\boldsymbol{x}_{i_k})$ given by Eq. (5.2).

Let $\boldsymbol{\sigma}(\boldsymbol{z})$ be the softmax function of $\boldsymbol{z} = (z_1, \cdots, z_K)$ with each component $\boldsymbol{\sigma}(z_k) = \exp(z_k)/\sum_i \exp(z_i)$, its derivative is

$$\frac{\partial \boldsymbol{\sigma}(z_k)}{\partial z_i} = \begin{cases} \boldsymbol{\sigma}(z_k)(1 - \boldsymbol{\sigma}(z_k)), & i = k \\ -\boldsymbol{\sigma}(z_k)\boldsymbol{\sigma}(z_i), & i \neq k. \end{cases} \tag{5.5}$$

Denoting $a_{j,i_k} = \boldsymbol{W_j} \boldsymbol{h}_{\boldsymbol{x}_{i_k}} + b_j$ as the j-th logit in Eq. (5.2) for sample $\boldsymbol{x}_{i_k}$, then $f_{j,i_k} = \boldsymbol{\sigma}(a_{j,i_k})(j = 0, \cdots, K-1)$. Again with chain rule and Eq. (5.5):

$$\begin{aligned}
\frac{\partial f_{k,i_k}}{\partial \eta} &= \sum_{j=1}^{K} \frac{\partial f_{k,i_k}}{\partial a_{j,i_k}} \frac{\partial a_{j,i_k}}{\partial \eta} \\
&= f_{k,i_k}(1 - f_{k,i_k}) \frac{\partial a_{k,i_k}}{\partial \eta} - f_{k,i_k} \sum_{j \neq k} f_{j,i_k} \frac{\partial a_{j,i_k}}{\partial \eta}.
\end{aligned} \tag{5.6}$$

Since Eq. (5.4) holds valid for any component η of $\boldsymbol{\theta}$, we specifically consider the case when $\eta = b_1$. Therefore we have $\partial a_{1,i_k}/\partial b_1 = 1$ and $\partial a_{j,i_k}/\partial b_1 = 0 (j = 2, \cdots, K)$. Then Eq. (5.6) becomes:

$$\frac{\partial f_{k,i_k}}{\partial b_1} = \begin{cases} f_{1,i_1}(1 - f_{1,i_1}), & k = 1 \\ -f_{k,i_k} f_{1,i_k}, & k \neq 1. \end{cases} \tag{5.7}$$

Plug Eq. (5.7) back into Eq. (5.4) and rearrange the terms, we have

$$\lambda_1 \sum_{i_1 \in S_1} (1 - f_{1,i_1}) = \sum_{k=2}^{K} \lambda_k \sum_{i_k \in S_k} f_{1,i_k}. \tag{5.8}$$

With the same calculations, we can obtain other $K-1$ similar equations, each of which corresponds to one class. Remember f_{j,i_k} is the probability of sample $\boldsymbol{x}_{i_k}$ from the k-th class being predicted into the j-th class, and Eq. (5.8) reveals the quantitative relation between weights λs, model predicted probabilities and training samples. Notice that CNN is often trained with L_2 regularization to prevent overfitting. If the bias term b_ks are not penalized, Eq. (5.8) still holds valid. Another possible issue is that the calculation relies on the use of bias terms b_k in the output layer. As using bias increases CNN's flexibility and is not harmful to CNN performance, our analysis is still applicable to a wide range of CNN models trained with cross-entropy loss.

We observe in Eq. (5.8), $\sum_{i_1 \in S_1}(1 - f_{1,i_1})/N_1$ (approximately) represents the expected probability of CNN incorrectly predicting a sample of class 1 and $\sum_{i_k \in S_k} f_{1,i_k}/N_k$ the expected probability of CNN misclassifying a sample of class $k (k \neq 1)$ into class 1. If we assume that the training data can well represent the true data distribution that testing data also follow, the learning property of trained CNN shown in Eq. (5.8) can be generalized to testing data.

More specifically, since the CNN model is a continuous mapping and the softmax output is bounded between 0 and 1, by the uniform law of large numbers [101], we have the

following system of K equations *once CNN is trained*:

$$\begin{cases} \lambda_0 N_0(1-\bar{p}_{0\to 0}) \approx \sum_{k\neq 0} \lambda_k N_k \bar{p}_{k\to 0} \\ \qquad\qquad \vdots \\ \lambda_{K-1} N_{K-1}(1-\bar{p}_{K-1\to K-1}) \approx \sum_{k\neq K-1} \lambda_k N_k \bar{p}_{k\to K-1}, \end{cases} \tag{5.9}$$

where for indices i and j, $\bar{p}_{i\to j}$ represents the expected probability of CNN predicting a sample from class i into class j:

$$\bar{p}_{i\to j} = \mathrm{E}_{\boldsymbol{x}\sim P(\boldsymbol{x}|y=i)} f_j(\boldsymbol{x}),$$

where $P(\boldsymbol{x}|y=i)$ is the true data distribution for the i-th class.

Binary Case with LGL For binary classification problem ($K=2$), Eq. (5.9) gives us the following relation about CNN predicted probabilities:

$$\frac{1-\bar{p}_{0\to 0}}{\bar{p}_{1\to 0}} \approx \frac{\lambda_1 N_1}{\lambda_0 N_0}. \tag{5.10}$$

- In the conventional LGL where each class is weighted equally ($\lambda_0 = \lambda_1$), Eq. (5.10) becomes $1-\bar{p}_{0\to 0} = N_1 \bar{p}_{1\to 0}/N_0$. If data exhibit severe imbalance, say $N_0 = 10N_1$, then we must have ($\bar{p}_{1\to 0} < 1$)

$$\bar{p}_{0\to 0} = 1 - \frac{\bar{p}_{1\to 0}}{10} > 0.9.$$

If $t = 0.5$ is the decision making threshold, this implies that the trained neural net-

work can correctly predict a majority class (e.g., class 0) sample, confidently (at least) with probability 0.9, on average. However, for minority class, the predictive performance is more complex which depends on the trained model and data distribution. For example, if two classes can be well separated and the model made very confident predictions, say $\bar{p}_{0\to 0} = 0.98$, then we must have $\bar{p}_{1\to 1} = 0.8$ for the minority class, implying a good predictive performance on class 1. If $\bar{p}_{0\to 0} = 0.92$, then we have $\bar{p}_{1\to 1} = 0.2$. This means the predicted probability of a minority sample being minority is 0.2 on average. Hence, the classifier must misclassify most minority samples ($0.2 < 0.5$), resulting in very poor predictive accuracy for minority class.

- If LGL is reweighted using inverse-class frequencies, $\lambda_0 = 1/N_0$ and $\lambda_1 = 1/N_1$, the equation above is equivalent to $\bar{p}_{0\to 0} = 1 - \bar{p}_{1\to 0} = \bar{p}_{1\to 1}$. Since predictions are made by $y = \arg\max_i f_i(\boldsymbol{x})$ and $f_1(\boldsymbol{x}) > f_0(\boldsymbol{x})$ means $f_1(\boldsymbol{x}) > 0.5$, we can have a deterministic relation: if either class 0 or 1 can be well predicted (e.g., $\bar{p}_{i\to i} > 0.5$), reweighting by class inverse frequencies can guarantee performance improvement for the minority class. However, the extent of “goodness" depends on the separability of the underlying data distributions of the two classes.

Simulations for Eq. (5.10) We conduct simulations under two settings for checking Eq. (5.10). The imbalance ratio is set to 10 in training data ($N_0 = 1000, N_1 = 10000$), testing data size is $(1000, 1000)$; both training and testing data follow the same data distribution. As the property only relies on the last fully connected hidden layer, we use the following setup:

- Sim1: $P_1(x|y = 1) = \mathcal{N}(-1.5, 1) + \mathcal{U}(0, 0.5)$, $P_2(x|y = 0) = \mathcal{N}(1.5, 1) + \mathcal{U}(-0.5, 0)$.

λ_0	$\frac{1}{2}$	$\frac{N_0}{N_0+N_1}$	$\frac{2N_0}{(2N_0+N_1)}$
RHS	10	1	0.5
LHS (Sim1)	10.05 (1.13)	1.00 (0.09)	0.50 (0.04)
LHS (Sim2)	10.12 (0.67)	1.01 (0.05)	0.50 (0.03)

Table 18: Simulation results (along with standard deviation) for Eq. (5.10) over 100 runs, $\lambda_1 = 1 - \lambda_0$. RHS represents theoretical value on the right-hand side of (5.10); LHS the simulated value on the left hand side.

Logistic regression is fitted. $\mathcal{N}$ and $\mathcal{U}$ represents normal and uniform distribution respectively.

- Sim2: $P_1(\boldsymbol{x}|y=1) = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\sigma}_1)$, $P_0(\boldsymbol{x}|y=0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0)$, where $\boldsymbol{\mu}_1 = (0,0,0)$, $\boldsymbol{\mu}_0 = (1,1,1)$, $\boldsymbol{\sigma}_1 = 1.2\boldsymbol{I}$, $\boldsymbol{\sigma}_0 = \boldsymbol{I}$. A one-hidden-layer forward neural network of layer size $(3, 10, 1)$ with sigmoid activation.

Table 18 shows simulation results under three λ settings. We see from the Table that simulated values match with the theoretical values accurately, demonstrating the correctness of Equation (5.10).

Multi-class Case with SML Because $\sum_k \bar{p}_{i\to k} = 1$ and Eq. (5.9) has $K(K-1)$ variables with only K equations, we can't exactly solve it quantitatively for a relation among those $\bar{p}_{i\to j}$'s when $K > 2$. For the special case when weights are chosen as the inverse-class frequencies $\lambda_k = 1/N_k$, considering for class 1, we have $(1 - \bar{p}_{1\to 1}) \approx \sum_{k\neq 1} \bar{p}_{k\to 1}$. Multi-class classification $(K > 2)$ does not have a deterministic relation as in the binary case, as predictions are made by $y = \arg\max_i f_i(\boldsymbol{x})$ and we don't have a decisive threshold for decision making (like the 0.5 in binary case). Our findings match the results in [166] in the sense that class-wise reweighting for multi-class is indeterministic. However, our results are solely based on the mathematical property of the backpropagation algorithm

from optimization theory whereas [166] is based on decision theory.

Learning property of LGL and SML As the class-wise reweighting mechanism is explained in Eq. (5.9), those equations also reveal the property of decision boundaries for LGL and SML. For comparison, the decision boundary of support vector machine (SVM) [19] is determined by those support vectors that maximize the margin and those samples with larger margin have no effects on the position of decision boundary. On the contrary, all samples have their contribution to the decision boundary in LGL and SML so that their averaged probabilities that the model produces must satisfy Eq. (5.9). In particular for the binary case, we can see that if classes are balanced, the model must make correct predictions with equal confidence for the positive and negative classes, on average; whereas for imbalanced data, the decision boundary will be pushed towards the minority class in a position with Eq (5.10) always maintained. Another observation is that if the expectation of model predicted probabilities doesn't match with its mode (e.g skewed distribution), the magnitude of tradeoff between performance of the majority and minority class depends on the direction of skewness. If the distribution of the majority class skews away from the decision boundary, upweighting minority class will boost model performance at a small cost of performance degradation for the majority class than if it skews towards the decision boundary. This implies that estimating the shape of data distribution in the latent feature space and choosing the weights accordingly would be very helpful to improve model overall performance.

5.4 Proposed Approach: In-negative Class Reweighted LGL

In this section, we focused on LGL for multi-class classification via one-*vs.*-all approach. In addition to the theoretical merits of LGL mentioned in the introduction section that LGL

is capable of better capturing the structure of data manifold than SML, the guarantee of achieving good performance after properly reweighting (e.g., Eg.(5.10)) is also desirable as the one-*vs.*-all approach naturally introduces data imbalance issue.

