Flexible Sparse Learning of Feature Subspaces

Yuting Ma

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2017

ABSTRACT

Flexible Sparse Learning of Feature Subspaces

Yuting Ma

It is widely observed that the performances of many traditional statistical learning methods degenerate when confronted with high-dimensional data. One promising approach to prevent this downfall is to identify the intrinsic low-dimensional spaces where the true signals embed and to pursue the learning process on these informative feature subspaces. This thesis focuses on the development of flexible sparse learning methods of feature subspaces for classification. Motivated by the success of some existing methods, we aim at learning informative feature subspaces for high-dimensional data of complex nature with better flexibility, sparsity and scalability.

The first part of this thesis is inspired by the success of distance metric learning in casting flexible feature transformations by utilizing local information. We propose a nonlinear sparse metric learning algorithm using a boosting-based nonparametric solution to address metric learning problem for high-dimensional data, named as the *sDist* algorithm. Leveraged a rank-one decomposition of the symmetric positive semi-definite weight matrix of the Mahalanobis distance metric, we restructure a hard global optimization problem into a forward stage-wise learning of weak learners through a gradient boosting algorithm. In each step, the algorithm progressively learns a sparse rank-one update of the weight matrix by imposing an L_1 regularization. Nonlinear feature mappings are adaptively learned by a hierarchical expansion of interactions integrated within the boosting framework. Meanwhile, an early stop-

ping rule is imposed to control the overall complexity of the learned metric. As a result, without relying on computationally intensive tools, our approach automatically guarantees three desirable properties of the final metric: positive semi-definiteness, low rank and element-wise sparsity. Numerical experiments show that our learning model compares favorably with the state-of-the-art methods in the current literature of metric learning.

The second problem arises from the observation of high instability and feature selection bias when applying online methods to highly sparse data of large dimensionality for sparse learning problem. Due to the heterogeneity in feature sparsity, existing truncation-based methods incur slow convergence and high variance. To mitigate this problem, we introduce a stabilized truncated stochastic gradient descent algorithm. We employ a soft-thresholding scheme on the weight vector where the imposed shrinkage is adaptive to the amount of information available in each feature. The variability in the resulted sparse weight vector is further controlled by stability selection integrated with the informative truncation. To facilitate better convergence, we adopt an annealing strategy on the truncation rate. We show that, when the true parameter space is of low dimension, the stabilization with annealing strategy helps to achieve lower regret bound in expectation.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, it is hard to overstate my gratitude to my Ph.D adviser and mentor, Professor Tian Zheng. During my years at Columbia University, she guided my way into academic research and offered me with invaluable advice in both work and life. Her passions in statistics and data science have provided me constant inspirations. Regardless of her tight schedule, she was always able to give me the most-needed suggestions. She encouraged me to try out new ideas and to constantly improve existing methods. It was also such an enjoyment to assist her teaching two courses at Columbia. I have to admitted that, because of her creativity in teaching, I have learned much more from being her teaching assistant than I first learned the subjects in classroom. Her advisership had made the challenging doctoral study much fun and fruitful.

I would like to thank my committee members, Professor Lauren Hannah, Professor Arian Maleki, Professor Pei Wang and Processor John Wright, for their insightful comments and suggestions. I would also like to thank all the professors and staffs, especially Dood Kalicharan and Anthony Cruz, at the Statistics Department, for their consistent helps throughout the years.

I wish to express my gratitude to my dearest friends inside and outside Columbia University. To my roommate and officemate Phyllis Wan who shared the joy and hardship of life with me for more than three years, to my cousin Yu Han who helped me to overcome the most difficult time and brought constant laughter and inspirations to my life, to Fei Long and Ran Xu who made the colors of my time at Columbia, to my fellow Ph.D students at the Statistics Department who brought me encouragement and inspirations, and to Xiao Yu who helped me revise this dissertation and takes on

a journey with me, without you all, my Ph.D life would be dull.

Lastly, but most importantly, I wish to thank my parents, Jian Ma and Jingping He. Thank you for not giving up on me when you saw me standing by the back wall of the classroom as a penalty when I was in elementary school. Thank you for having big hearts when you were constantly called by my teachers for complaining me never writing my homework. Thank you for holding belief on me when no one saw hope on me. Thank you for your patience, sacrifice, tolerance, supports, encouragement and everlasting love that I will never be able to pay back. You are the biggest fortune of my life.

To my parents

Chapter 1

Introduction

With advances in modern technology, an unprecedentedly large number of data are produced and stored. The dimensionalities of these datasets have exploded, usually in the exponential order of sample sizes, which makes extracting patterns from them a challenging problem for traditional statistical methods. Due to the large dimensionality, it is difficult to attain the generalizability and computational efficiency as in the low-dimensional counterparts. One consensus is that, instead of being truly high-dimensional, the data are embedded in a noisy high-dimensional space and thus can be efficiently summarized in feature subspaces of much lower dimensions. In order to improve accuracy and interpretability of the estimations, learning informative feature subspaces has become a necessary step for learning tasks in high dimensions. In this thesis, we dedicate to develop efficient learning algorithms for supervised learning of informative feature subspaces with high-dimensional data. Based on established learning schemes, we propose to induce sparsity via shrinkage for extracting low-dimensional informative feature subspaces. We construct flexible classifiers that are nonparametric and adapt to local variations with the ability to consider the potential nonlinear interactions among features. The learning processes are made possible and scalable by employing *stochastic learning* and *boosting framework*.

1.1 Background

The performances of many popular learning methods in low dimensional setting degenerate when confronted with high-dimensional data. One promising approach to lift this barrier is to learn intrinsic low-dimensional spaces where the true signal lie and pursue the learning process on these feature subspaces. The benefits of doing it can be seen from both theoretical and practical perspectives. First, learning a low-dimensional representation alleviates the renown problem of *curse of dimensionality*. Secondly, learning on a reduced feature subspace increases the interpretability of the resulted model for more transparent scientific explanations. Last but not the least, with a much smaller number of dimensions in consideration, the computations can be drastically accelerated with manageable storage.

The main reason that traditional learning methods in low-dimensional setting suffers in high dimensionality is due to the *curse of dimensionality* [Bellman, 1961]. Their performances deteriorate quickly as the dimension of the search space increases in terms of estimation accuracy, stability and computational efficiency. Particularly, distances, such as the Euclidean distance, serve as the foundations of many statistical learning algorithms that quantify the notion of similarity. However, it is shown in [Hastie *et al.*, 2009] that, as dimension increases, the differences in the Euclidean distances between pairs of data points diminish with finite sample size. Consider a simple scenario where n data points uniformly distribute in a p-dimensional unit ball centered at the origin. We can compute the median distance from the origin to the closet data point as follows:

$$d(p,n) = \left(1 - \frac{1}{2}^{1/n}\right)^{1/p}. \tag{1.1}$$

From (1.1), it is easy to see that when p is large, most data points lie on the boundary of the unit ball, making them indistinguishable in terms of distances. The situation is worsened by the rising computational costs for calculating the pairwise distances. As a result, many classical distance-based learning algorithms are not applicable for

high-dimensional data, including the k-Nearest Neighbor classifier (kNN) [Marimont and Shapiro, 1979], the kernel-based methods such as Support Vector Machine (SVM) [Cortes and Vapnik, 1995] and the kernel regression [Nadaraya, 1964], *etc.*. Moreover, it is discussed in [Hughes, 1968] that, with finite sample sizes, the predictive power diminishes as the dimensionality increases, which is known as the *Hugues phenomenon*. Therefore, in a high-dimensional feature space, when each feature only has a finite number of possible values, an enormously large training sample is required to achieve similar performance as in low-dimensional feature space. From the computational perspective, the algorithms need to search within an exponentially increasing space for closest neighbors. Some methods require amounts of time and memory that are exponential in the number of dimensions of the data. To resolve the curse of dimensionality, researchers usually assume that there are some intrinsic low-dimensional feature subspaces where lie the true signal of the data. With low-dimensionality, distances maintain their functionality as measures of similarity on these feature subspaces. Since less irrelevant and noisy features are used in the model, the estimations based on feature subspaces are less subject to noise and to over-fitting.

Not only are the data growing in volume and in dimensionality, but the understanding that people wish to gain from them is increasingly sophisticated. For example, in genetic studies, while predictive models could be based on the expression of more than a few thousands of genes, it is motivated to search for short lists of predictive genes that can concisely explain the observed patterns. The ability of interpreting the resulted model based on small sets of genes may shed lights on decoding biological processes involved in the disease and suggest novel findings for curing [Haury *et al.*, 2011]. Therefore, learning informative feature subspaces is also meaningful for real scientific applications to bring transparent interpretations. Unlike the mysterious "blackbox" algorithms, it provides tangible insights for scientific research. When the dimensions of feature subspaces are reduced to two- or three-dimensional, they also facilitate straightforward visualization of the data to reveal their complex structures.

The computational advantage with low-dimensional feature subspaces is rather obvious after previous discussions. The computational costs for matrix computations exponentially decrease as the dimensionalites are shrunkened. Using feature subpaces makes nonparametric distance-based learning algorithms possible even with large sample sizes. It also requires less intermediate storage spaces and a much smaller number of parameters. For instance, in the distance metric learning literature [Bellet *et al.*, 2013], to obtain an adaptive distance metric, one needs to store a $p \times p$ dense matrix to treat each feature discretionarily, where p is the data dimension. The computations between any pair of data points involve $O(p^2)$ complexity. Instead, in sparse metric learning, the distances are computed by considering only a d-dimensional feature subspace, where $d << p$. The needs of storage and distance computation are both quadratically cut down.