Multi-modality Neglect Problem In spite of those merits of LGL, it also introduces the multi-modality neglect problem for multi-class classification. Since the expectation of model predicted probability must satisfy Eq (5.10) for LGL, the averaging effect might be harmful for model performance. In the one-*vs.*-all approach, the negative class consists of all the remaining non-target classes, which follows a multi-modal distribution (one modality for each non-target class). LGL treats all non-target classes equally in the learning process. If there is a hard non-target class that shares non-trivial similarity with the target class, its contribution in LGL might be averaged out by other easy non-target classes. In other words, those easy non-target classes (e.g., correctly predicted as the negative class with high probabilities) would compensate the predicted probability of the hard non-target class so that the probabilistic relation in Eq (5.10) is maintained. Consequently, model could incorrectly predict samples from the hard non-target class into the target class, inducing large predictive error for that class. This phenomenon is not desirable as we want LGL to pay more attention on the separation of the target-class with that hard class, meanwhile maintain the separation from the remaining easy non-target classes.

To this end, we propose an improved version of LGL to reweight each non-target class's contribution within the negative class. Specifically, for the target class k (e.g., positive class, labeled as $y = 1$) and all non-target classes (e.g., negative class, labeled as $y = -1$), a two-level reweighting mechanism is applied in LGL, which we term as **in-negative-class**

reweighted LGL (LGL-INR):

$$
\begin{aligned}
L_k^{\text{INR}}(\boldsymbol{\theta}) = &- \frac{1}{N_k} \sum_{\boldsymbol{x} \in S_k} \log p(y=1|\boldsymbol{x};\boldsymbol{\theta}) \\
&- \sum_{j=0, j\neq k}^{K-1} \lambda_j \frac{1}{N_j} \sum_{\boldsymbol{x} \in S_j} \log(1 - p(y=1|\boldsymbol{x};\boldsymbol{\theta})),
\end{aligned}
\tag{5.11}
$$

where $p(y=1|\boldsymbol{x};\boldsymbol{\theta})$ is the predicted probability of sample $\boldsymbol{x}$ belonging to the positive class and λ_j is the weight for class j as a sub-class of the negative class.

The first reweighting is at the level of positive *vs.* negative class. If we require $\sum_j \lambda_j = 1$, using inverse-frequencies will maintain the balance between the positive and negative class, as one-*vs.*-all is likely to introduce class imbalance. The second level of reweighting is within the negative class: we upweight the contribution of a hard sub-class by assigning a larger λ, making LGL-IGR focus more on the learning for that class.

Choice of λs When there are a large number of classes, treating all λs as hyperparameters and selecting the optimal values are not feasible in practice as we generally don't have the prior knowledge about which classes are hard. Instead, we adopt a strategy that assigns the weights during the training process. For each non-target class $j (j \neq k)$, let S_j^{MB} be the subset of S_j in the mini-batch, we use the mean predicted probability

$$
\bar{p}_j = \frac{1}{|S_j^{\text{MB}}|} \sum_{\boldsymbol{x} \in S_j^{\text{MB}}} p(y=1|\boldsymbol{x},\boldsymbol{\theta})
$$

as the class-level hardness measurement. A larger $\bar{p}_j$ implies class j is harder to separate

from the target class k. We then transform those $\bar{p}_j$'s using softmax to get λ_j:

$$\lambda_j = \frac{\exp(\beta \bar{p}_j)}{\sum_{i \neq k} \exp(\beta \bar{p}_i)},$$

where $\beta \geq 0$ is the temperature that can smooth ($0 \leq \beta \leq 1$) or sharpen ($\beta > 1$) each non-target class's contribution [16]. LGL-INR adaptively shifts its learning focus to those hard classes, meanwhile keep attentive on those easy classes. Note that this strategy only introduces one extra parameter in LGL-INR.

With the competition mechanism imposed by $\sum \lambda_j = 1$, LGL-INR can be viewed as a smoothed learning objective between the one-*vs.*-one and one-*vs.*-all approach: when $\beta = 0$, $\lambda_j = 1/K - 1$, all non-target classes are weighted equally, which is the in-negative-class balanced LGL using inverse-class frequencies; when β is very large, λ_j concentrates on the hardest class (e.g., $\lambda_j \approx 1$) and LGL-INR approximately performs one-*vs.*-one classification. We don't specifically fine-tune the optimal value of β and $\beta = 1$ works well in our experiments.

5.5 Experiments

We evaluate LGL-INR on several benchmark datasets for image classification. Note that in our experiments, applying LGL in multi-class classification naturally introduces data imbalance which is handled in our LGL-INR formulation. Our primary goal here is to demonstrate that LGL-INR can be used as a drop-in replacement for LGL and SML with competitive or even better performance, rather than outperform the existing best models using extra training techniques. For fair comparison, all loss functions are evaluated in the same test setting. Code will be made publicly available after the reviewing process.

Model	Architecture
CNN2C	CV(C20K5S1)-MP(K2S2)- CV(C50K5S1)-MP(K2S2)-800-10
CNN5C	CV(C32K3S1)-BN-CV(C64K3S1)-BN- CV(C128K3S1)-MP(K2S2)-CV(C256K3S1)- BN-CV(C512K3S1)-MP(K8S1)-512-10

Table 19: CNN architectures used for MNIST-type datasets. C-channel represents number, K-kernel size, S-stride, BN-batch normalization and MP-max pooling

Model	Loss	Dataset		
		MNIST	FMNIST	KMNIST
CNN2C	LGL	99.15	89.44	94.37
	SML	99.09	**91.15**	95.13
	LGL-INR	**99.29**	**91.15**	**96.43**
CNN5C	LGL	99.36	92.35	96.35
	SML	99.47	93.15	96.39
	LGL-INR	**99.63**	**93.54**	**97.46**

Table 20: Predictive top-1 accuracy rate (%) on the standard testing data of MNIST-type datasets.

5.5.1 Experiment Setup

Dataset We perform experiments on four MNIST-type datasets, MNIST, Fashion-MNIST (FMNIST) [158], Kuzushiji-MNIST (KMNIST) [17] and CIFAR10. FMNIST and KMNIST are intended as drop-in replacements for MNIST which are harder than MNIST. Both datasets are gray-scale images consisting of 10 classes of clothing and Japanese character respectively. CIFAR10 consists of colored images of size 32×32 from 10 objects.

Model setup We test three loss functions on each dataset with different CNN architectures. For MNIST-type datasets, two CNNs with simple configurations are used. The first one (CNN2C) has two convolution layers and the other one (CNN5C) has 5 convolution layers with batch normalization [58]. For CIFAR10, we use MobilenetV2 [52] and Resnet-18 [48] with publicly available implementations.

Implementation details All models are trained with the standard stochastic gradient descent (SGD) algorithm. The training setups are as follows. For MNIST-type data, the learning rate is set to 0.01, the momentum is 0.5, batch size 64, number of epoch is 20. We don't perform any data augmentation. For CIFAR data, we train the models with 100 epochs and set batch size to 64. The initial learning rate is set to 0.1, and divide it by 10 at 50-th and 75-th epoch. The weight decay is 10^{-4} and the momentum in SGD is 0.9. Data augmentation includes random crop and horizontal flip. We train all models without pretraining on large-scale image data. Model performance is evaluated by the top-1 accuracy rate and we report this metric on the testing data from the standard train/test split of those datasets for fair performance evaluation. For LGL-INR, we report the results using $\beta = 1$.

5.5.2 Predictive Results

Table 20 and Table 21 shows the classification accuracy using LGL, SML and LGL-INR on the MNIST-type and CIFAR10 dataset respectively. From the table, we can observe that for all three loss functions, model with larger capacity yields higher accuracy. On MNIST-type data, LGL yields overall poorer performance than SML. This is because in those datasets, some classes are very similar to each other (like shirt *vs.* coat in FMNIST) and the negative class consists of 9 different sub-classes. Hence the learning focus of LGL may get distracted from the hard sub-classes due to the averaging behavior of LGL as shown in Eq (5.9). However, SML doesn't suffer this problem as all negative sub-classes are treated equally. On CIFAR10, LGL achieves better accuracy than SML. This is possibly due to the lack of very similar classes as in MNIST-type data. This observation demonstrates LGL's potential as a competitive alternative to SML in some classification tasks.

Loss	Model	
	MobilenetV2	Resnet18
LGL	92.40	91.55
SML	91.11	91.32
LGL-INR	**93.34**	**93.68**

Table 21: Predictive top-1 accuracy rate (%) on the standard testing data of CIFAR10 using different models.

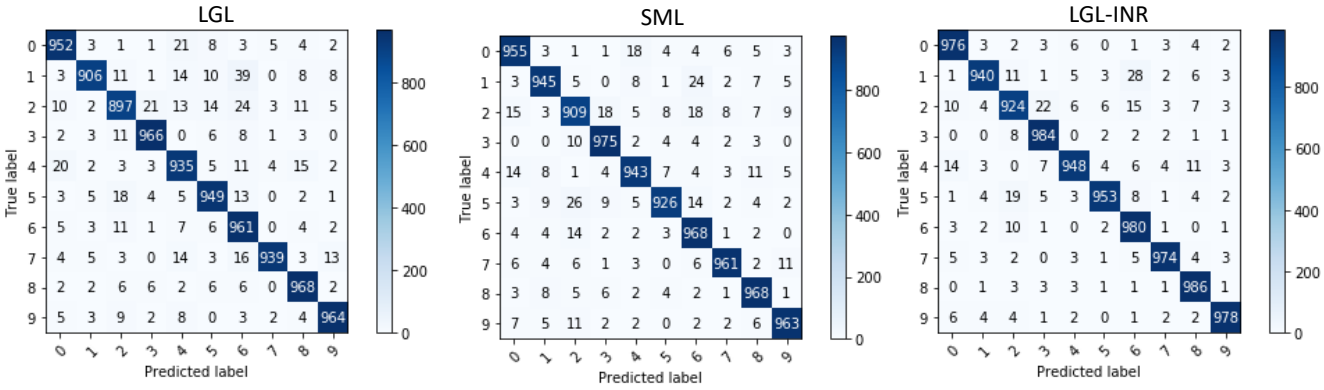

Figure 16: Confusion matrix on KMNIST testing data for LGL, SML and LGL-INR. Model: CNN2C. See Table 20 for overall accuracy. Notably, LGL-INR outperforms LGL in all 10 classes and SML in 9 classes except Class 1 (LGL-INR 940 *vs.* SML 945), in terms of per-class accuracy.

On the other hand, LGL-INR adaptively pays more attention on the hard classes while keeps its separation from easy classes. This enables LGL-IRN to outperform LGL and SML notably. Comparing LGL-IRN with LGL, we see that the multi-modality neglect problem deteriorates LGL's ability of learning discriminative features representation, which can be relieved by the in-negative class reweighing mechanism; comparing LGL-IRN with SML, focusing on learning hard classes (not restricted to classes similar to the target class) is beneficial. Also, the adaptive weight assignment in the training process doesn't require extra effort on the weight selection, making our method widely applicable.

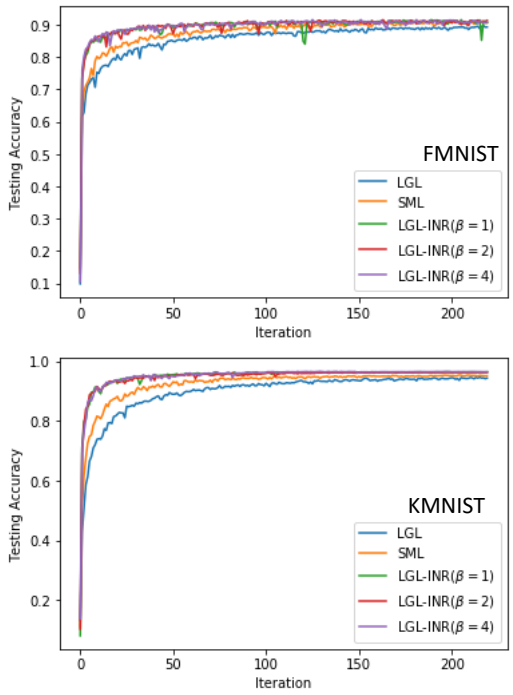

Figure 17: Testing top-1 accuracy on FMNIST and MNIST.

5.5.3 Further Analysis

We check the predictive behavior of LGL-INR in detail by looking at the confusion matrix on testing data. Here, we use CNN2C and KMNIST dataset as an example. Fig 16 show the results. We observe that for LGL, Class 1 and 2 have the lowest accuracy among 10 classes. By shifting LGL's learning focus on hard classes, LGL-INR significantly improves model performance on class 1 and 2. This is within our expectation backed by the theoretical depiction of LGL's learning behavior. SML does not have the multi-modality neglect problem as each class is treated equally in the learning process, yet it also does not pay more attention to the hard classes. This makes LGL-INR advantageous: LGL-INR outperforms SML on 9 classes out of 10. For example, class 0 have 18 samples misclassified into class 4 whereas only 6 are misclassified in LGL-INR.

β	1	2	4
Accuracy	96.43	96.29	96.43

Table 22: Accuracy of different β values on KMNIST. Model: CNN2C.

Figure 17 displays the training accuracy curve for LGL, SML and LGL-INR on FMNIST and KMNIST. Under the same training protocol, LGL-INR achieves slightly faster convergence rate than SML and LGL with comparative (FMNIST) or better (KMNIST) performance, implying that focusing on learning hard classes may facilitate model training process.

We also check the sensitivity of the temperature parameter β in LGL-INR weighting mechanism. Mathematically, a large or small value for β is not desirable as the LGL-INR is reduced to an approximate one-*vs.*-one or a class-balanced learning objective. We test $\beta = 1, 2, 4$ on KMNIST. As shown in Table 22 and Fig. 17, model performance is not sensitive to β in this range, making LGL-INR a competitive alternative to LGL or SML without introducing much hyper-parameter tuning.

5.6 Conclusion

In this chapter, motivated to explain the class-wise reweighting mechanism in LGL and SML, we theoretically deprived a system of probability equations that depicts the learning behavior of LGL and SML, as well as explains the roles of those class-wise weights in the loss function. By examining the difference in the effects of the weight mechanism on LGL and SML, we identify the multi-modality neglect problem is the major obstacle that can negatively affect LGL's performance in multi-class classification. We remedy this shortcoming of LGL with a in-negative-class reweighting mechanism. The proposed method shows its effectiveness on several benchmark image datasets. For future works, we plan to incor-

porate the estimation of data distribution in the model training process to further improve the efficacy of the reweighting mechanism.

CHAPTER 6 LEARNING COMPACT FEATURES VIA IN-TRAINING REPRESENTATION ALIGNMENT

6.1 Introduction

Recently, deep neural networks (DNNs) have achieved remarkable performance improvements in a wide range of challenging tasks in computer vision [68, 47, 55, 75], natural language processing [140, 16] and healthcare informatics [99, 84, 74, 82]. For supervised learning, DNNs can be viewed as a feature extractor followed by a linear classifier on the latent feature space, which are jointly trained using stochastic gradient descent (SGD). Specifically, in each iteration of SGD, a mini-batch of m samples $\{(x_i, y_i)\}_{i=1}^{m}$ is sampled from the training data $\{(x_i, y_i)\}_{i=1}^{n} (n > m)$. The gradient of loss function $L(x, \theta)$ is calculated on the mini-batch, and network parameter θ is updated via one step of gradient descent (learning rate α):

$$\begin{aligned} &\frac{1}{n}\sum_{i=1}^{n}\nabla_\theta L(x_i,\theta) \approx \frac{1}{m}\sum_{i=1}^{m}\nabla_\theta L(x_i,\theta), \\ &\theta \leftarrow \theta - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}\nabla_\theta L(x_i,\theta). \end{aligned} \tag{6.1}$$

This update in Eq.(6.1) can be interpreted from two perspectives. (1) From the conventional approximation perspective, the true gradient of the loss function (i.e., gradient on the entire training data) is approximated by the mini-batch gradient. As each mini-batch contains useful information for the learning tasks and its gradient computation is in expensive, large DNNs can be efficiently and effectively trained with modern computing infrastructures. (2) **Eq. (6.1) can also be interpreted as an exact gradient descent update on the mini-batch**. In other words, SGD updates network parameters θ to achieve

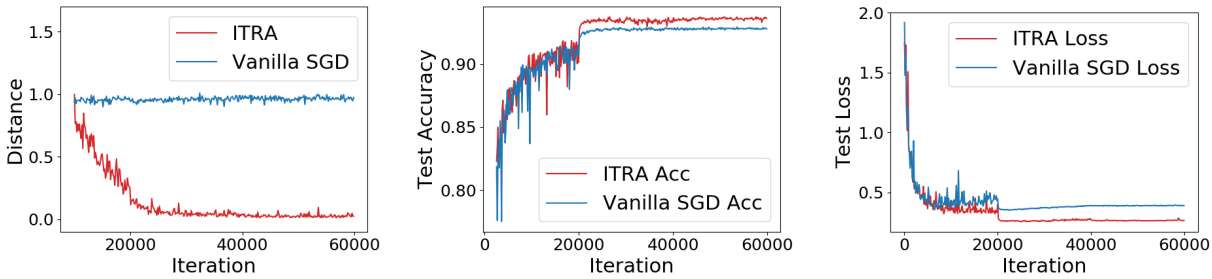

Figure 18: A comparison of ITRA and vanilla SGD training on the CIFAR10 testing data. Left: normalized distance between samples of the same class from different mini-batches used in training; middle: testing accuracy; right: testing cross-entropy loss. The model is Resnet18.

maximum improvement in fitting the mini-batch. As each mini-batch is usually uniformly sampled from training data, such exact update inevitably introduces the undesirable mini-batch-dependent noise and bias in the backpropagation, resulting in the over-adaption of model parameters to that mini-batch.

A natural question then to ask is, *"can we reduce the over-adaption to mini-batches?*", to reduce the mini-batch dependence on SGD update in Eq. (6.1). In this chapter, we propose In-Training Representation Alignment (ITRA) that aims at reducing the mini-batch over-adaption by aligning feature representation of different mini-batches that is learned by the feature extractor in SGD. Our motivation for feature alignment is: *if the SGD update using one mini-batch A is helpful for DNNs learning good feature representations with respect to the entire data, then for another mini-batch B, their feature representation should align well with each other*. In this way, we can reduce mini-batch over-adaption by forcing accommodation of SGD update to B and reducing dependence of the parameter update on A. Ideally, if the distribution $P(h)$ of latent feature h is known as a prior, we could explicitly match the mini-batch feature h_{mb} with $P(h)$ via maximum likelihood. However, in practice, $P(h)$ is not

known or does not even have an analytic form. To achieve this, we utilize the maximum mean discrepancy (MMD) [41] from statistical hypothesis testing for the two-sample problem. MMD is differentiable that can be trained via back propagation. Moreover, we show in an analysis that the gradient of MMD enjoys several good theoretical merits. Based on the theoretical analysis, ITRA reduces SGD update adaption to mini-batches by implicitly strengthening the supervision signal of high-density samples via an adaptively weighting mechanism (details are provided in later sections), where high-density samples are closely clustered to form modalities for each class.

To check effect of gradient update on feature representation learning, an illustrative example on CIFAR10 dataset is presented in Figure 18. The model is Resnet18 with BN layers trained with cross-entropy (CE) loss. We calculate the distance between a pair of same-class samples from two mini-batches respectively and plot the normalized distance (due to different magnitude of latent features trained with different methods) in the left panel of Figure 18, after model training is stabilized and achieves relatively good performance. We see that when model is trained only with CE loss in vanilla SGD, the distance stabilizes while the training makes progresses. This is due to that as long as the model captures the classification pattern for each class, vanilla SGD adapts to mini-batch samples to achieve gain for the loss function yet does not further encourage feature alignment to learn compact feature representations. From optimization perspective, we can also understand this with the gradient of CE loss: when a training sample is well learned by SGD with confident prediction, its contribution in the gradient vanishes, implying that SGD updates mostly focus on those samples with less confidence (i.e., smaller predicted probability). Hence, vanilla SGD has little effect on the compactness of feature representations.

However, in ITRA, the distance between a pair of samples keeps decreasing. This implies ITRA can help DNN learn more compact feature representations by aligning different mini-batches. DNNs can benefit from such compactness and hence achieve higher accuracy and lower loss (Fig. 18 middle and right panels).

We summarize our contributions as follows. (1) We propose a novel and general training strategy ITRA for training DNNs. ITRA augments conventional SGD with regularization by forcing feature alignment of different mini-batches to reduce mini-batch over-adaption. ITRA can be combined with existing regularization approaches and applied on a broad range of network architectures and loss functions. (2) We provide a theoretical analysis on the desirable effects of ITRA and explains why ITRA help reduce the over-adaption of vanilla SGD to mini-batches. With MMD, ITRA has an adaptively weighting mechanism that can help neural networks learn more discriminative feature representations and avoid the assumption of uni-modality on data distribution. Results on benchmark datasets demonstrate that training with ITRA can significantly improve DNN performances, compared with other state-of-the-art methods.

6.2 Related Work

Modern architectures of DNNs usually have an extremely large number of model parameters, which often outnumbers the available training data. To reduce overfitting in training DNNs, regularizations are needed. Those regularization methods include classic ones such as L_1/L_2-norm penalties and early stopping [44, 37]. For deep learning, many new approaches are proposed motivated by the SGD training dynamics. For example, dropout [131] and its variants [33, 36] achieve regularization by reducing the co-adaption of hidden neurons of DNNs. In the training process, dropout randomly sets some hidden

neurons' activation to zero, resulting in an averaging effect of a number of sub-networks. [58] proposes batch normalization (BN) to reduce the internal covariate shift caused by SGD. By maintaining the mean and variance of mini-batches, BN regularizes DNNs by discouraging the adaption to the mini-batches. For image classification, data-augmentation types of regularization are developed [23, 34]. Different from those approaches, our proposed ITRA is motivated by the perspective of exact gradient update for each mini-batch in SDG training. ITRA achieves regularization by encouraging the alignment of feature representations of different mini-batches. Those methods are compatible with ITRA for training DNNs and hence can be applied in conjunction with ITRA.

Another line of regularization are loss function-based that the supervision loss is augmented with other loss functions under different considerations. For example, label smoothing [142] corrupts the true label with a uniformly-distributed noise to discourage DNNs' over-confident predictions for training data. [149, 153] propose a strategy that assumes the latent feature representation to follow a known parametric distribution. With explicitly assuming the uni-modality of data distributions for each class, the supervision loss (i.e., cross-entropy) and maximum likelihood are simultaneously optimized. However, this assumption may be too strict as the true data distribution is generally unknown. ITRA avoids the distribution assumption and applies non-parametric MMD for feature alignment.

Recently, regularization methods based on knowledge distillation (KD) [51, 39] have attracted much attention. The idea behind this approach is that the distribution of model predicted probabilities for each class, rather than the probability for sample's true class, contains richer information about the data manifold. For example, [64] proposes the network as regularization approach that jointly trains a target and auxiliary network where

the target network is boosted with the knowledge distilled from the auxiliary network. [32] develops born-again neural networks (BAN). In BAN, multiple DNNs of the same architecture are sequentially trained using soft labels generated from its previous generation. [162] trains two DNNs with mutual knowledge distillation by optimizing the KL-divergence of model predicted probabilities. Though KD-based approaches are effective, multiple DNNs are trained which is computationally inefficient, whereas our ITRA is trained only in one network.

To match the distribution of different mini-batches, ITRA uses MMD as its learning objective. MMD [40, 41] is a probability metric for testing whether two finite sets of samples are generated from the same distribution. Using a universal kernel (i.e., Gaussian kernel), minimizing MMD encourages to match all moments of the empirical data distribution. MMD has been widely applied in many machine learning tasks. For example, [85] and [71] use MMD to train unsupervised generative models by matching the generated distribution with the data distribution. Another application of MMD is for the domain adaption. To learn domain-invariant feature representations, [91] uses MMD to explicitly match feature representations from different domains. There are also other probability-based distance metrics applied in domain adaption such as $\mathcal{A}$-divergence [4] and Wasserstein distance [127]. However, these metrics are non-differentiable while the differentiability of MMD enables the adaptive weighting mechanism in ITRA. Moreover, our goal is different from those applications. In ITRA, we do not seek exact distribution matching. Instead, we use MMD as a regularization to improve SGD training.