There exists voluminous works on feature subspace learning. From a different perspective, feature subspace learning is an analogy to dimension reduction, which has attracted much attention in statistics and machine learning communities during the past decades. In addition to reduce dimensions to manageable sizes, we contemplate more on the meanings of the reduced subspaces and the patterns of data points unfold on them, which is the reason we focus on the learning of feature subspaces. By and large, feature subspace learning methods can be classified into two categories: *feature selection* methods and *feature transformation* methods.

Feature selection methods focus on identifying a subset of individual features that are considered to be significant for the learning objectives. Usually, they are categorized into the following two types or their mixture depending on how the selection procedure is combined with the model building [Guyon and Elisseeff, 2003].The *filter methods* evaluate the intrinsic characteristics of the features regardless of the learners used. This group of methods includes the siginificance-test-based methods such as the Golub's weighted voting method [Golub *et al.*, 1999], the significance analysis [Tusher *et al.*, 2001] and information theoretic methods [Ding and Peng, 2005]. On

the controrary, the *wrapper methods* are embedded within the specific learners, in which the feature selection criteria is related to the learning performance, such as LASSO [Tibshirani, 1996a], Recursive Support Vector Machine (R-SVM) [Zhang *et al.*, 2006] and Random Forest [Breiman, 2001], *etc.*. Depending on how the selected features would be used, the resulted feature subspaces can be formed either by treating each selected feature as an one-dimensional feature subspace or by concatenating the entire set of selected features as one multi-dimensional feature subspace.

On the other hand, feature transformation methods seek for transformations of the original input feature space to spaces of lower dimensions. Unsupervised learning and supervised learning are two different setups for this kind of methods. In unsupervised problems, as no response variable is given, the algorithms pursue the goal of extracing low-dimensional embedding while preserving the original spatial relationships among data points as they are in the original high-dimensional space. Among unsupervised feature transformation methods, the most famous linear approaches should be the Principal Component Analysis (PCA) [Pearson, 1901]. Nonlinear unsupervised feature transformation is often referred to as *manifold learning*. Popular methods include ISOMAP [Tenenbaum *et al.*, 2000], locally linear embedding [Roweis and Saul, 2000], laplacian eigenmaps [Belkin and Niyogi, 2002], stochastic neighborhood embedding [Hinton and Roweis, 2002], *etc.*. In the context of supervised learning, the goal can be translated as to searching for feature subspaces on which data points with similar response variable are drawn closer to each other while dissimilar ones are dispelled further away. Linear discriminant analysis [Fisher, 1936] is a classical example of linear approaches for supervised learning. Additive models, such as multiplicative adaptive regression splines (MARS) [Friedman and others, 1991], learn a set of feature subspaces which can be either linear or nonlinear transformations of the original features. In the extensions, multilinear subspace learning and its counterparts construct low-dimensional subspaces for tensor data [Vasilescu and Terzopoulos, 2003]. Metric learning [Yang and Jin, 2006], [Bellet *et al.*, 2013] is a general class of methods

that offers a variety of flexible methods for learning linear transformations of the input features with extensions to nonlinear ones. It is seen in the literature that feature transformation is overall more effective than feature selection for high-dimensional data. However, without sparsity regularization, the learned feature subspaces are subject to the lack of interpretability if either the number of involved input features or the number of subspaces is large.

1.2 Motivation

Inspired by the success of various feature subspace learning methods, we are motivated to extend the state-of-the-art methods with some characteristics which we believe can help to accommodate high-dimensional data of more complex nature with better efficiency: flexibility, sparsity and scalability.

In the mainstream of statistical research, linear models retain their popularity despite of their simplicity. In both regression and classification problems, they enjoy theoretical soundness, practical effectiveness, and easy interpretation. However, with growing complexity in modern data, linear models are no longer capable of fulfilling the needs of modeling and predictions. It is very unlikely that the true relationships between the response variable and the input features are simply linear, especially in the high-dimensional settings. For instance, in microarray data, co-expressions of genes are often observed in biological conditions and processes which cannot be captured by linear model. Some crucial genes are only influential when jointly considered with others. Therefore, it is desirable to have the ability of modeling complex data with greater flexibility. In order to achieve flexibility, there are several approaches that can be utilized. Firstly, we can leverage information of local structures. With a reasonably large set of training data, we can always approximate theoretically optimal conditional expectations by nearest-neighbors averages. Relying on local information, it requires no assumption on the underlying global model, which, regardless of the

form, is very likely to be wrong for high-dimensional data. Local methods, such as the simple yet effective k-Nearest Neighbor (kNN) [Cover and Hart, 1967], have gained lasting popularity owning to their nature adaptions to irregular and local variations. Nevertheless, local methods are based on a foundamental assumption that data points that are close in Euclidean distance are similar to each other, which might not be valid for high dimensional data due to the curse of dimensionality on distance, articulated in Section 1.1. The question remains in how local information shall be used with the presense of a large amount of noises and how spatial information can be translated into semantical similarity. Another vehicle is to include nonlinearity in order to evolving from linear models. Polynomials, splines [Craven and Wahba, 1978] and kernel methods [Nadaraya, 1964] are widely used elements for achieving nonlinearity. With a large number of input features, learning with all polynomial terms is impractical subject to prohibitively high computational costs. The selection of significant polynomial terms is an NP-hard problem. Alternatively, kernel methods are extensively studied for their power of modeling highly nonlinear data and for the "kernel trick" that enables manageable computation [Hofmann *et al.*, 2008]. Whearas with kernel functions, the basis feature space is expanded to possibly infinite dimension. The interpretation of the resulted model becomes unclear in the original feature space. Moreover, it is noted that many kernel-based methods do not scale with large sample sizes [Rahimi and Recht, 2007]. In all, for high-dimensional data, it is still very challenging to cast a flexible model with desired properties, such as locality and nonlinearity, and with manageable computational costs and straightforward interpretations.

On the other hand, in the recent years we have witnessed an explosion of research interests in deep architectures based on neural network. Deep architectures learn complex mapping by transforming inputs through multiple layers of nonlinear processing with millions or even billions parameters. However, for the sake of its overwhelming complications, deep learning methods are often seen as "blackbox" algorithms whose

results cannot be easily interpreted. They involve difficult nonlinear optimization and many heuristics that conceal the underlying mechanism from discovery. Like many, we are intrigued by the success of models with complex structure yet drawn to the principle of parsimony: the simplest explanation of a given phenomenon should be preferred over more complicated ones. Hence, it is our motivation to steer the feature engineering adaptively with the learning process to achieve both sparsity in parameter space and good interpretation. We aim at casting a strategy that learns a low-dimensional informative features subspace even when flexible approaches are adopted.

Another major bottleneck that obstructs high-dimensional data modeling is the high computational costs. As mentioned in Section 1.1, learning low-dimensional feature subspaces in high-dimensional data leads to hard combinatorial problems [Natarajan, 1995]. The flexibility merits that we are looking for, such as locality and nonlinearity, are usually computationally expensive in traditional methods. For example, metric learning methods exploit local information, using distance as measure of similarity. However, they often involve the burdensome semi-definite programming, which has worst-case complexity of $O(p^{6.5})$, where p is the data dimension. Kernel methods is a classical option to introduce nonlinearity that take advantage of the "kernel trick". Yet it is not scalable as the storage and computation of the kernel matrix take $O(n^2)$ and $O(n^2p)$ respectively, where n is the sample size. Another computational concern rises when there are too many tuning parameters. Normally the final decision is made based on cross-validations by searching over a grid of parameter combinations. If an algorithm is not designed to automatically tailor to the data at hand, the computational cost might be multiplied for a scrutiny of the optimal parameter values. Consequently, in real applications, it is unrealistic to pursue good performance without constraining computational complexity for high-dimensional data. Whereas, the tradeoff between performance and computational efficiency is not straighforward and can be task-specific. Moreover, for many high-dimensional data, it is often ob-

served that the data matrices are highly sparse. It remains unclear of how one can exploit the data sparsity to devise efficient computation at the same time as streamlining the search of informative feature subspaces.

1.3 A Brief Overview

In this thesis, we address the problem of feature subspace learning for classification. The goal is to develop flexible frameworks to induce sparsity in feature space. We incorporate local information under a non-parametric framework such that the algorithm adapts to local variations. We further extend to nonlinear feature subspaces to consider potential higher-order interactions among features and to increase model flexibility. We resort to boosting algorithm and online learning framework to ensure scalability for data with both high dimensionality and massive sample size. These techniques not only provide wheels for increasing computational efficiency but also controls on over-fitting.