6.3 Preliminary: Maximum Mean Discrepancy

Given two finite sets of samples $S_1 = \{x_i\}_{i=1}^n$ and $S_2 = \{y_i\}_{i=1}^m$, MMD [40, 41] is constructed to test whether S_1 and S_2 are generated from the same distribution. MMD compares the sample statistics between S_1 and S_2, and if the discrepancy is small, S_1 and S_2 are then likely to follow the same distribution.

Using the kernel trick, the empirical estimate of MMD [40] w.r.t. S_1 and S_2 can be rewritten as:

$$\begin{aligned}\text{MMD}(S_1, S_2) = \big[&\frac{1}{n^2}\sum_{i,j=1}^{n}\mathcal{K}(x_i, x_j) + \frac{1}{m^2}\sum_{i,j=1}^{m}\mathcal{K}(y_i, y_j) \\ &- \frac{2}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathcal{K}(x_i, y_j)\big]^{1/2},\end{aligned}$$

where $\mathcal{K}(\cdot,\cdot)$ is a kernel function. [40] shows that if $\mathcal{K}$ is a characteristic kernel, then asymptotically MMD = 0 if and only S_1 and S_2 are generated from the same distribution. A typical choice of $\mathcal{K}$ is the Gaussian kernel with bandwidth parameter σ: $\mathcal{K}(x, y) = \exp(-\frac{||x-y||^2}{\sigma})$. With Gaussian kernel, minimizing MMD is equivalent to matching all orders of moments of the two datasets [85].

6.4 In-Training Representation Alignment

The Proposed ITRA The idea of ITRA is to reduce the DNN over-adaption to a mini-batch if we view the SGD iteration as an exact update for that mini-batch. In terms of feature learning, we attempt to train the feature extractor to encode less mini-batch dependence into the feature representation. For this purpose, we could ideally enforce the feature representation of the used mini-batch to align with the true data distribution. However, as the true distribution is unknown, we sample a different mini-batch as an approxi-

mation to the true distribution in each iteration of SGD.

More formally, let $f_\theta(x)$ be a convolutional neural network model for classification that is parameterized by θ. It consists of a feature extractor $h = E_{\theta_e}(x)$ and a linear classifier $C_{\theta_c}(h)$ parameterized by θ_e and θ_c respectively. Namely, $f_\theta(x) = C_{\theta_c}(E_{\theta_e}(x))$ and $\theta = \{\theta_e, \theta_c\}$. Without ambiguity, we drop θ in f, E and C for notational simplicity.

In each iteration of SGD, let $S_{(1)} = \{(x_i^{(1)}, y_i^{(1)})\}_{i=1}^{m_1}$ be the mini-batch of m_1 samples. Then the loss function using cross-entropy (CE) on $S_{(1)}$ can be written as

$$L_{mb}(\theta) = -\frac{1}{m_1}\sum_{i=1}^{m_1} \log f_{y_i^{(1)}}(x_i^{(1)}), \tag{6.2}$$

where $f_{y_i^{(1)}}(x_i^{(1)})$ is the predicted probability for $x_i^{(1)}$'s true label $y_i^{(1)}$. SGD performs one gradient descent step on L_{mb} w.r.t. θ using Eq. (6.1). To reduce θ's dependence on S_1 in this exact gradient descent update, we sample from the training data another mini-batch $S_{(2)} = \{(x_i^{(2)}, y_i^{(2)})\}_{i=1}^{m_2}$ to match the latent feature distribution between $S_{(1)}$ and $S_{(2)}$ using MMD:

$$\begin{aligned} H_{(1)} &= \{h_i^{(1)} = E(x_i^{(1)}) : i = 1, \cdots, m_1\}, \\ H_{(2)} &= \{h_i^{(2)} = E(x_i^{(2)}) : i = 1, \cdots, m_2\}, \\ \text{Match}(\theta_e; H_{(1)}, H_{(2)}) &= \text{MMD}(H_{(1)}, H_{(2)}). \end{aligned} \tag{6.3}$$

Our proposed ITRA modifies the conventional gradient descent step in SGD by augmenting the cross-entropy loss (Eq. (6.2)) with the matching loss, which justifies the name of ITRA:

$$\theta \leftarrow \theta - \alpha\nabla_\theta\big[L_{mb}(\theta) + \lambda\text{Match}(\theta_e; H_{(1)}, H_{(2)})\big], \tag{6.4}$$

Table 23: Classification accuracy (in %) and CE loss trained with and without ITRA, on the testing data of QMNIST, KMNIST and FMNIST.

		QMNIST	KMNIST	FMNIST
Acc	w/ ITRA	**98.94**	**94.42**	**90.79**
	w/o ITRA	98.86	94.19	90.52
CE	w/ ITRA	**0.037**	**0.196**	**0.257**
	w/o ITRA	**0.037**	0.227	0.269

where λ is the tuning parameter controlling the contribution of the matching loss. Note that mini-batch $S_{(2)}$ is not used in the calculation of cross-entropy loss $L_{mb}(\theta)$.

Initial results using ITRA To test the effectiveness of ITRA, we performed initial experiments using three MNIST-type datasets: QMNIST [159], KMNIST [17] and FMNIST [158]. We trained a simple CNN of two convolutional layers with and without ITRA under the exactly same setting. Experiment details are provided in the supplemental material. Table 23 shows the classification accuracy and CE loss (i.e., negative log-likelihood) on the standard testing data. From Table 23, we see that ITRA has overall better accuracies and smaller testing losses than the vanilla SGD training for three datasets, which implies a regularization effect of ITRA that can improve model performance.

Class-conditional ITRA For classification tasks, we could utilize the label information and further refine the match loss as a sum of class-conditional matching loss, termed as **ITRA-c** $(k = 1, \cdots, K)$:

$$
\begin{aligned}
H_{(1)}^k &= \{h_i^{(1)} = E(x_i^{(1)}) : y_i = k, i = 1, \cdots, m_1\} \\
H_{(2)}^k &= \{h_i^{(2)} = E(x_i^{(2)}) : y_i = k, i = 1, \cdots, m_2\} \\
\text{Match}_c(\theta_e; H_{(1)}, H_{(2)}) &= \frac{1}{K} \sum_{k=1}^{K} \text{MMD}(H_{(1)}^k, H_{(2)}^k),
\end{aligned}
\tag{6.5}
$$

where K is the total number of classes and $y_i = k$ the true label of sample x_i. The ITRA-c update is

$$\theta \leftarrow \theta - \alpha \nabla_\theta \big[L_{mb}(\theta) + \lambda \text{Match}_c(\theta_e; H_{(1)}, H_{(2)}) \big]. \tag{6.6}$$

6.4.1 Analysis on ITRA

On learning compact feature representations To further gain insight on the desirable effects of ITRA on the SGD training procedure, we analyze the matching loss at the sample level. With the same notation in Eq. (6.5), the matching loss for class k is

$$M := \text{Match}_k = \text{MMD}(H^k_{(1)}, H^k_{(2)}).$$

Since MMD is symmetric with respect to $H^k_{(1)}$ and $H^k_{(2)}$, without loss of generality, we consider sample $x^{(1)}_i$ with its feature representation $h^{(1)}_i = E(x^{(1)}_i)$ from $H^k_{(1)}$ (but the CE loss is not symmetric and only calculated on the first mini-batch $H_{(1)}$). Then the gradient of matching loss with respect to $h^{(1)}_i$ is (we drop the superscript (1) in $x^{(1)}_i$ and $h^{(1)}_i$ for simplicity)

$$\begin{aligned} \nabla_{h_i} M = & \frac{1}{\sqrt{M}} \nabla_{h_i} \big[\frac{1}{m_1^2} \sum_{j=1}^{m_1} \mathcal{K}(h_i, h^{(1)}_j) \\ & - \frac{2}{m_1 m_2} \sum_{j=1}^{m_2} \mathcal{K}(h_i, h^{(2)}_j) \big]. \end{aligned}$$

For Gaussian kernel $\mathcal{K}(x, y)$, its gradient with respect to x is $\nabla_x \mathcal{K}(x, y) = -2 \exp(-\frac{||x-y||^2}{\sigma}) \frac{x-y}{\sigma}$. Note that σ is data-dependent and treated as hyperparameter. Hence, it is not back propagated in the training process and in practice set as the median of sample pairwise distances

[41, 91, 85]. By the linearity of gradient operator, we have

$$\begin{aligned}\nabla_{h_i} M = &- \frac{2}{\sqrt{M}}\Big[\frac{1}{m_1^2}\sum_{j=1}^{m_1}\exp(-\frac{||h_i - h_j^{(1)}||^2}{\sigma})\frac{h_i - h_j^{(1)}}{\sigma} \\ &- \frac{2}{m_1 m_2}\sum_{j=1}^{m_2}\exp(-\frac{||h_i - h_j^{(2)}||^2}{\sigma})\frac{h_i - h_j^{(2)}}{\sigma}\Big].\end{aligned} \tag{6.7}$$

We notice that for function $g_a(x) = \exp(-x^2/a)x/a$ (a is some constant), $g_a(x) \to 0$ exponentially as $x \to \infty$. Hence, for fixed σ, using the triangle inequality of L_2 norm,

$$\begin{aligned}||\nabla_{h_i} M|| \leq &\frac{2}{\sqrt{M}}\Big[\frac{1}{m_1^2}\sum_{j=1}^{m_1} g_\sigma(||h_i - h_j^{(1)}||) \\ &+ \frac{2}{m_1 m_2}\sum_{j=1}^{m_2} g_\sigma(||h_i - h_j^{(2)}||)\Big].\end{aligned} \tag{6.8}$$

Within the mini-batch, $\sqrt{M}$ remain as constant for all samples. From Eq. (6.8), we observe that when x_i deviates significantly away from the majority of samples of the same class, i.e., noisy samples or outliers, $||h_i - h_j^{(1)}||$ and $||h_i - h_j^{(2)}||$ are large, the magnitude of its gradient in matching loss diminishes. In other words, x_i will only provide signal from the supervision loss (e.g., CE loss) and its impact on matching loss is negligible. On the other hand, training ITRA with matching loss promotes the alignment of feature representations of samples that stay close in the latent feature space. From the data distribution perspective, samples deviating from the majority are likely of low-density or even outliers. Then such behavior of ITRA implies that it can help DNNs to better capture information from high density areas and reduce the distraction of “low density" samples in learning feature representations on the data manifold.

On reducing over-adaption to mini-batches The analysis above shows that low-density

samples only provide supervision signal in ITRA, we now analyze how ITRA reduces the over-adaption to mini-batches. It turns out that this effect is achieved by an adaptively weighted feature alignment mechanism, which *implicitly* boosts the supervision signal from high-density samples and resultantly downweights relatively the contribution of low-density samples.

To understand this, we examine the full gradient of supervision loss L and matching loss MMD. Note that in ITRA, the gradient of supervision loss is only calculated on one mini-batch. Without loss of generality, we consider sample x_i from the first mini-batch. The full gradient of $L(x_i)$ and M=MMD$(x_i, H^k_{(2)})$ with respect to h_i is (using the same notation as above)

$$\begin{aligned}\nabla_{h_i}(M+L) =& \frac{4}{\sqrt{M}m_2}\sum_{j=1}^{m_2}\exp(-\frac{||h_i-h_j^{(2)}||^2}{\sigma})\frac{h_i-h_j^{(2)}}{\sigma} \\ &+\nabla_{o_i}L\cdot\frac{\partial o_i}{\partial h_i},\end{aligned}$$

where o_i is the output for x_i. Let $A=\sum_{j=1}^{m_2}\exp(-||h_i-h_j^{(2)}||^2/\sigma)$ and $w_j=\exp(-||h_i-h_j^{(2)}||^2/\sigma)/A$ ($\sum_{j=1}^{m_2}w_j=1$), then equivalently:

$$\nabla_{h_i}(M+L)=\frac{4A}{\sqrt{M}m_2\sigma}(h_i-\sum_{j=1}^{m_2}w_jh_j^{(2)})+\nabla_{o_i}L\cdot\frac{\partial o_i}{\partial h_i}. \tag{6.9}$$

When ITRA converges and DNNs is well trained with good performance, $||\nabla_{h_i}(M+L)||\approx 0$ and $||\nabla_{o_i}L\cdot\partial o_i/\partial h_i||$ is close to zero, we have $||h_i-\sum_{j=1}^{m_2}w_jh_j^{(2)}||<\epsilon$ (ϵ is a small scalar). In other words, ITRA promotes the feature representation h_i of x_i to align with the weighted average $\sum_{j=1}^{m_2}w_jh_j^{(2)}(\sum_{j=1}^{m_2}w_j=1)$, where each w_j is adaptively adjusted in the training

process based on similarity between h_i and $h_j^{(2)}$ in the latent feature space. As mini-batch samples are uniformly sampled from the training data, it is expected that on average, the majority of $\{h_j^{(2)}\}_{j=1}^{m_2}$ are from high-density area of the data distribution. For DNNs with good generalizability, DNNs must perform well for samples from those areas (as testing samples are more likely to be generated from high-density areas in the data manifold). Hence, provided that sample x_i is of high-density that already provides useful supervision signal, ITRA further boosts its contribution by aligning h_i with $\sum_{j=1}^{m_2} w_j h_j^{(2)}$ of other high-density samples in the 2nd mini-batch. *The adaptive weight w_j is critical*: if sample $h_j^{(2)}$ is of low-density and deviates far from x_i, its weight w_j is automatically adjusted small, having vanishing contribution in the gradient. This in turn downweights relatively the contribution of low-density samples in SGD, resulting in the reduction of over-adaption to mini-batches.

Accommodating multi-modalities The adaptively weighting mechanism brings another benefit: if the data distribution (for each class) is of multi-modality in the latent feature space, ITRA automatically aligns x_i with its corresponding modality. Specifically, without loss of generality, assume there are two modalities md_1 and md_2, $\{h_j^{(2)}\}$ consists of samples from md_1 and md_2 and x_i is generated from md_1. We can rewrite $h_i - \sum_{j=1}^{m_2} w_j h_j^{(2)} = h_i - (\sum_{j\in md_1} w_j h_j^{(2)} + \sum_{j\in md_2} w_j h_j^{(2)})$. As x_i is generated from md_1 and deviates from md_2, implying that x_i is closer to samples from the same modality than those from the other modality. Hence, with the adaptively weighting mechanism in Eq. (6.9), $w_j \approx 0$ $(j \in md_2)$ and $h_i - \sum_{j=1}^{m_2} w_j h_j^{(2)} \approx h_i - \sum_{j\in md_1} w_j h_j^{(2)}$. That is, align x_i only with samples from the same modality. Therefore, ITRA avoids the uni-modality assumption on data distribution as in [153, 149] and justifies the advantage of nonparametric MMD for feature alignment.

6.5 Experiments

In this section, we evaluate the ITRA strategy on benchmark datasets of image classification. In our experiments, ITRA-c is tested as it provides implicit label information with better supervision in the training process. We implement our codes in Pytorch [109] and utilize Nivdia RTX 2080TI GPU for computation acceleration.

Datasets We test ITRA on five benchmark datasets Kuzushiji-MNIST (KMNIST) [17], Fashion-MNIST (FMNIST) [158], CIFAR10, CIFAR100 [67] and STL10 [18]. KMNIST and FMNIST are two gray-scale image datasets that are intended as alternatives to MNIST. Both datasets consist of 70,000 (28×28) images from 10 different classes of Japanese character and clothing respectively, among which 60,000 are used for training data and the remaining 10,000 for testing data. CIFAR10 and CIFAR100 ar colored image datasets of 32×32 resolution. It consists of 50,000 training and 10,000 testing images from 10 and 100 classes respectively. STL10 is another colored image dataset where each image is of size 96×96. Original STL10 has 100,000 unlabeled images, 5,000 labeled for training and 8,000 labeled for testing. In our experiment, we only use the labeled subset for evaluation.

Comparison In the experiments, in addition to the vanilla SGD training as the baseline (i.e., w/o ITRA), we also compare ITRA with other loss-function based regularization methods. All DNNs in our experiments include BN layers as part of the model architectures. Label smoothing [142] (LSR) is a target-based regularization that the Dirac distribution for ground truth label is replaced with a mixture of Dirac distribution and uniform distribution; center loss [153] (Center) encourages interclass compactness of feature representations which augments the cross-entropy loss with the maximum likelihood [149]

assuming that each class follows Gaussian distribution in latent feature space.

Implementation Details[6] Through all experiments, the optimization algorithm is the standard stochastic gradient descent with momentum and the loss function is cross-entropy (CE) loss. In ITRA-c, CE loss is further combined with the matching loss Eq. (6.6) in each iteration.

For the initial experiments on QMNIST, KMNIST and FMNIST, the CNN architecture of 2 convolutional layers is Conv(C20K5S1)-MP(K2S2)-Conv(C50K5S1)-MP(K2S2)-100-10, where CxKySz for a convolutional layer means x convolutional kernels with kernel size y, stride z; MP represents max-pooling. To train the network, we use the SGD algorithm with a momentum of 0.5. The number of epochs is 50. Learning rate is 0.01 and multiplied by 0.2 every 20 epochs. Batch size is set to 150. We don't use L_2 regularization. In ITRA, the tuning parameter λ is set to 1.

For KMNIST and FMNIST in "Experiments" section, we build a 5-layer convolutional neural network (CNN) with batch normalization applied. The CNN architecture is Conv(C32K3S1) - BN - Conv(C64K3S1) - BN - Conv(C128K3S1) - MP(K2S2) - Conv(C256K3S1) - BN - Conv(C512K3S1) - MP(K8S1) - 512 - 10, where BN represents batch-normalization. Momentum is set to 0.5, batch size 150, number of epochs 50, initial learning rate 0.01 and multiplied by 0.2 at 20th and 40th epoch. No data augmentation is applied. For CIFAR10 and STL10, we use publicly available implementation of VGG13 [130], Resnet18 [47] and MobilenetV2 [53]. All models are trained with 150 epochs, SGD momentum is set to 0.5, initial learning rate is 0.5 and multiplied by 0.1 every 50 epochs, batch size 150. We resize STL10 to 32×32. For colored image datasets, we use random crop and horizontal flip for

[6] Code is provided at https://github.com/Dichoto/ITRA

Table 24: Accuracy (in %, larger is better) and CE (smaller is better) on KMNIST and FMNIST testing data.

	KMNIST			FMNIST		
	λ	Acc	CE	λ	Acc	CE
Baseline	-	95.57	0.183	-	92.43	0.294
LSR	0.1	95.60	0.181	0.1	92.47	0.292
Center	0.1	94.90	0.214	0.1	92.10	0.263
ITRA	0.8	**95.79**	**0.170**	0.6	**92.57**	**0.224**

data augmentation. For CIFAR100 with more classes, we use weight decay of 5×10^{-4} and a mini-batch size of 300.

In all experiments, networks are trained with each method under the same setting (learning rate, batch size et al.). For the bandwidth parameter in Gaussian kernels, we follow the practice in [40, 41, 91] that takes the heuristic of setting σ as the median squared distance σ_{Med} between two samples and use a mixture of Guassian kernels with bandwidth set as a multiple of σ_{Med}. In our experiments, we use 5 kernels $k_{\text{mix}}(x, y) = \frac{1}{5}\sum_{i=1}^{5} k_{\sigma_i}(x, y)$ with $\{\sigma_i = 2^i \sigma_{\text{Med}} : i = 0, \cdots, 4\}$. To select the tuning parameter λ in each method, we split out 5,000 images from the training data as validation dataset. For λ in ITRA, we test {0.2, 0.4, 0.6, 0.8, 1} for checking ITRA's sensitivity to it. Note that when $\lambda = 0$, ITRA is equivalent to vanilla SGD training. In LSR, we select the tuning parameter from $\{0.2, 0.15, 0.1, 0.05\}$; for center loss, $\{0.2, 0.15, 0.10, 0.05\}$ are tested. We utilize the standard train/test split given in the benchmark datasets and train all models once on the training data and performances are evaluated on the testing data.

6.5.1 Results

For evaluation, we report the Top-1 accuracy and CE loss values for all methods. The optimal hyperparameter λ for each method is also shown and results on other tunning

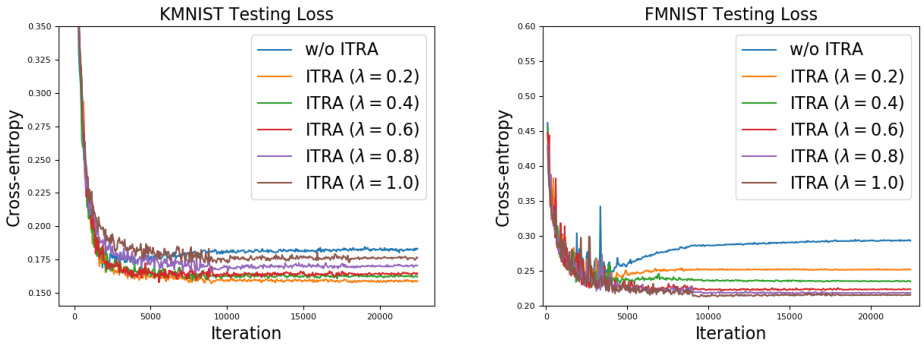

Figure 19: Testing loss w.r.t. different λ values on KMNIST and FMNIST

Table 25: Accuracy (in %, larger is better) and CE loss (smaller is better) of Resnet18, VGG13 and MobilenetV2 on CIFAR10, STL10 and CIFAR100.

		CIFAR10			**STL10**			**CIFAR100**		
		λ	Acc	CE	λ	Acc	CE	λ	Acc	CE
Resnet18	Baseline	-	92.99	0.40	-	70.88	1.63	-	74.19	1.05
	LSR	0.1	92.73	0.42	0.1	71.08	1.55	0.1	74.21	1.04
	Center	0.1	92.30	0.35	0.1	70.97	1.10	0.05	73.98	0.98
	ITRA	0.8	**93.70**	**0.27**	0.6	**72.78**	**1.05**	0.6	**74.88**	**0.97**
VGG13	Baseline	-	92.49	0.47	-	74.40	1.55	-	71.72	1.46
	LSR	0.1	92.53	0.46	0.1	74.50	1.51	0.1	71.75	1.43
	Center	0.05	92.11	0.38	0.05	74.04	1.16	0.05	71.65	1.31
	ITRA	0.8	**92.72**	**0.33**	0.8	**75.80**	**0.93**	0.6	**72.55**	**1.22**
MobilenetV2	Baseline	-	88.55	0.62	-	59.09	2.14	-	66.42	1.57
	LSR	0.1	88.77	0.61	0.1	59.01	2.12	0.1	66.60	1.55
	Center	0.1	88.81	0.53	0.1	58.24	**1.46**	0.05	66.39	1.51
	ITRA	1.0	**89.37**	**0.43**	0.6	**62.02**	1.60	0.6	**67.23**	**1.49**

parameter values are shown in supplemental material. .

KMNIST Table 24-KMNIST shows the predictive performance for KMNIST. From the Table, we see that training with ITRA achieves better results in terms of accuracy and CE. In terms of the testing loss, ITRA always has a smaller loss value compared with other methods. The testing loss with respect to different λ values are shown in Figure 19 (a) of the supplemental materials. As CE is equivalent to negative log-likelihood, smaller CE value implies the network makes predictions on testing data with higher confidence on average.