In the first part of this thesis, Chapter 2, is inspired by the success of distance metric learning in casting flexible linear transformation of feature space. We propose a nonlinear sparse metric learning algorithm using a boosting-based non-parametric solution to address metric learning problem for high-dimensional data, named as the *sDist* algorithm. In distance metric learning, we consider the Mahalanobis distance as a generalization of the Euclidean distance. The Mahalanobis distance between any two points $\mathbf{x}_i$ and $\mathbf{x}_j$ is parametrized by a symmetric and positive semi-definite weight matrix W:

$$d_W(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T W (\mathbf{x}_i - \mathbf{x}_j)}.$$

From a supervised learning perspective, a "good" Mahalanobis distance metric is supposed to draw similar points closer in distance and to pull dissimilar points further away. A major challenge of distance metric learning is the optimization for the weight matrix W based on the given similarity/dissimilarity constraints while retain-

ing its symmetry and positive semi-definiteness. Traditional methods may incur up to $O(p^{6.5})$ computational complexity by using semi-definite programming. The novel contribution of our approach is that we mathematically convert a hard global optimization problem into a sequence of simple local optimization via boosting. By recognizing a rank-one decomposition of the weight matrix W, the additive structure of forward stagewise learning automatically guarantees the required properties of symmetry and positive semi-definiteness of W without resorting computationally intensive tools. A gradient boosting algorithm is devised to obtain a feature-wise sparse rank-one update of the weight matrix at each step. To extend from linear distance metric, nonlinear features are progressively and adaptively learned by a hierarchical expansion of interactions incorporated within the boosting algorithm. By including solely the interactions that appear to be necessary with respect to the learning objective, we manage to identify significant nonlinear feature subspaces while maintaining low computational costs and good interpretability. Meanwhile, an early stopping rule is imposed to control the overall complexity of the learned metric. As a result, our approach guarantees three desirable properties of the final metric: positive semi-definiteness, low rank and element-wise sparsity. Feature selection might be carried out as a spontaneous by-product of our algorithm that provides insights of variable importance not only marginally but also jointly in higher orders. Numerical experiments show that our learning model compares favorably with the state-of-the-art methods in the current literature of metric learning.

Although the *sDist* algorithm achieves preferable performances for learning informative feature subspaces in high-dimensional space, it is not yet effective if the data is not only of large dimenaionality but also highly sparse. High-dimensional sparse data are observed in many important applications, such as the bag-of-word features in text mining and mass-spectrometric data in chemical analysis. Different from data with dense features, learning with sparse data is notoriously hard. Since the locations of nonzero entries are covered up by the predominant number of zero entries, learning

with highly sparse data usually incurs severe instability and computational difficulties. Particularly, online learning is subject to high fluctuations of the occurrences of nonzero entries. It requires extra efforts to identify the locations of nonzero entries. In the second part of this thesis, Chapter 3, we will address this particular computational issue of high-dimensional sparse data for sparse online learning problem. We develop a novel tool based on stochastic gradient descent (SGD), which is one of the most commonly used optimization methods in large-scale machine learning problems. [Langford *et al.*, 2009] introduce a sparse online learning method to induce sparsity via truncated gradient. With high-dimensional sparse data, however, this method suffers from slow convergence and high variance due to the heterogeneity in feature sparsity. To mitigate this issue, we introduce a stabilized truncated stochastic gradient descent algorithm. We employ a soft-thresholding scheme on the weight vector where the imposed shrinkage is adaptive to the amount of information available in each feature during the online learning process. The variability in the resulted sparse weight vector is further controlled by stability selection integrated with the informative truncation. To facilitate better convergence, we adopt an annealing strategy on the truncation rate. This technique leads to a balanced trade-off between exploration and exploitation in learning a sparse weight vector. We show that, in expectation, the stabilization with annealed rejection rate helps attain lower regret bound the original non-stabilized algorithm as well as faster convergence. Numerical experiments show that our algorithm compares favorably with the original algorithm in terms of prediction accuracy, attained sparsity and stability. The proposed method is not only beneficial for improving the *sDist* algorithm but also other sparsity-inducing algorithms for high-dimensional sparse data.

In the last chapter of this thesis, Chapter 4, we present potential applications of the proposed algorithms in machine learning, particularly in computer vision. In the applications, we integrate the stabilized truncated stochastic gradient descent algorithm of Chapter 3 into the sparse metric learning framework of Chapter 2 as the

computational engine. Such a combination boosts the performance of *sDist* framework and makes it applicable for high-dimensional applications with highly sparse entries. In computer vision applications, the resulted feature subspaces provide visible interpretation of the key factors for distinguishing different classes. On the other hand, by recognizing kernel as a generalized distance measure, we can further improve the flexibility of the proposed method by exploring a greater, even infinitely dimensional, feature space. To this end, we extend the distance metric learning framework of the *sDist* algorithm in Chapter 2 to kernel machine. Existing kernelized distance metric learning methods rely on the "kernel trick" and are obstructed by the prohibitively large kernel matrix applied on data with large sample sizes. Instead, we identify a random process associated with a positive-definite translation-invariant kernel function. By constructing random feature vectors with tractable computational cost, we obtain an unbiased estimation of the kernel function which is used to decode the kernel learning problem. The random features are sparsified by imposing a *first-order Markov chain prior* on the nonzero atoms to identify low-dimensional feature subspaces. This part of works directs our future research. At last, we conclude this thesis by summarizing distinctive characteristics and common merits of the proposed methods.

Chapter 2

Boosted Sparse Nonlinear Distance Metric Learning

2.1 Introduction

Beyond its physical interpretation, distance can be generalized to quantify the notion of similarity, which puts it at the heart of many learning methods, including the k-Nearest Neighbors (kNN) method, the k-means clustering method and the kernel regressions. The conventional Euclidean distance treats all dimensions equally. With the growing complexity of modern datasets, however, Euclidean distance is no longer efficient in capturing the intrinsic similarity among individuals given a large number of heterogeneous input variables. This increasing scale of data also poses a curse of dimensionality such that, with limited sample size, the unit density of data points is largely diluted, rendering high variance and high computational cost for Euclidean-distance-based learning methods. On the other hand, it is believed that the true informative structure with respect to the learning task is embedded within an intrinsic low-dimensional manifold [Johnson and Lindenstrauss, 1984], on which model-free distance-based methods, such as kNN, are capable of taking advantage of the inherent structure. It is therefore desirable to construct a generalized measure of distance in

a low-dimensional nonlinear feature space for improving the performance of classical distance-based learning methods when applied to complex and high dimensional data.

We first consider the Mahalanobis distance as a generalization of the Euclidean distance. Let $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a set of points in a feature space $\mathcal{X} \subseteq \mathbb{R}^p$. The Mahalanobis distance metric between any two points $\mathbf{x}_i$ and $\mathbf{x}_j$ parameterized by a weight matrix W is given by:

$$d_W(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T W (\mathbf{x}_i - \mathbf{x}_j)}, \tag{2.1}$$

where $W \in \mathbb{R}^{p \times p}$ is symmetric positive semi-definite (PSD), denoted as $W \succeq 0$. The Mahalanobis distance can also be interpreted as the Euclidean distance between the points in a new (sub)space that are linearly projected by L:

$$d_W(\mathbf{x}_i, \mathbf{x}_j) = ||L(\mathbf{x}_i - \mathbf{x}_j)||_2, \tag{2.2}$$

where $LL^T = W$ can be found by the Cholesky Decomposition. From a general supervised learning perspective, a "good" Mahalanobis distance metric for an outcome y at $\mathbf{x}$ is supposed to draw samples with similar y values closer in distance based on $\mathbf{x}$, referred to as the *similarity objective*, and to pull dissimilar samples further away, referred to as the *dissimilarity objective*, in the projected space.

There has been considerable research on the data-driven learning of a proper weight matrix W for the Mahalanobis distance metric in the field of *distance metric learning*. A comprehensive survey [Yang and Jin, 2006] pointed out that both accuracy and efficiency of distance-based learning methods can significantly benefit from using the Mahalanobis distance with a proper W. A detailed comparison with related methods is presented in Section 2.5. While existing algorithms for metric learning have been shown perform well across various learning tasks, each is not sufficient in dealing with some basic requirements. First, a desired metric should be flexible in adapting local variations as well as capturing nonlinearity in the data. Second, in high-dimensional settings, it is preferred to have a sparse and low-rank weight matrix W for better generalization with noisy inputs and for increasing interpretability of the

fitting model. Finally, the algorithm should be efficient in preserving all properties of a distance metric and be scalable with both sample size and the number of input variables.

In this chapter, we propose a novel method for a local sparse metric in a nonlinear feature subspace for binary classification, which will be referred to as *sDist*. Our approach constructs the weight matrix W through a gradient boosting algorithm that produces a sparse and low-rank weight matrix in a stage-wise manner. Nonlinear features are adaptively constructed within the boosting algorithm using a hierarchical expansion of interactions. The main and novel contribution of our approach is that we mathematically convert a global optimization problem into a sequence of simple local optimization via boosting, while efficiently guaranteeing the symmetry and the positive semi-definiteness of W without resorting to the computationally intensive semi-definite programming. Instead of directly penalizing on the sparsity of W, *sDist* imposes a sparsity regularization at each step of the boosting algorithm that builds a rank-one decomposition of W. The rank of the learned weight matrix is further controlled by the sparse boosting method proposed in [Bühlmann and Yu, 2006]. Hence, three important attributes of a desirable sparse distance metric are automatically guaranteed in the resulting weight matrix: positive semi-definiteness, low rank and element-wise sparisty. Moreover, our proposed algorithm is capable of learning a sparse metric on nonlinear feature space, which leads to a flexible yet highly interpretable solution. Feature selection might be carried out as a spontaneous by-product of our algorithm that provides insights of variable importance not only marginally but also jointly in higher orders.

This chapter is organized as follows. In Section 2 we briefly illustrate the motivation for our method using a toy example. Section 3 dissects the global optimization for linear sparse metric learning into a stage-wise learning via gradient boosting algorithm. Section 4 extends the framework in Section 3 to the nonlinear sparse metric learning by hierarchical expansion of interactions. Section 5 provides some practical

remarks on implementing the proposed method in practice. Results from numerical experiments are presented in Section 6. Finally, Section 7 concludes by summarizing our main contributions and sketching several directions of future research.

2.2 An Illustrative Example

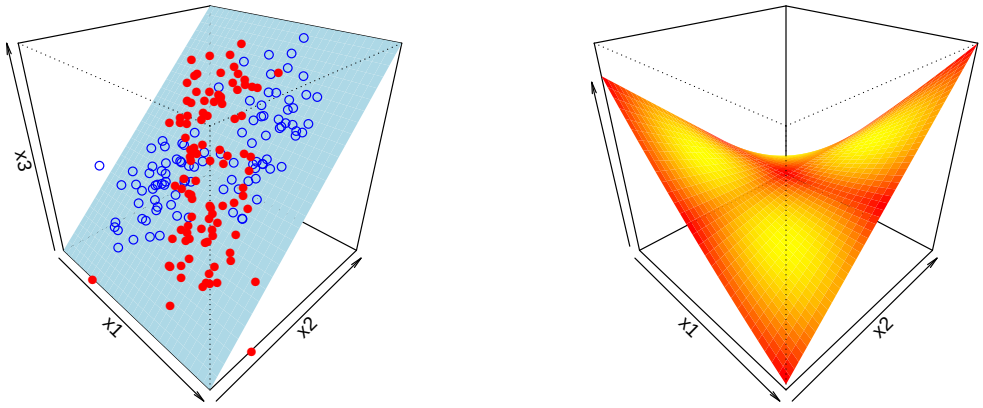


Figure 2.1: An illustrative example of the XOR binary classification problem. *Left*: Training dataset is consisted of sample points from two classes that are distributed on four clusters aligned at the crossing diagonals on a three-dimensional plane. 200 data points are generated from four bivariate Gaussian distributions and are projected into the designated three-dimensional plane as illustrated in the figure. *Right*: The transformed subspace learned by the *sDist* algorithm. Two horizontal dimensions are the first two input variables selected by *sDist*, that is, x_1 and x_2 in this case. The vertical dimension z is the first principal component of the transformed subspace defined as $L\phi(\mathbf{x})$, $LL^T = W$, which displays the overall shape of the surface on which new distances are computed. The colors on the grid indicates the true class probability of each class on the log scale. The yellow color indicates high probability in the class generative probability distribution and the red color indicates low probability. Since it is difficult to visualize two overlapping probability distributions in one plane, we use the same color scale for both classes and just focus on the magnitude of class generative probability distributions in each area.

Before introducing the details of *sDist*, we offer here a toy example in Figure 2.1 to illustrate the problem of interest. The *left* panel of Figure 2.1 demonstrates the

classical binary classification problem XOR (Exclusive OR) in a 3-dimensional space, in which sample points with the same class label are distributed in two clusters positioned diagonally from each other. The XOR example is commonly used as a classical setting for nonlinear classification in the literature. In the original space, sample points cannot be linearly separated. It is also observed that the vertical dimension x_3 is redundant, as it provides no additional information regarding the class membership aside from x_1 and x_2. Hence, it is anticipated that there exists a nonlinear subspace on which points on the opposite diagonals of the tilted surface are closer to each other. The subspace is also expected to be constructed solely based on a minimum set of variables that are informative about the class membership. The *right* panel of Figure 2.1 is the transformed subspace learned by the proposed *sDist* algorithm, which is only based on the informative variables x_1 and x_2. In particular, the curved shape of the resulted surface ensures that sample points with the same class label are drawn closer and those with opposite label are pulled further apart.

2.3 Boosted Linear Sparse Metric Learning

In this section, we first discuss the case of learning a linear sparse metric. Extension to nonlinear metric is discussed in Section 4. Assume that we are given a dataset $\mathcal{S} = \{\mathbf{x}_i, y_i\}$, $i = 1, \ldots, N$, $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^p$, where $\mathcal{X}$ is the input feature space and p is the number of dimensions of the input vector[1] and the class label $y_i \in \{-1, 1\}$. The original feature space consists of all input variables $x_1, \ldots, x_p$. Consider an ideal scenario where there exists a metric parametrized by W such that, in the W-transformed space, classes are separable. Then a point should, on average, be closer to the points from the same class than to the ones from the other class in its local neighborhood. Under this proposition, we propose a simple but intuitive discriminant

[1]For simplicity, we only consider datasets with numerical features in this chapter, on which distances are naturally defined.

function at $\mathbf{x}_i$ between classes characterized by W:

$$f_{W,k}(\mathbf{x}_i) = d^-_{W,k}(\mathbf{x}_i) - d^+_{W,k}(\mathbf{x}_i) \tag{2.3}$$

with

$$d^-_{W,k}(\mathbf{x}_i) = \frac{1}{k} \sum_{j \in S^-_k(\mathbf{x}_i)} (\mathbf{x}_i - \mathbf{x}_j)^T W (\mathbf{x}_i - \mathbf{x}_j)$$
$$d^+_{W,k}(\mathbf{x}_i) = \frac{1}{k} \sum_{j \in S^+_k(\mathbf{x}_i)} (\mathbf{x}_i - \mathbf{x}_j)^T W (\mathbf{x}_i - \mathbf{x}_j)$$

where $S^+_k(\mathbf{x}_i)$ and $S^-_k(\mathbf{x}_i)$ are the set of k nearest neighbors of $\mathbf{x}_i$ with the same class labels and with the opposite class labels as y_i, respectively. The predicted class label is obtained by $\hat{y} = 1$ if $\hat{f}_W(x) > 0$ and $\hat{y} = -1$ otherwise. For simplicity, we will now drop k in the notations $d^-_{W,k}$ and $d^+_{W,k}$ as k is fixed throughout the algorithm.

The base classifier in (2.3) serves as a continuous surrogate function of the kNN classifier, which is differentiable with respect to the weight matrix W. Instead of using the counts of the negative and the positive sample points in local neighborhoods, we adopt the continuous value of distances between two class to indicate the affinity to the negative and the positive classes. In Figure 2.2, we present a detailed comparison of the performance of the proposed classifier (2.3) with the kNN classifier at different values of k on the real dataset Ionosphere. It suggests that, with small k ($k \leq 11$) which is normally used in neighborhood-based method, f_W consistently outperforms kNN classifier with aligned pattern in terms of the average test errors based on 20 randomly partitioned cross-validations.

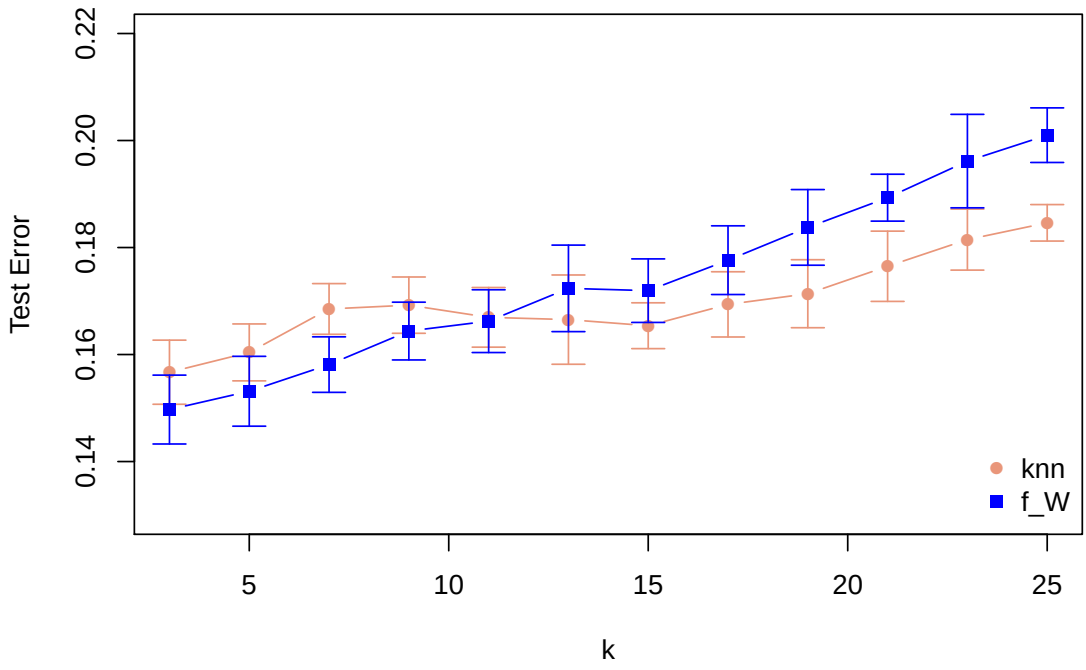


Figure 2.2: Comparison between f_W in (2.3) and the kNN classifier in terms of the average test errors based on 20 randomly partitioned 5-fold cross-validations using the real dataset Ionosphere.

Without any prior information , the local neighborhoods are first identified using the Euclidean distance. As we will introduce later in the gradient boosting algorithm, the local neighborhoods can be updated according to the learned distance metric regularly as the algorithm proceeds. The frequency of local neighborhood updates is determined based on the tradeoff between accuracy and computational costs in various applications. When the domain knowledge of local similarity relationships are available, local neighborhoods can be constructed with better precision.