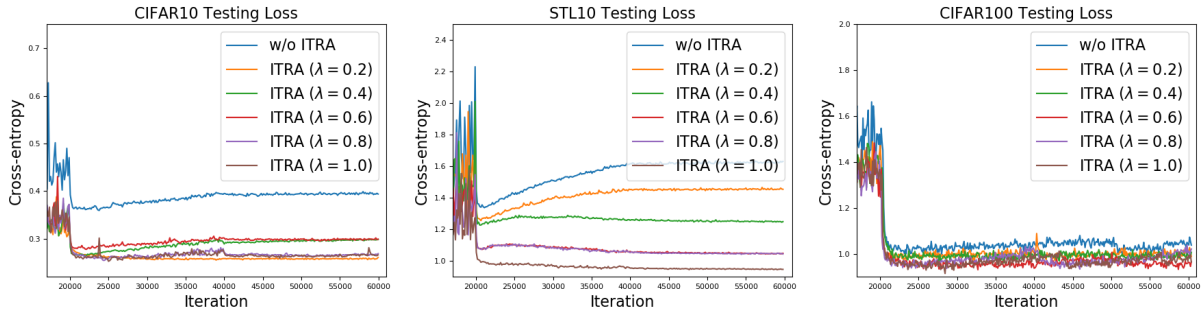

Figure 20: Testing loss w.r.t. different λ values on CIFAR10, STL10 and CIFAR100. CNN Model is Resnet18.

In each iteration of ITRA, there is a tradeoff between the CE and matching loss. This leads to that ITRA has a regularization effect by alleviating the over-confident predictions on training data. As a result, the smaller gap between training and testing loss implies ITRA has better generalization performance.

FMNIST Table 24-FMNIST shows the predictive performance for FMNIST testing data. As can be seen from the Table, ITRA has better predictive accuracy than other methods and smaller testing loss (the testing loss for different λs are shown in Figure 19 (b) in the supplemental materials). When trained with vanilla SGD, we observe that the increasing testing loss exhibits a trend of overfitting, which is due to that FMNIST has a significant number of hard samples (e.g., those from pullover, coat and shirt classes). However, ITRA is capable of regularizing the training process hence prevents overfitting and stabilizes the testing loss as shown in the figure.

CIFAR10 In Table 25-CIFAR10, we present the performance of Resnet18, VGG13 and MobilenetV2 on CIFAR10. From Table, we see that ITRA achieves the best performance compared among all four methods. When λ is set with a relatively large value of 0.8 or 1, ITRA can improve the accuracy by a margin 0.71% for Resnet, 0.23% for VGG13 and 0.82%

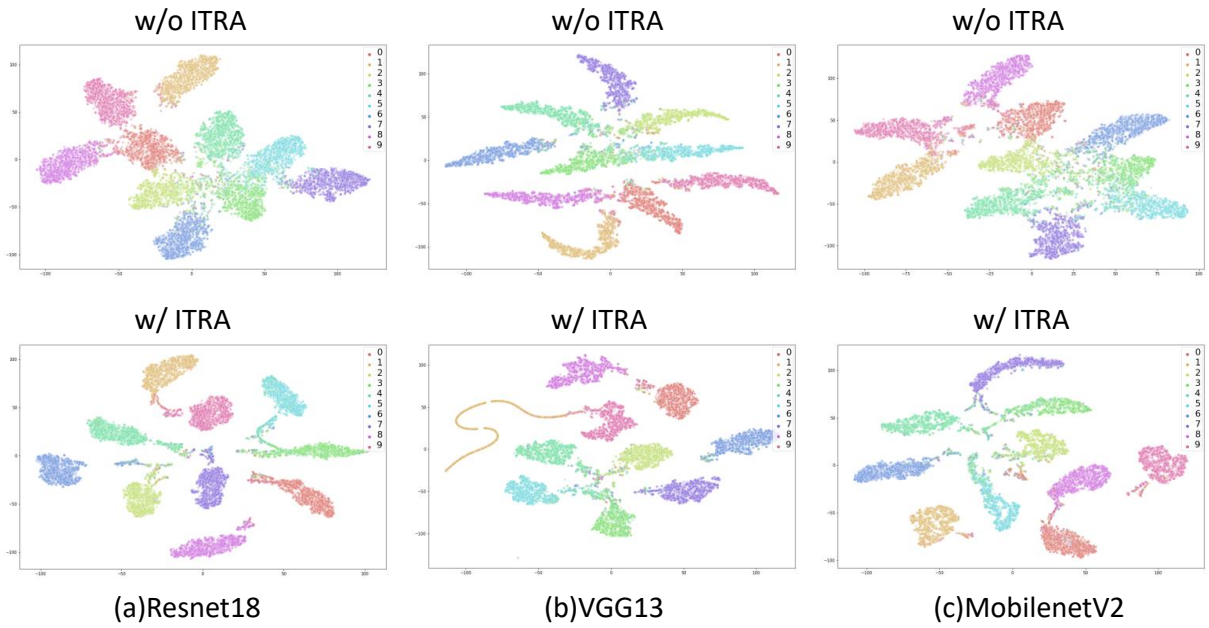

Figure 21: T-SNE plot for CIFAR10 testing data. Networks are trained with λ that achieves best accuracy in Table 25.

for MobilenetV2. This is due to that larger λs incorporate stronger implicit supervision information as mini-batches from the same class are matched. We also plot the CE loss for different λs in Figure 20 (a) w.r.t. Resnet (for illustration purpose). Comparing with baseline, we see that training with ITRA results in significant gain in CE, regardless of network architecture: Resent $32.6\%((0.40-0.27)/0.40)$, VGG 29.4% and MobilenetV2 30.6%. This pattern also holds for other λ values. A closer gap between training and testing losses usually implies better generalization as it means a closer distribution match between train and testing data. From this perspective, ITRA can regularize DNNs to learn feature representations with better generalizability.

STL10 The results on STL10 is shown in Table 25-STL10. Similar to CIFAR10, a larger value of λ results in higher accuracy with significant margin for ITRA, i.e., Resnet 1.9%, VGG13 1.4% and MobilenetV2 2.93%. In terms of CE loss, all methods have similar training

losses that are close to zero. However, ITRA and Center have significant better testing loss than other the baseline and LSR. From the Table, Resnet can improve testing loss $35.6\%(0.581/1.630)$, VGG 39.5% and MobilenetV2 20.6%. As shown in Figure 20 (b), when trained with the baseline, the testing loss shows an increasing trend as a sign of overfitting while ITRA can alleviate this trend as λ increases.

CIFAR100 Table 25-CIFAR100 shows the results for CIFAR100. Compared with other methods, ITRA achieves the best accuracy on the CIFAR100 testing data. In terms of CE loss, ITRA also results in the smallest loss values. As shown in Figure 20 (c), for other λ values, ITRA always improves CE over vanilla SGD. However, the improvements in CE loss is not as significant as other datasets like CIFAR10 and STL10, 7.6% for Resnet18, 15.6% VGG13 and 6.5% MobilenetV2. Since CIFAR100 has 100 classes with 500 training samples for each class, among which some classes are mutually similar, for example, otter *v.s.* beaver, this implies that samples from those classes may stay close in the feature space. Resultantly classifiers make less confident predictions for those samples and hence leads to larger CE values, which holds true for all methods. However, DNNs can still benefit from ITRA as ITRA can explicitly learn more compact feature representations, resulting in marked accuracy improvement on the testing data.

6.5.2 Learning Properties of ITRA

Implicit supervision We observe that ITRA with a relatively larger λs tends to have better performances when compared with smaller λs. and outperforms the vanilla SGD training (Smaller λs achieves at least comparable performances with vanilla SGD). A plausible reason for this phenomenon is that ITRA-c provides implicit supervision in the learning process by matching two random mini-batches from the *same* class. This leads to that

ITRA can reduce the testing CE loss significantly, in particular for CIFAR10 and STL10 datasets. Given a sample $(x, y = k)$, its CE loss is calculated as $-\log f_k(x)$, where f_k is the predicted probability for x's true class label k. A smaller testing CE loss implies a larger probability f_k. With larger λs, ITRA can benefit from the stronger implicit supervision and hence improve network performance.

Learning compact feature representations From the geometric perspective, samples from the same class should stay close (i.e., intra-class compactness) and those from different classes are expected to stay far apart (i.e., inter-class separability) in the feature space (so that f_k output by softmax is large). We visualize the distribution of CIFAR10 testing samples with T-SNE [93] in Figure 21. From the figure, we have the following observations: (1) ITRA learns feature representation that is much tighter with clearer inter-class margin than that learned by vanilla SGD training. (2) The data distribution in the latent space learned by ITRA exhibits a consistent pattern that for each class, the majority of testing samples are closely clustered to form a data manifold, while a small subset of samples deviate from the majority. This phenomenon concurs with our analysis that the matching loss provides diminishing gradient signals for “low-density" samples while encourages the closeness of “high-density" samples. Hence, ITRA can effectively capture the “typical pattern" of each class but can miss some hard samples that overlap with other classes. This explains why ITRA achieves impressive improvement in CE loss but not as much in accuracy. Overall, ITRA still outperforms vanilla SGD training and can be used as a promising training prototype that enjoys theoretical merits as shown in the analysis for matching loss.

6.6 Conclusion

In this chapter, we propose a new training strategy, ITRA, as a loss function based regularization approach that can be embedded in the SGD training procedure. ITRA augments vanilla SGD with a matching loss that uses MMD as the objective function. We show that ITRA enjoys three theoretical merits that can help DNN learn compact feature representations without assuming uni-modality on the feature distribution. Experimental results demonstrate its excellent performance on classification tasks, as well as its impressive feature learning capacity. There are two possible directions for our future studies. The first one is to improve ITRA that can learn hard sample more effectively. The second one is the ITRA application in learning from poisoned datasets as ITRA is able to capture the high density areas (i.e., modalities) for each class where poisoned samples deviates far from those areas (e.g., erroneously labeled samples from other classes).

CHAPTER 7 CONCLUSION

In this dissertation, we present several methods for predictive modeling from different perspectives. With specific considerations in dealing with different data forms, our research works are of significant intellectual merits.

7.1 Summary of Contribution

In Chapter 2 and 3, based on the generalized linear models, we propose two novel regularization methods for multi-class classification and finite mixture of regression respectively. Those models incorporate l_1-, l_2- and $l_{2,1}$-norms in their formulation that are able to select features and feature groups. For high-dimensional problems where the number of features outnumbers that of training samples, their sparsity-inducing effect can achieve excellent performances as well as significantly enhance model's interpretability. In terms of optimization, we develop efficient block coordinate descent algorithms to solve those two models.

As healthcare informatics is an important application of predictive modeling, in Chapter 4, we build two DNN models, ATAN and DMNN, for cardiovascular disease risk prediction. The ATAN model uses multi-task learning as a regularization to tackle the small data problem. DMNN builds a mixture of neural networks to enable the discoveries of patient subgroup that maintains good predictive performance as well as subgroup-dependent risk factor identification.

In Chapter 5 and 6, we develop two models that improve DNNs performance by learning better feature representations than the conventional SGD training using cross-entropy loss. In Chapter 5, we systematically analyze the learning property of LGL and SML and then propose the theoretically motivated new loss, LGL-INR, that achieves excellent results

on benchmark datasets. In Chapter 6, we take a new perspective on the conventional SGD training that inspires our ITRA using MMD. Through an in-depth analysis, we demonstrate that our method enjoys three theoretical merits that improve DNNs performance as well as help DNN learn compact feature representations.

7.2 Future Directions

The methods developed in this dissertation are demonstrated effective for predictive modeling. There are still possibilities for further improvements in our future study. We discuss the potential developments as follows.

In Chapter 2, we propose CCSOGL for multi-class classification that simultaneously selects class-dependent features and feature groups. While this approach is effective, it heavily relies on the availability of prior knowledge about the grouping information of features. However, when the structure of feature groups is unknown, our method is not applicable. To extend CCSOGL to those cases, a promising approach is to incorporate bi-clustering [104] in CCSOGL that is capable of clustering features. Each cluster can be viewed as a feature group and CCSOGL can utilize such information to achieve feature selection at the group level.

In Chapter 3, based on $l_{2,1}$-norm, a sparsity-inducing regularization method is developed for finite mixture of regression. For predictive modeling, regression and classification are two major tasks. Hence, a direct development of our method is to extend it to the finite mixture of classification where each component is modeled by logistic regression.

With the recent surge in the availability of electronic health records (EHR), learning form EHR has attracted much attention in the machine learning community. In Chapter 4, we develop two DNN models for cardiovascular disease prediction. While our methods

can achieve accurate predictive performance, it only utilizes a limited amount of patient information (e.g., demographics and lab results) and does not fully utilize the rich information in EHR data that consists of unlabeled data, medical images, clinical texts and time series data. Hence, for future studies, we can exploit the recent advances in representation learning using DNNs to learn effective patient representations from the multimodal EHR data that can incorporated in predictive modeling.

The learning property derived of LGL and SML in Chapter 5 reveals that the distribution of latent features plays an important role for the classification performance. If data exhibits multi-modality for each class, then the inference of feature distribution can enable us to design new algorithm to focus on classification of modalities that are close in the latent feature space. We expect that predictive modeling can benefit from such refinement with "locality" awareness. Hence, the incorporation of distribution inference into the model learning process is a promising research direction.

By viewing the SGD update as an exact update on the mini-batch, in Chapter 6, we propose ITRA that can regularize DNNs by forcing the alignment of feature representations of two mini-batches. The theoretical analysis demonstrates that ITRA achieves three desirable effects that can encourage DNNs to learn compact feature representations. In particular, we shows that ITRA can implicitly boost the contribution of "high-density" samples and relatively downweight that of "low-density" samples in the supervision loss to reduce the over-adaption to mini-batches. Directly motivated by this property, a future direction of our study is to learn from poisoned data based on ITRA. The idea is that when the input features of a training sample (i.e., x) are corrupted or the sample is erroneously labeled (i.e., y), we can view this (x, y) as a low-density sample, as it deviates from samples that

have the same class label. Hence, downweighting such samples in ITRA can benefit model training by reducing the distraction of poisoned samples.

APPENDIX

Conference Publications

C1. Deng Pan, Xiangrui Li, Xin Li and Dongxiao Zhu, "Explainable Recommendation via Interpretable Feature Mapping and Evaluating Explainability", *IJCAI 2020, pp. 2690-2693*.

C2. Xiangrui Li, Xin Li, Deng Pan, Dongxiao Zhu, “On the Learning Property of Logistic and Softmax Losses for Deep Neural Networks", *AAAI 2020, pp. 4739-4746*.

C3. Xiangrui Li, Jasmine Hect, Moriah Thomason, Dongxiao Zhu, “Interpreting Age Effects of Human Fetal Brain from Spontaneous fMRI using Deep 3D Convolutional Neural Networks", *ISBI 2020, pp. 1424-1427*.

C4. Xiangrui Li, Dongxiao Zhu, Phillip Levy, “Predicting Clinical Outcomes with Patient Stratification via Deep Mixture Neural Networks", *AMIA Summits 2020, pp. 367-376*.

C5. Xiangrui Li, Dongxiao Zhu, Phillip Levy, “Predictive Deep Network with Leveraging Clinical Measure as Auxiliary Task", *BIBM 2017, pp. 786-791*.

C6. Xiangrui Li, Dongxiao Zhu, Ming Dong, Milad Zafar Nezhad, Alexander Janke, and Phillip Levy, “SDT: A Tree Method for Detecting Patient Subgroups with Personalized Risk Factors", *AMIA Summits 2017, pp. 193-202*.

C7. Nezhad Milad Zafar, Dongxiao Zhu, Xiangrui Li, Kai Yang, and Phillip Levy. "SAFS: A deep feature selection approach for precision medicine", *BIBM 2016, pp. 501-506*.

Preprints

R1. Xin Li*, Xiangrui Li*, Deng Pan*, Dongxiao Zhu, "Improving Adversarial Robustness via Probabilistically Compact Loss with Logit Constraints".

R2. Xiangrui Li, Deng Pan, Xin Li and Dongxiao Zhu, "Learning Compact Features vis In-Training Representation Alignment".

Journal Publications

J1. Xiangrui Li, Dongxiao Zhu, Phillip Levy, “Leveraging auxiliary measures: a deep multi-task neural network for predictive modeling in clinical research", *BMC Medical Informatics and Decision Making* 18.4 (2018): 45-53.

J2. Xiangrui Li, Dongxiao Zhu, “Robust Feature Selection via Sparse $\ell_{2,1}$-Norm in Finite Mixture of Regression", *Pattern Recognition Letters* 108 (2018): 15-22.

J3. Xiangrui Li, Dongxiao Zhu, and Ming Dong, “Multinomial Classification with Class-conditional Overlapping Sparse Feature Groups", *Pattern Recognition Letters* 101 (2018): 37-43.

REFERENCES

[1] L. R. Acharya, T. Judeh, G. Wang, and D. Zhu. Optimal structural inference of signaling pathways from unordered and overlapping gene sets. *Bioinformatics*, 28(4):546–556, 2012.

[2] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[3] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[4] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[5] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[6] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[7] R. Caruana. Multitask learning. *Learning to learn*, pages 95–133, 1998.

[8] A. T. Chaganty and P. Liang. Spectral experts for estimating mixtures of linear regressions. In *ICML*, pages 1040–1048, 2013.

[9] Z. Che, D. Kale, W. Li, M. T. Bahadori, and Y. Liu. Deep computational phenotyping. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 507–516. ACM, 2015.

[10] Z. Che, S. Purushotham, R. Khemani, and Y. Liu. Interpretable deep models for icu outcome prediction. In *AMIA Annual Symposium Proceedings*, volume 2016, page 371. American Medical Informatics Association, 2016.

[11] B. Chen, W. Deng, and H. Shen. Virtual class enhanced discriminative embedding learning. In *Advances in Neural Information Processing Systems*, pages 1942–1952, 2018.

[12] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[13] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine Learning for Healthcare Conference*, pages 301–318, 2016.

[14] E. Choi, A. Schuetz, W. F. Stewart, and J. Sun. Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association*, 24(2):361–370, 2016.

[15] E. Choi, C. Xiao, W. Stewart, and J. Sun. Mime: Multilevel medical embedding of electronic health records for predictive healthcare. In *Advances in Neural Information Processing Systems*, pages 4547–4557, 2018.

[16] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585, 2015.

[17] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha. Deep learning for classical japanese literature, 2018.

[18] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[19] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[20] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. Class-balanced loss based on effective number of samples. *arXiv preprint arXiv:1901.05555*, 2019.

[21] C. Desmedt, F. Piette, S. Loi, Y. Wang, F. Lallemand, B. Haibe-Kains, G. Viale, M. Delorenzi, Y. Zhang, M. S. d'Assignies, et al. Strong time dependence of the 76-gene prognostic signature for node-negative breast cancer patients in the transbig multicenter independent validation series. *Clinical cancer research*, 13(11):3207–3214, 2007.

[22] R. B. Devereux, T. G. Pickering, G. A. Harshfield, H. D. Kleinert, L. Denby, L. Clark, D. Pregibon, M. Jason, B. Kleiner, J. S. Borer, et al. Left ventricular hypertrophy in patients with hypertension: importance of blood pressure response to regularly recurring stress. *Circulation*, 68(3):470–476, 1983.

[23] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[24] Q. Dong, X. Zhu, and S. Gong. Single-label multi-class image classification by deep logistic regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3486–3493, 2019.

[25] P. K. Dunstan, S. D. Foster, F. K. Hui, and D. I. Warton. Finite mixture of regression modeling for high-dimensional count and biomass data in ecology. *Journal of Agricultural, Biological, and Environmental Statistics*, 18(3):357–375, 2013.

[26] E. Dusseldorp and I. Van Mechelen. Qualitative interaction trees: a tool to identify qualitative treatment–subgroup interactions. *Statistics in medicine*, 33(2):219–237,

2014.

[27] D. Eddelbuettel. *Seamless R and C++ integration with Rcpp*. Springer, 2013.

[28] A. H. El-Gharbawy, V. S. Nadig, J. M. Kotchen, C. E. Grim, K. B. Sagar, M. Kaldunski, P. Hamet, Z. Pausova, D. Gaudet, F. Gossard, et al. Arterial pressure, left ventricular mass, and aldosterone in essential hypertension. *Hypertension*, 37(3):845–850, 2001.

[29] C. Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[30] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[31] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[32] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

[33] H. Gao, J. Pei, and H. Huang. Demystifying dropout. In *The 36th International Conference on Machine Learning (ICML 2019)*, 2019.

[34] X. Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017.

[35] T. D. Gedeon. Data mining of inputs: analysing magnitude and functional measures. *International Journal of Neural Systems*, 8(02):209–218, 1997.

[36] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10727–10737, 2018.

[37] I. Goodfellow, Y. Bengio, and A. Courville. Deep learning, 2016.

[38] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[39] J. Gou, B. Yu, S. J. Maybank, and D. Tao. Knowledge distillation: A survey. *arXiv preprint arXiv:2006.05525*, 2020.

[40] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2007.

[41] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[42] Y. Guo and W. Xue. Probabilistic multi-label classification with sparse feature learning. In *IJCAI*, 2013.

[43] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[44] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning: Data mining, inference, and prediction. 2nd edtion. *NY Springer*, 2009.

[45] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[46] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, 9:1263–1284, 2008.

[47] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages

770–778, 2016.

[48] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[49] A. Helvacı, B. Çopur, and M. Adaş. Correlation between left ventricular mass index and calcium metabolism in patients with essential hypertension. *Balkan medical journal*, 30(1):85, 2013.

[50] M. Henry, Y. Kitamura, and B. Salanié. Partial identification of finite mixtures in econometric models. *Quantitative Economics*, 5(1):123–144, 2014.

[51] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[52] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[53] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[54] C. Huang, Y. Li, C. Change Loy, and X. Tang. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5375–5384, 2016.

[55] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger. Convolutional networks with dense connectivity. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[56] D. R. Hunter and K. Lange. A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37, 2004.

[57] M. Hurn, A. Justel, and C. P. Robert. Estimating mixtures of regressions. *Journal of Computational and Graphical Statistics*, 12(1):55–79, 2003.

[58] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[59] L. Jacob, G. Obozinski, and J.-P. Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440. ACM, 2009.

[60] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[61] N. Jafarpour, D. Precup, M. Izadi, and D. Buckeridge. Using hierarchical mixture of experts model for fusion of outbreak detection methods. In *AMIA Annual Symposium Proceedings*, page 663, 2013.

[62] W. Jiang and M. A. Tanner. Hierarchical mixtures-of-experts for exponential family regression models: approximation and maximum likelihood estimation. *Annals of Statistics*, pages 987–1011, 1999.

[63] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[64] J. G. S. Z. Kai Tian, Yi Xu. Network as regularization for training deep neural networks: Framework, model and performance. In *34th AAAI Conference on Artificial Intelligence*, 2020.

[65] A. Khalili and J. Chen. Variable selection in finite mixture of regression models. *Journal of the american Statistical association*, 12(479):1025–1038, 2007.

[66] J. Kim and H. S. Mahmassani. A finite mixture model of vehicle-to-vehicle and day-to-day variability of traffic network travel times. *Transportation Research Part C: Emerging Technologies*, 46:83–97, 2014.

[67] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[68] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[69] T. A. Lasko, J. C. Denny, and M. A. Levy. Computational phenotype discovery using unsupervised feature learning over noisy, sparse, and irregular clinical data. *PloS one*, 8(6):e66341, 2013.

[70] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[71] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017.

[72] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. Feature selection: A data perspective. *arXiv preprint arXiv:1601.07996*, 2016.

[73] J. Li, N. Wu, Y. Li, K. Ye, M. He, and R. Hu. Cross-sectional analysis of serum calcium levels for associations with left ventricular hypertrophy in normocalcemia individuals with type 2 diabetes. *Cardiovascular diabetology*, 14(1):43, 2015.

[74] X. Li, J. Hect, M. Thomason, and D. Zhu. Interpreting age effects of human fetal brain from spontaneous fmri using deep 3d convolutional neural networks. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 1424–1427. IEEE, 2020.

[75] X. Li, X. Li, D. Pan, and D. Zhu. On the learning property of logistic and softmax losses for deep neural networks. In *AAAI*, pages 4739–4746, 2020.

[76] X. Li, D. Pan, X. Li, and D. Zhu. Learning compact features via in-training representation alignment. *arXiv preprint arXiv:2002.09917*, 2020.