Alternatively, $f_W(\mathbf{x}_i)$ can be represented as a inner product between the weight matrix W and the data information matrix D defined below which contains all infor-

mation of training sample point $\mathbf{x}_i$ for classification:

$$\hat{f}_W(\mathbf{x}_i) = \langle D_i, W \rangle, \tag{2.4}$$

where

$$D_i = \frac{1}{k}\left[\sum_{j \in S_k^-(\mathbf{x}_i)} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T - \sum_{j \in S_k^+(\mathbf{x}_i)} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T\right]$$

and $\langle \cdot, \cdot \rangle$ stands for the inner product for vectorized matrices. This alternative formulation of $\hat{f}_W(\mathbf{x}_i)$ suggests an effective strategy for improving the computation of W since D_i can be pre-calcuated and remains fixed when updating W.

For learning W, we evaluate the performance of this classifier $f_W(\mathbf{x}_i)$ using the exponential loss, which is commonly used as a smooth objective function in binary classification:

$$L(\mathbf{y}, f_W) = \sum_{i=1}^{N} L(y_i, f_W(\mathbf{x}_i)) = \sum_{i=1}^{N} \exp(-y_i \langle D_i, W \rangle) \tag{2.5}$$

Our learning task is then translated to derive a weight matrix W on the original feature space that minimizes the loss function in (2.5). The optimization of this objective function, however, is generally intractable for high dimensional data. Our proposed method, *sDist*, seeks optimization of the objective function via optimizing adaptable sub-problems such that a feasible solution can be achieved. In short, the building block of *sDist* are: a gradient boosting algorithm which learns a rank-one update of the weight matrix W at each step; a sparsity regularity on each rank-one update to enforce the element-wise sparsity and reduced rank of weight matrix with simultaneous preservation of the positive semi-definiteness, and a sparse boosting criterion that regularizes over the total number of boosting steps to achieve sparsity in the overall complexity of the final weight matrix.

2.3.1 Metric Learning via Boosting

In the distance metric learning literature, much effort has been put forward to learn the weight matrix W by solving a single optimization problem globally, as in PGDM

[Xing *et al.*, 2002] and LMNN [Blitzer *et al.*, 2005]. However, this optimization task turns out to be either computationally intractable or susceptible to local optima.

Boosting [Freund, 1995] offers a stagewise alternative to a single optimization. The motivation for boosting is that one can use a sequence of small improvements to derive a better global solution. Under the classification setting, boosting combines the outputs of many *weak* learners trained sequentially to produce a final aggregated classifier. Here, a weak learner is a classifier that is constructed to be only modestly better than a random guess. Subsequent *weak* learners are trained with more weights on previously mis-classified cases, which reduces dependence among the trained learners and produces a final learner that is both stable and accurate. Such an ensemble of *weak* learners has been proven to be more powerful than a single complex classifier and has better generalization performance [Hastie *et al.*, 2009]. In [Shen *et al.*, 2009] and [Bi *et al.*, 2011], a boosting algorithm has been implemented for learning a full distance metric, which has motivated the proposed algorithm in this chapter. More specifically, their important theorem on trace-one semi-definite matrices is also central to the theoretical basis of our approach.

Adopting a boosting scheme, *sDist* is proposed to learn a weight matrix W in a stepwise fashion to avoid over-fitting to the training data in one optimization process. To construct the gradient boosting algorithm, we first decompose the learning problem into a sequence of weak learners. It is shown in [Shen *et al.*, 2009] that for any symmetric positive semi-definite matrix $W \in \mathbb{R}^{p\times p}$ with trace 1, it can be decomposed into a linear convex span of symmetric positive semi-definite rank-one matrices:

$$W = \sum_{m=1}^{M} w_m Z_m, \quad \text{rank}(Z_m) = 1 \text{ and } tr(Z_m) = 1, \tag{2.6}$$

where $w_m \geq 0$, $m = 1, \ldots, M$, and $\sum_{i=1}^{M} w_m = 1$. The parameter $M \in \mathbb{Z}^+$ is the number of boosting iterations. Since any symmetric rank-one matrix can be written as an

outer product of a vector to itself. We further decompose W as

$$W = \sum_{m=1}^{M} w_m \xi_m \otimes \xi_m, \quad ||\xi_m||_2 = 1 \text{ for all } m = 1, 2, \ldots, M. \tag{2.7}$$

Based on this decomposition, we propose a gradient boosting algorithm that, within each step m, learns a rank-one matrix $Z_m = \xi_m \otimes \xi_m$ and its non-negative weight w_m. Each learned Z_m can be regarded as a small transformation of the feature space in terms of scaling and rotation. We use the following base learner in the gradient boosting algorithm:

$$g_m(\mathbf{x}_i) = \langle D_i, Z_m \rangle. \tag{2.8}$$

In consecutive boosting steps, the target discriminant function is constructed as a stage-wise additive expansion. At the m^{th} step, the estimated discriminant function is updated by adding the base learner $g_m(\cdot)$ with weight w_m to the existing weight matrix $\hat{W}_{m-1}$ that is learned from the previous $m-1$ steps:

$$\begin{aligned} f_{W_m}(\mathbf{x}_i) &= f_{W_{m-1}}(\mathbf{x}_i) + w_m g_m(\mathbf{x}_i) \\ &= \langle D_i, \sum_{j=1}^{m-1} w_j Z_j \rangle + w_m \langle D_i, Z_m \rangle \\ &= \langle D_i, \hat{W}_{m-1} + w_m Z_m \rangle = \langle D_i, \hat{W}_m \rangle \end{aligned}$$

where the resulted composite $\hat{W}_m$ is shown to be a weighted sum of Z_m's learned from all previous steps. Therefore, the rank-one matrices obtained at each boosting step are assembled to construct the desired weight matrix, reversing the decomposition in (2.7). In this process, the required symmetry and positive semi-definiteness of weight matrix are automatically preserved without imposing any constraint on the base learners. Moreover, the number of total boosting steps M caps the overall rank of the final weight matrix. Thus, by selecting an appropriate M, we can achieve an optimal reduced rank distance metric that we will discuss with more detail in Section 3.3.

In the gradient boosting algorithm, the learning goal is to attain the minimum of the loss function in (2.5). We achieve it by using a steepest-descent minimization in the functional space of f_W in (2.3), which is characterized by the weight matrix W. The optimization problem in each boosting step is divided into two substeps, for $m = 1, \ldots, M$:

- **Finding the rank-one matrix Z_m given the previous aggregation $\hat{W}_{m-1}$.** The residuals from the previous $m-1$ steps are:

$$r_i^{(m)} = \left[-\frac{\partial L(y_i, f)}{\partial f} \right]_{f=f_{\hat{W}_{m-1}}} = y_i \exp(-y_i f_{\hat{W}_{m-1}}(\mathbf{x}_i)) \tag{2.9}$$

for $i = 1, \ldots, n$. Then the next rank-one matrix Z_m is obtained by minimizing the loss function on the current residuals for a new weak learner of the form, that is,

$$Z_m = \underset{Z \in \mathbb{R}^{p \times p},\ \mathrm{rank}(Z)=1}{\arg\min} \sum_{i=1}^{n} L(r_i^{(m)}, g(\mathbf{x}_i)) = \underset{Z}{\arg\min} \sum_{i=1}^{n} \exp(-r_i^{(m)} \langle D_i, Z \rangle). \tag{2.10}$$

Recall that

$$r_i^{(m)} g_m(\mathbf{x}_i) = r_i^{(m)} \langle D_i, Z_m \rangle = r_i^{(m)} \langle D_i, \xi_m \otimes \xi_m \rangle = \xi_m^T (r_i^{(m)} D_i) \xi_m.$$

Hence (2.10) is equivalent to identifying

$$\xi_m = \underset{\xi \in \mathbb{R}^p,\ ||\xi||_2 = 1}{\arg\min} \sum_{i=1}^{n} \exp(-\xi^T r_i^{(m)} D_i \xi). \tag{2.11}$$

The rank-one update of weight matrix is calculated as $Z_m = \xi_m \otimes \xi_m$.

Optimization in (2.11) with the corresponding residuals $r_i^{(m)}$'s weight heavier on sample points with small margins in the feature space transformed by the distance metric learned from all previous steps. Thus, the boosting algorithm progressively improves over the "difficult" points in terms of discerning its class membership.

However, (2.11) is non-convex and suffers from local minima and instability. Instead of pursuing the direct optimization on the objective function in (2.11), we resort to an approximation of it by the first order Taylor expansion, which is commonly used in optimizing non-convex exponential objective functions. It allow us to take advantage of the exponential loss in the binary classification task as well as avoid the expensive computational cost of considering a higher order of expansion. This approximation results in a simpler convex minimization problem :

$$\xi_m = \underset{\xi \in \mathbb{R}^p,\ ||\xi||_2=1}{\arg\min} \ -\xi^T A_m \xi \tag{2.12}$$

where $A_m = \sum_{i=1}^{n} r_i^{(m)} D_i$.

It is worthnoting that solving (2.12) is equivalent to computing the the eigenvector associated with the largest eigenvalue of A_m via eigen-decomposition.

- **Finding the positive weight w_m given Z_m**: The optimal weight in the m^{th} step minimizes (2.5) given the learned Z_m from the previous step. Specifically, with $g_m(\mathbf{x}_i) = \langle D_i, Z_m \rangle$

$$\tilde{w}_m = \underset{w \geq 0}{\arg\min} \sum_{i=1}^{n} L(y_i, f_{\hat{W}_{m-1} + wZ_m}(\mathbf{x}_i)). \tag{2.13}$$

$\tilde{w}_m$ in (2.13) is obtained by solving

$$\frac{\partial L}{\partial \omega} = -\sum_{i=1}^{n} r_i^{(m)} g_m(\mathbf{x}_i) \exp(-w y_i g_m(\mathbf{x}_i)) = 0$$

with simple algorithms such as the bisection algorithm [Boyd and Vandenberghe, 2004].Then $\mathbf{w}$ is obtained by normalizing $\mathbf{w} = \frac{\tilde{\mathbf{w}}}{||\tilde{\mathbf{w}}||_2}$.

At the end of the m^{th} step, the weight matrix for the learned Mahalanobis distance metric is updated by

$$\hat{W}_m = \hat{W}_{m-1} + w_m Z_m \tag{2.14}$$

The full algorithm is summarized in Algorithm 2.2 in Section 4.

2.3.2 Sparse Learning and Feature Selection

In the current literature of sparse distance metric learning, a penalty of sparsity is usually imposed on the columns of the weight matrix W or L, which is inefficient in achieving both element-wise sparsity and low rank in the resulting W. For instance, Sparse Metric Learning via Linear Programming (SMLlp) [Rosales and Fung, 2006] is able to obtain a low-rank W but the resulting W is dense, rendering it not applicable to high-dimensional datasets and being lack of feature interpretability. Other methods, such as Sparse Metric Learning via Smooth Optimization (SMLsm) [Ying *et al.*, 2009], cannot preserve the positive semidefiniteness of W while imposing constraints for element-wise sparsity and low rank. These methods often rely on computationally intensive projection to the positive-semidefinite cone in their optimization steps. With the rank-one decomposition of W, we can achieve element-wise sparsity and low rank of the resulted weight matrix simultaneously by regularizing both ξ at each boosting step and the total number of boosting steps M. First, we enforce the element-wise sparsity by penalizing on the l_1 norm of ξ. This measure not only renders a sparse linear transformation of the input space but also performs feature selection and provides a small subset of features relevant to the class difference as output at each step. Then every $Z = \xi\xi^T$ is a sparse rank-one matrix that contributes to a final sparse weight matrix. Here the sparse transformation corresponds to a sparse approximation of the true weight matrix W, by which dimensions with zero weights are implicitly discarded in the space transformation. The positive-semidefiniteness of W is preserved by the non-negative summation of rank-one updates. Specifically, (2.12) is replaced by a penalized minimization problem:

$$\xi_m = \underset{\xi \in \mathbb{R}^p,\ ||\xi||_2 = 1}{\arg\min} \; -\xi^T A_m \xi + \lambda_\xi \sum_{j=1}^{p} |\xi_j| \tag{2.15}$$

where $\lambda_\xi > 0$ is the regularizing parameter on ξ.

As pointed out in Section 3.1, (2.12) can be solved as a eigen-decomposition problem. The optimization problem in (2.15), appended with a single sparsity constraint

on the eigenvector associated with the largest eigenvalue, is shown in [Yuan and Zhang, 2013] as a sparse eigenvalue problem. We adopt a simple yet effective solution that adapts the truncated iterative power method introduced in [Yuan and Zhang, 2013] for obtaining the largest sparse eigenvectors with at most κ nonzero entries. Power methods provide a scalable solution to compute the largest eigenvalue and the corresponding eigenvector of high-dimensional matrices without using the computationally intensive matrix decomposition. The truncated power iteration applies the hard-thresholding shrinkage method on the largest eigenvector of A_m. At each boosting step, we solve the constrained optimization problem in (2.15) using the truncated power method as given in Algorithm 2.

Algorithm 2.1 Truncated power method for solving (2.15) at the m^{th} boosting step

Input: $A_m \in \mathbb{R}^{p_m \times p_m}$, $\kappa \in \{1, 2, \ldots, p_m\}$, and the regularizing parameter $\lambda_0 > 0$

1) Initialization: $A_0 = A_m$, $\xi_0 = \frac{\mathbf{1}_{p_m}}{\sqrt{p_m}}$

2) Iteration: For $t = 1, 2, \ldots,$ repeat until convergence

(a) Update $\lambda_t = 10\lambda_0$ until $A_t = A_{t-1} + \lambda_t I_p$ becomes positive semi-definite.

(b) Compute $\hat{\xi}_t = \frac{A_t \xi_{t-1}}{||A_t \xi_{t-1}||}$.

(c) Let $F_t = \text{supp}(\hat{\xi}_t, \kappa)$ be the indices of $\hat{\xi}_t$ with the largest κ absolute values. Compute $\tilde{\xi}_t = \text{Truncate}(\hat{\xi}_t, F_t)$.

(d) Normalize $\xi_t = \frac{\tilde{\xi}_t}{||\tilde{\xi}_t||}$.

Output: $\xi_m = \xi_t$

It is worth noting that A_m in each step of gradient boosting is not guaranteed to be positive semi-definite. Thus, to ensure that the objective function to be non-decreasing, we add a positive diagonal matrix $\tilde{\lambda} I_p$ to the matrix A for $\tilde{\lambda}$ large enough such that $\tilde{A} = A + \tilde{\lambda} I_p$ is positive semi-definite and symmetric. Such change only adds a constant term to the objective function, which produces a different sequence of iterations, and there is a clear tradeoff. If $\tilde{\lambda}$ dominates A, the objective function becomes approximately a squared norm, and the algorithm tends to terminate in

only a few iterations. In the limiting case of $\tilde{\lambda} \to \infty$, the method will not move away from the initial iterate. To handle this issue, we adapt a stochastic method that gradually increase $\tilde{\lambda}$ during the iterations and we do so only when the monotonicity is violated, as shown in the step 1 of Algorithm 2.1. This truncated power method allows fast computation of the largest κ-sparse eigenvalue. For s high-dimensional but sparse matrix A_m, it also supports sparse matrix computation, which decreases the complexity from $O(p^3)$ to $O(\kappa pT)$, where T is the number of iterations.

Using parameter κ in the sparse eigenvalue problem spares the effort of tuning the regularizing parameter λ_ξ to achieve the desirable level of sparsity. Under the context of *sDist*, κ indeed controls the level of *module effect* among input variables, namely, the joint effect of selected variables on the class membership. Inputs that are marginally insignificant can have substantial influence when joined with others. The very nature of the truncated iterative power method enables us to identify informative variables in groups within each step. These variables are very likely to constitute influential interaction terms that explain the underlying structure of decision boundary which are hard to discern marginally. This characteristic is deliberately utilized in the construction of nonlinear feature mapping adaptively, which will be discussed in detail in Section 2.4. In practice, the value of κ can be chosen based on domain knowledge, depending on the order of potential interactions among variables in the application. Otherwise, we use cross-validation to select the ratio between κ and the number of features p, denoted as ρ, at each boosting step as it is often assumed that the number of significant features is relatively proportional to the total number of features in real applications.

2.3.3 Sparse Boosting

The number of boosting steps M, or equivalently the number of rank-one matrices, bounds the overall sparsity and the rank of resulted weight matrix. Without controlling over M from infinitely large, the resulted metric may fail to capture the

low-dimensional informative representation of the input variable space. Fitting with infinitely many weak learners without regularization will produce an over-complicated model that causes over-fitting and poor generalization performance. Hence, in addition to sparsity control over ξ, we incorporate an automatic selection of the number of weak learners M into the boosting algorithm by formulating it as an optimization problem. This optimization impose a further regularization on the weight matrix W to enforce a low-rank structure. Therefore, the resulting W is ensured to have reduced rank if the true signal lies in a low dimensional subspace as well as guaranteeing the overall element-wise sparsity.

To introduce the sparse boosting for choosing an M, we first rewrite the output discriminant function as a hat operator Υ_m, mapping the feature space to the reduced and transformed space, i.e., $\Upsilon_m : X \to \tilde{X}_m$, in which $\tilde{X}_m$ is the transformed space by $\hat{L}_m$, $\hat{L}_m \hat{L}_m^T = \hat{W}_m$. Therefore, we have

$$f_{\hat{W}_m}(x) = \langle D, \hat{W}_m \rangle = f(\Upsilon_m(X)).$$

Here Υ_m is uniquely defined by the positive semi-definiteness of $\hat{W}_m$. Hence, we define the complexity measure of the boosting process at the m^{th} step by the generalized definition of degrees of freedom in [Green *et al.*, 1994]:

$$C_m = tr(\Upsilon_m) = tr(\hat{L}_m). \tag{2.16}$$

With the complexity measure in (2.16), we adopt the sparse boosting strategy introduced in [Bühlmann and Yu, 2006]. First, let the process carry on for a large number, M, of iterations; then the optimal stopping time $\hat{m}$ is the minimizer of the stopping criterion

$$\hat{m} = \underset{1 \le m \le M}{\arg\min} \left\{ \sum_{i=1}^{N} L(y_i, f_{\hat{W}_m}(\mathbf{x}_i)) \right\} + \lambda_C C_m \tag{2.17}$$

where $\lambda_C > 0$ is the regularizing parameter for the overall complexity of W.

This objective is rather intuitive: ξ_m's are learned as sparse vectors and thus $Z_m = \xi_m \otimes \xi_m$ has nonzero entries mostly on the diagonal at variables selected in ξ_m.

Therefore, $tr(\hat{L_m})$ is a good approximation of the number of selected variables that explicitly indicates the level of complexity of the transformed space at step m.