[77] X. Li, D. Pan, and D. Zhu. Defending against adversarial attacks on medical imaging ai system, classification or detection? *arXiv preprint arXiv:2006.13555*, 2020.

[78] X. Li and D. Zhu. Robust feature selection via l2, 1-norm in finite mixture of regression. *Pattern Recognition Letters*, 108:15–22, 2018.

[79] X. Li and D. Zhu. Robust detection of adversarial attacks on medical images. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 1154–1158. IEEE, 2020.

[80] X. Li, D. Zhu, and M. Dong. Multinomial classification with class-conditional overlapping sparse feature groups. *Pattern Recognition Letters*, 101:37–43, 2018.

[81] X. Li, D. Zhu, M. Dong, M. Z. Nezhad, A. Janke, and P. D. Levy. Sdt: A tree method for detecting patient subgroups with personalized risk factors. *AMIA Summits on Translational Science Proceedings*, 2017:193, 2017.

[82] X. Li, D. Zhu, and P. Levy. Predictive deep network with leveraging clinical measure as auxiliary task. In *Bioinformatics and Biomedicine (BIBM), 2017 IEEE International Conference on*, pages 786–791. IEEE, 2017.

[83] X. Li, D. Zhu, and P. Levy. Leveraging auxiliary measures: a deep multi-task neural network for predictive modeling in clinical research. *BMC medical informatics and decision making*, 18(4):126, 2018.

[84] X. Li, D. Zhu, and P. Levy. Predicting clinical outcomes with patient stratification via deep mixture neural networks. *AMIA Summits on Translational Science Proceedings*, 2020:367, 2020.

[85] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

[86] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient l 2, 1-norm minimization. In *UAI*, pages 339–348, 2009.

[87] P. Liu, H. Cheng, T. M. Roberts, and J. J. Zhao. Targeting the phosphoinositide 3-kinase pathway in cancer. *Nature reviews Drug discovery*, 8(8):627–644, 2009.

[88] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *International Conference on Machine Learning*, pages 507–516, 2016.

[89] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *HLT-NAACL*, pages 912–921, 2015.

[90] W.-Y. Loh, X. He, and M. Man. A regression tree approach to identifying subgroups with differential treatment effects. *Statistics in medicine*, 34(11):1818–1833, 2015.

[91] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[92] J. Lu, G. Getz, E. A. Miska, E. Alvarez-Saavedra, J. Lamb, D. Peck, A. Sweet-Cordero, B. L. Ebert, R. H. Mak, A. A. Ferrando, et al. Microrna expression profiles classify human cancers. *nature*, 435(7043):834, 2005.

[93] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[94] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

[95] G. McLachlan and D. Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[96] L. Meier, S. Van De Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[97] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[98] R. Miotto, L. Li, B. A. Kidd, and J. T. Dudley. Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6:26094, 2016.

[99] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.

[100] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[101] W. K. Newey and D. McFadden. Large sample estimation and hypothesis testing. *Handbook of econometrics*, 4:2111–2245, 1994.

[102] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

[103] M. Z. Nezhad, D. Zhu, X. Li, K. Yang, and P. Levy. Safs: A deep feature selection approach for precision medicine. In *Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference on*, pages 501–506. IEEE, 2016.

[104] M. Z. Nezhad, D. Zhu, N. Sadati, K. Yang, and P. Levi. Subic: A supervised bi-clustering approach for precision medicine. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 755–760. IEEE, 2017.

[105] F. Nie, H. Huang, X. Cai, and C. H. Ding. Efficient and robust feature selection via joint l2, 1-norms minimization. In *NIPS*, pages 1813–1821, 2010.

[106] M. Olson, A. Wyner, and R. Berk. Modern neural networks generalize on small data sets. In *Advances in Neural Information Processing Systems*, pages 3619–3628, 2018.

[107] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[108] D. W. Parsons, T.-L. Wang, Y. Samuels, A. Bardelli, J. M. Cummins, L. DeLong, N. Silliman, J. Ptak, S. Szabo, J. K. Willson, et al. Colorectal cancer: mutations in a signalling pathway. *Nature*, 436(7052):792–792, 2005.

[109] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[110] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[111] J. P. Pestian, C. Brew, P. Matykiewicz, D. J. Hovermale, N. Johnson, K. B. Cohen, and W. Duch. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 97–104. Association for Computational Linguistics, 2007.

[112] A. Piovesan, N. Molineri, F. Casasso, I. Emmolo, G. Ugliengo, F. Cesario, and G. Borretta. Left ventricular hypertrophy in primary hyperparathyroidism. effects of successful parathyroidectomy. *Clinical endocrinology*, 50(3):321–328, 1999.

[113] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. Kim, L. C. Goumnerova, P. M. Black, C. Lau, et al. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436–442, 2002.

[114] K. Puniyani, S. Kim, and E. P. Xing. Multi-population gwa mapping via multi-task regularized regression. *Bioinformatics*, 26(12):i208–i216, 2010.

[115] S. Purushotham, C. Meng, Z. Che, and Y. Liu. Benchmarking deep learning models on large healthcare datasets. *Journal of biomedical informatics*, 83:112–134, 2018.

[116] Pytorch. `http://pytorch.org`.

[117] Y. Qi, O. Tastan, J. G. Carbonell, J. Klein-Seetharaman, and J. Weston. Semi-supervised multi-task learning for predicting interactions between hiv-1 and human proteins. *Bioinformatics*, 26(18):i645–i652, 2010.

[118] R. E. Quandt and J. B. Ramsey. Estimating mixtures of normal distributions and switching regressions. *Journal of the American statistical Association*, 73(364):730–738, 1978.

[119] N. Rao, C. Cox, R. Nowak, and T. T. Rogers. Sparse overlapping sets lasso for multi-task learning and its application to fmri analysis. In *Advances in neural information processing systems*, pages 2202–2210, 2013.

[120] N. Rao, R. Nowak, C. Cox, and T. Rogers. Classification with the sparse group lasso. *Signal Processing, IEEE Transactions on*, 64(2):448–463, 2016.

[121] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[122] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[123] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[124] P. Schlattmann. *Medical applications of finite mixture models*. Springer, 2009.

[125] M. L. Seltzer and J. Droppo. Multi-task learning in deep neural networks for improved phoneme recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6965–6969. IEEE, 2013.

[126] C. W. Seymour, J. N. Kennedy, S. Wang, C.-C. H. Chang, C. F. Elliott, Z. Xu, S. Berry, G. Clermont, G. Cooper, H. Gomez, et al. Derivation, validation, and potential treatment implications of novel clinical phenotypes for sepsis. *Jama*, 321(20):2003–2017, 2019.

[127] J. Shen, Y. Qu, W. Zhang, and Y. Yu. Wasserstein distance guided representation learning for domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[128] N. Simon, J. Friedman, and T. Hastie. A blockwise descent algorithm for group-penalized multiresponse and multinomial regression. *arXiv preprint arXiv:1311.6529*, 2013.

[129] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

[130] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[131] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[132] N. Städler, P. Bühlmann, and S. Van De Geer. L1-penalization for mixture regression models. *Test*, 19(2):209–256, 2010.

[133] J. E. Staunton, D. K. Slonim, H. A. Coller, P. Tamayo, M. J. Angelo, J. Park, U. Scherf, J. K. Lee, W. O. Reinhold, J. N. Weinstein, et al. Chemosensitivity prediction by transcriptional profiling. *Proceedings of the National Academy of Sciences*, 98(19):10787–10792, 2001.

[134] X. Su, C.-L. Tsai, H. Wang, D. M. Nickerson, and B. Li. Subgroup analysis via recursive partitioning. *Journal of Machine Learning Research*, 10(Feb):141–158, 2009.

[135] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43):15545–15550, 2005.

[136] H.-I. Suk, S.-W. Lee, D. Shen, A. D. N. Initiative, et al. Subclass-based multi-task learning for alzheimer's disease diagnosis. *Frontiers in aging neuroscience*, 6, 2014.

[137] H.-I. Suk and D. Shen. Deep learning-based feature representation for ad/mci classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 583–590. Springer, 2013.

[138] Y. Sun, S. Ioannidis, and A. Montanari. Learning mixtures of linear classifiers. In *ICML*, pages 721–729, 2014.

[139] Q. Suo, F. Ma, and et al. A multi-task framework for monitoring health conditions via attention-based recurrent neural networks. In *AMIA annual symposium proceedings*, volume 2017, page 1665. American Medical Informatics Association, 2017.

[140] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112,

2014.

[141] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[142] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[143] F. Tang, C. Xiao, F. Wang, and J. Zhou. Predictive modeling in urgent care: a comparative study of machine learning approaches. *JAMIA Open*, 1(1):87–98, 2018.

[144] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[145] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[146] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1):387–423, 2009.

[147] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

[148] M. Vincent and N. R. Hansen. Sparse group lasso and high dimensional multinomial classification. *Computational Statistics & Data Analysis*, 71:771–786, 2014.

[149] W. Wan, Y. Zhong, T. Li, and J. Chen. Rethinking feature distribution for loss functions in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9117–9126, 2018.

[150] L. Wang, M. Dong, E. Towner, and D. Zhu. Prioritization of multi-level risk factors for obesity. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1065–1072. IEEE, 2019.

[151] L. Wang, D. Zhu, E. Towner, and M. Dong. Obesity risk factors ranking using multi-task learning. In *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 385–388. IEEE, 2018.

[152] Y.-X. Wang, D. Ramanan, and M. Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems*, pages 7029–7039, 2017.

[153] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer, 2016.

[154] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A comprehensive study on center loss for deep face recognition. *International Journal of Computer Vision*, 127(6-7):668–683, 2019.

[155] I. R. White, P. Royston, and A. M. Wood. Multiple imputation using chained equations: issues and guidance for practice. *Statistics in medicine*, 30(4):377–399, 2011.

[156] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[157] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4460–4464. IEEE, 2015.

[158] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[159] C. Yadav and L. Bottou. Cold case: The lost mnist digits. In *Advances in Neural Information Processing Systems*, pages 13443–13452, 2019.

[160] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[161] D. Zhang, D. Shen, A. D. N. Initiative, et al. Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in alzheimer's disease. *NeuroImage*, 59(2):895–907, 2012.

[162] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu. Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4320–4328, 2018.

[163] Y. Zhang and Q. Yang. A surey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

[164] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014.

[165] Z. Zhang and M. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788, 2018.

[166] Z.-H. Zhou and X.-Y. Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010.

ABSTRACT

IMPROVING PREDICTIVE MODELING VIA EFFECTIVE FEATURE SELECTION AND REPRESENTATION LEARNING

by

XIANGRUI LI

August 2020

Advisor: Dr. Dongxiao Zhu

Major: Computer Science

Degree: Doctor of Philosophy

Predictive modeling (a.k.a. supervised learning) is a machine learning paradigm that has enormous important applications for real-world problems. With the recent surge of data in volume and complexity, effectively capturing the information in input features that is relevant to targets is critical to the success of predictive modeling. Tackling this challenge requires different techniques depending on the specific applications. In this dissertation, we develop several methods to improve the performance of predictive models.

Specifically, in the case of small n, large p problem, we propose two sparsity-inducing regularization methods for multi-class logistic regression and finite mixture of linear regression, respectively. Those regularization can not only improve predictive performance, but also greatly enhance model interpretability. As an important application of predictive modeling, in healthcare informatics, we develop two DNN models for risk prediction. We also study the learning property of logistic and softmax losses for deep neural networks; we derive a system of equations that quantitatively depicts the property of the decision boundary. Based on this property, an improved version of logistic loss is proposed. Finally,

to improve the training of DNNs with stochastic gradient descent, we develop a regularization method of feature alignment via maximum mean discrepancy.

AUTOBIOGRAPHICAL STATEMENT

Xiangrui Li received his B.S. and M.S. of Mathematics both from University of Science and Technology of China, China in 2010 and 2013 respectively. He is currently a Ph.D. candidate in the Lab of Machine Learning and Predictive Analytics led by Dr. Dongxiao Zhu, Department of Computer Science, Wayne State University. His research interests are on predictive modeling, machine learning and healthcare informatics.