2.4 Boosted Nonlinear Sparse Metric Learning

The classifier defined in (2.3) only works when the signal of class membership is inherited within a linear transformation of the original input variable space, which is rarely the case in practice. In this section, we introduce nonlinearity via feature mapping of $\mathcal{X}$ so that the Mahalanobis distance metric is learned on a nonlinearly transformed feature space and then incorporated with the classifier in (2.3). That is, we map the original input data $\mathbf{x} \in \mathcal{X} \subseteq \mathcal{R}^p$ to a nonlinear feature vector $\phi(\mathbf{x}) \in \mathcal{R}^{\tilde{p}}$, where $\tilde{p} \geq p$, and then learn a linear transformation defined by W in $\mathbb{R}^{\tilde{p}}$ corresponding to a nonlinear transformation in $\mathbb{R}^p$. The new discriminant function brings is then defined as

$$f_W^{\phi}(\mathbf{x}_i) = \langle D_i^{\phi}, W_m \rangle \tag{2.18}$$

where

$$\begin{aligned} D_i^{\phi} &= \tfrac{1}{k} \textstyle\sum_{j \in S_k^-(\mathbf{x}_i)} [\phi^{(m)}(\mathbf{x}_i) - \phi^{(m)}(\mathbf{x}_j)][\phi^{(m)}(\mathbf{x}_i) - \phi^{(m)}(\mathbf{x}_j)]^T \\ &\quad -\tfrac{1}{k} \textstyle\sum_{j \in S_k^+(\mathbf{x}_i)} [\phi^{(m)}(\mathbf{x}_i) - \phi^{(m)}(\mathbf{x}_j)][\phi^{(m)}(\mathbf{x}_i) - \phi^{(m)}(\mathbf{x}_j)]^T \end{aligned} \tag{2.19}$$

Learning a "good" feature mapping in the entire infinite nonlinear feature space is infeasible. In [Torresani and Lee, 2006], Torresani and Lee resort to the "kernel" trick and construct the Mahalanobis distance metric on the basis expansion of kernel functions in RKHS. Taking a different route, Kedem *et al* [Kedem *et al.*, 2012] abort the reliance on the Mahalanobis distance metric and learn a distance metric on the non-linear basis functions constructed by regression trees. Although these methods provide easy-to-use "black box" algorithms that offers extensive flexibility in modeling a nonlinear manifold, they are sensitive to the choices of model parameters and are subject to the risk of overfitting. The superfluous set of basis functions also hinders

the interpretability of the resulting metric model with respect to the relevant factors of class separation.

In this chapter, we restrict the feature mapping $\phi(\mathbf{x})$ to the space of polynomial functions of the original input variables $x_1, \ldots, x_p$. The construction of nonlinear features is tightly incorporated in the boosted metric learning algorithm introduced in Section 3. Accordingly, a proper metric is learned in concert with the building of essential nonlinear mappings suggested in the data.

Our choice of the polynomial feature mapping enables a straightforward incorporation into the adaptive learning algorithm. We initialized $\phi(\mathbf{x}) = (x_1, x_2, \ldots, x_p)^T$ as the identity mapping at step 0. In the subsequent steps, based on solution the regularized optimization problem (2.15), we expand the feature space by including interaction terms and polynomial terms among the selected variables. This allows the boosting algorithm to benefit from the consideration of interactions without running into overwhelming computational burden and storage need. The full polynomial expansion will result in $(2^p)^2$ dimensions for a single D_i^{ϕ}.

The polynomial feature mapping also permits selection of significant nonlinear features. Kernel methods are often preferred in nonlinear classification problems due to its flexible infinite-dimensional basis functions. However, for the purpose of achieving sparsity in the weight matrix, each basis function need to be evaluated for making the selection toward a sparse solution. Hence, using kernel methods in such a case is computationally infeasible due to its infinite dimensionality of basis functions. With adaptively expanding polynomial features, optimizing (2.15) on the expanded feature space is able to identify not only significant input variables but also informative interaction terms and polynomial terms.

Before we layout the details of the adaptive feature expansion algorithm, we define the following notions: Let $\mathcal{C}_m = \{\tilde{x}_1, \ldots, \tilde{x}_{p_m}\}$ be the set of candidate variables at step m, where $\tilde{x}$ represents the candidate feature, and $\tilde{p}_m$ is the cardinality of the set $\mathcal{C}_m$, that is, the number of features at step m. The set $\mathcal{C}_m$ includes the entire set

of original variables as well as added interaction terms that we consider in the next step. Denote $\mathcal{S}_m$ as the cumulative set of unique variables selected up to step m, and $\mathcal{A}_m$ be the set of variables newly selected in step m. Then,

Step 0 : Set $\mathcal{C}_0 = \{\tilde{x}_1 = x_1, \ldots, \tilde{x}_p = x_p\}$, the set of the original variables.

Step 1 : Select $\mathcal{A}_1 \subset \mathcal{C}_0$ by the regularized optimization in (2.15) with prespecifed $|\mathcal{A}_1| = \kappa$.

$$\text{Set} \quad \mathcal{S}_1 = \mathcal{A}_1; \quad \mathcal{C}_1 = \mathcal{C}_0 \cup (\mathcal{S}_1 \otimes \mathcal{A}_1)$$

where the operator "$\otimes$" is defined as

$$\mathcal{S}_1 \otimes \mathcal{A}_1 = \{\tilde{x}_i \tilde{x}_j : \tilde{x}_i \in \mathcal{S}_1, \tilde{x}_j \in \mathcal{A}_1\}$$

Step m , $m = 2, \ldots, M$: Select $\mathcal{A}_m \subset \mathcal{C}_{m-1}$. Then

$$\mathcal{S}_m = \mathcal{S}_{m-1} \cup \mathcal{A}_m, \quad \mathcal{C}_m = \mathcal{C}_{m-1} \cup (\mathcal{S}_m \otimes \mathcal{A}_m) \tag{2.20}$$

Then $\phi(\mathbf{x})$ at the m^{th} step of the algorithm is defined as $\phi^{(m)}(\mathbf{x}) \triangleq X_{\mathcal{C}_{m-1}}$, the vector[2] whose components are elements in $\mathcal{C}_{m-1}$

It is worth noting that, in updating $D_i^{\phi^{(m)}}$, there is no need to compute the entire matrix, the cost of which is on the order of np_m^3. Instead, taking advantage of the existing $D_i^{\phi^{(m-1)}}$, it is only required to add $\delta_m \triangleq (p_m - p_{m-1})$ rows of pairwise products between the newly added terms and currently selected ones and make the resulting matrix symmetric. The extra computational cost is reduced to $O(n\delta_m^3)$ and $\delta_m \ll p_m$ when p is large. Therefore, the method of expanding the feature space in the stepwise manner is tractable even when p is large. Since we only increase the dimension of feature space by a degree less than $\frac{1}{2}(\delta_m \kappa + \kappa)$ at each step and M is controlled by the sparse boosting, the proposed hierarchical expansion is computationally feasible even with high-dimensional input data.

[2]Here $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T$. When $\mathcal{C}$ is a set of variable or interactions of variables, $X_{\mathcal{C}}$ represents the columns of X (or products of columns of X) listed in $\mathcal{C}$.

We integrate the adaptive feature expansion for nonlinear metric learning into the boosted sparse metric learning algorithm in Section 3. The final algorithm is summarized in Algorithm 2.2. The details of how to choose the value of parameters κ, λ_C and M are elaborated in Section 5.

Algorithm 2.2 *sDist*: Boosted Nonlinear Sparse Metric Learning

Input Parameters: κ, M, and λ_C

1) Initialization: $\hat{W}_0 = I_{p\times p}$; $\mathcal{C}_0 = \{\tilde{x}_1 = x_1, \ldots, \tilde{x}_p = x_p\}$; residuals $r_i^{(0)} = y_i$, $i = 1, 2, \ldots, n$.

2) For $m = 1$ to M:

(a) Define the nonlinear feature mapping $\phi^{(m)}(\mathbf{x}) = X_{\mathcal{C}_{m-1}}$; Update $D_i^{\phi^{(m)}}$ according to (2.19)

(b) $A_m = \sum_{i=1}^{n} r_i^{(m)} D_i^{\phi^{(m)}}$.

(c) Get ξ_m from the regularized minimization problem:

$$\xi_m = \underset{\xi\in\mathbb{R}^{p_m}, ||\xi||_2=1}{\arg\min} \; -\xi^T A_m \xi + \lambda_\xi \sum_{j=1}^{p_m} |\xi_j|, \tag{2.21}$$

by the truncated iterative power method (Algorithm 2.1) with corresponding κ.

(d) Based on the sparse solution of ξ_m,update $\mathcal{A}_m, \mathcal{S}_m$ and $\mathcal{C}_m$. $g_m(\mathbf{x}_i) = \xi_m^T D_i^{\phi^{(m)}} \xi_m$ for $i = 1, 2, \ldots, n$.

(e) Get w_m from (2.13) by the bisection algorithm.

(f) Compute residuals $r_i^{(m)}$ based on (2.9):

$$r_i^{(m+1)} = r_i^{(m)} \exp(-y_i \omega_m g_m(\mathbf{x}_i)), \text{for } i = 1, \ldots, n.$$

(g) Update the weight matrix:

$$\hat{W}_m = \mathcal{I}_m^T \hat{W}_{m-1} \mathcal{I}_m + w_m \xi_m \xi_m^T$$

where $\mathcal{I}_m = (I_{p_{m-1}\times p_{m-1}}, \mathbf{0}_{p_{m-1}\times p_m - p_{m-1}})$, where $I_{p\times p}$ is the p by p identity matrix and $\mathbf{0}_{p\times q}$ is the zero matrix of dimension p by q.

3) Determine the optimal stopping time by solving

$$\hat{m} = \underset{1\le m\le M}{\arg\min} \sum_{i=1}^{N} L(y_i, \hat{W}_m) + \lambda_C C_m.$$

Then set the output $\hat{W} = \hat{W}_{\hat{m}}$.

2.5 Related Works

There is an extensive literature devoted on the problem of learning a proper W for the Mahalanobis distance. In this chapter, we focus on the problem of supervised metric learning for classification in which class labels are given in the training sample. In the following, we categorize current related methods in the literature into four groups: 1) global metric learning, 2) local metric learning, 3) sparse metric learning, and 4) nonlinear metric learning.

Global metric learning aims to learn a W that addresses the similarity and dissimilarity objectives at all sample points. Probability Global Distance Metric (PGDM) learning [Xing *et al.*, 2002] is an early representative method of this group. In PGDM, the class label (y) is converted into pairwise constraints on the metric values between pairs of data points in the feature ($\mathbf{x}$) space: equivalence (similarity) constraints that similar pairs (in y) should be close (in $\mathbf{x}$) by the learned metric; and in-equivalence (dissimilarity) constraints that dissimilar ones (in y) should be far away (in $\mathbf{x}$). The distance metric is then derived to minimize the sum of squared distances between data points with the equivalence constraints, while maintaining a lower bound for the ones with the in-equivalence constraints. The global optimum for this convex optimization problem is derived using Semi-Definite Programming (SDP). However, the standard SDP by the interior point method requires $O(p^4)$ storage and has a worst-case computational complexity of approximately $O(p^{6.5})$, rendering it computationally prohibitive for large p. Flexible Metric Nearest Neighbor (FMNN) [Friedman, 1994] is another method of this group, which, instead, adapts a probability framework for learning a distance metric with global optimality. It assumes a logistic regression model in estimating the probability for pairs of observations being similar or dissimilar based on the learned metric, yet suffering poor scalability as well.

The second group of methods, local metric learning methods, learn W by pursuing similarity objective within the local neighborhoods of observations and a large margin at the boundaries between different classes. For examples, see the Neighborhood

Component Analysis (NCA) [Goldberger *et al.*, 2005] and the Large Margin Nearest Neighbor (LMNN) [Blitzer *et al.*, 2005]. NCA learns a distance metric by stochastically maximizing the probability of correct class-assignment in the space transformed by L. The probability is estimated locally by the Leave-One-Out (LOO) kernel density estimation with a distance-based kernel. LMNN, on the other hand, learns W deterministically by maximizing the margin at class boundary in local neighborhoods. Adapting the idea of PGDM while focusing on local structures, it penalizes on small margins in distance from the query point to its similar neighbors using a hinge loss. It has been shown in [Blitzer *et al.*, 2005] that LMNN delivers the state-of-the-art performance among most distance metric learning algorithms. Despite its good performance, LMNN and its extensions suffers from high computational cost due to their reliance on SDP similar to PGDM. Therefore, they always require data pre-processing for dimension reduction, using *ad-hoc* tools, such as the Principal Component Analysis (PCA), when applied to high-dimensional data.

When the data dimension increases, learning a full distance metric becomes extremely computationally expensive and may easily run into over-fitting on high-dimensional data contaminated with noises. It is expected that a sparse distance matrix would produce a better generalization performance than its dense counterparts and afford a much faster and efficient distance calculation. Sparse metric learning is motivated by the demand of learning appropriate distance measures in high-dimensional space and can also lead to supervised dimension reduction. In the sparse metric learning literature, sparsity regularization can be introduced in three different ways: on the rank of W for learning a low-rank W, (e.g., [Torresani and Lee, 2006], [Hong *et al.*, 2011], [Rosales and Fung, 2006], [Liu *et al.*, 2010]), on the elements of W for learning an element-wise sparse W [Qi *et al.*, 2009], and the combination of the two [Ying *et al.*, 2009]. All these current strategies suffer from various limitations and computational challenges. First, a low-rank W is not necessarily sparse, as it may still involve all input variables. Methods such as [Rosales and Fung, 2006] impose

penalty on the trace norm of W as the proxy of the non-convex non-differentiable rank function, which usually involves heavy computation and approximation in maintaining both the status of low rank and the positive semi-definiteness of W. Searching for an element-wise sparse solution as in [Qi *et al.*, 2009] places the l_1 penalty on the off-diagonal elements of W. Again, the PSD of the resulting sparse W is hard to maintain in a computationally efficient way. Based on the framework of LMNN, Ying *et al.* [Ying *et al.*, 2009] combine the first two strategies and penalize on the $l_{(2,1)}$ norm[3] of W to regularize the number of non-zero columns in W. Huang *et al.* [Huang *et al.*, 2009] proposed a general framework of sparse metric learning. It adapts several well recognized sparse metric learning methods with a common form of sparsity regularization $tr(SW)$, where S varies among methods serving different purposes. As a limitation of the regularization, it is hard to impose further constraint on S to guarantee PSD in the learned metric.

As suggested in (2.2), the Mahalanobis distance metric implies a linear transformation of the original feature space. This linearity inherently limits the applicability of distance metric learning in discovering the potentially nonlinear decision boundaries. It is also common that some variables are relevant to the learning task only through interactions with others. As a result, linear metric learning is at the risk of ignoring useful information carried by the features beyond the marginal distributional differences between classes. Nonlinear metric learning identifies a Mahalanobis distance metric on a nonlinear mappings of the input variables, introducing nonlinearity via well-designed basis functions on which the distances are computed. Large Margin Component Analysis (LMCA)[Torresani and Lee, 2006] maps the input variables onto a high-dimensional feature space $\mathcal{F}$ by a nonlinear map $\phi : \mathcal{X} \to \mathcal{F}$, which is restricted to the eigen-functions of a Reproducing Kernel Hilbert Space (RKHS) [Aronszajn, 1950]. Then the learning objective is carried out using the "kernel trick" without explicitly compute the inner product. LMCA involves optimizing over a

[3]The $l_{(2,1)}$ norm of W is given by: $||W||_{(2,1)} = \sum_{h=1}^{p} (\sum_{k=1}^{p} W_{hk}^2)^{\frac{1}{2}}$ [Ying *et al.*, 2009]

non-convex objective function and is slow in convergence. Such heavy computation limits its scalability to relatively large datasets. Kedem *et al.* [Kedem *et al.*, 2012] introduce two methods for nonlinear metric learning, both of which derived from extending LMNN. χ^2-LMNN uses a nonlinear χ^2-distances for learning a distance metric for histogram data. The other method, GB-LMNN, exploits the gradient boosting algorithm that learns regression trees as the nonlinear basis functions. GB-LMNN relies on the Euclidean distance in the nonlinearly expanded features space without an explicit weight matrix W. This limits the interpretability of its results. Current methods in nonlinear metric learning are mostly based on black-box algorithms which are prone to over-fitting and have limited interpretability.

2.6 Practical Remarks

When implementing Algorithm 2.2 in practice, the performance of the *sDist* algorithm can be further improved in terms of both accuracy and computational efficiency by a few practical techniques, including local neighborhood updates, shrinkage, bagging and feature sub-sampling. We numerically evaluate the effect of the following parameters on a synthetic dataset in Section 6.

As stated in Section 3, the base classifier $f_{W,k}(x_i)$ in (2.3) is constructed based on local neighborhoods. Without additional domain knowledge about the local similarity structure, we search for local neighbors of each sample point using the Euclidean distance. While the actual neighbors found in the truly informative feature subspace may not be well approximated by the neighbors found in the Euclidean space of all features, the learned distance metrics in the process of the boosting algorithm can be used to construct a better approximation of the true local neighborhoods.The revised local neighborhoods are helpful in preventing the learned metric from overfitting to the neighborhoods found in the Euclidean distance and thus reduce overfitting to the training samples. In practice, we update local neighborhoods using the learned

metric at different steps of the booting algorithm. The frequency of the local neighborhood updates is determined by the trade-off between the predictive accuracy and the computational cost for re-computing distances between pairs of sample points. The actual value of updating frequency varies in real data applications and can be tuned by cross-validation.

Our solution is to update local neighborhoods using the learned metric during the boosting algorithm. Since the metric updated at each boosting step is trained to approximate the true similarity relationship, they can be used to construct a more accurate distance function for searching for the true local neighbors. The revised local neighborhoods are helpful in preventing the learned metric from overfitting to the neighborhoods found in the Euclidean distance and thus reduce overfitting to the training samples. In practice, the frequency of the local neighborhood updates is determined by the trade-off between the predictive accuracy and the computational cost for re-computing distances between pairs of sample points. The actual value of updating frequency varies in real data applications and can be tuned by cross-validation.

In addition to the sparse boosting in which the number of boosting steps is controlled, we can further regularize the learning process by imposing a shrinkage on the rank-one update at each boosting step. The contribution of Z_m is scaled by a factor $0 < \nu \leq 1$ when it is added to the current weight matrix W_{m-1}. That is, step 2g in Algorithm 2.2 is replaced by

$$\hat{W}_m = \mathcal{I}_m^T \hat{W}_{m-1} \mathcal{I}_m + \nu w_m \xi_m \xi_m^T. \tag{2.22}$$

The parameter ν can be regarded as controlling the learning rate of the boosting procedure. Such a shrinkage helps circumventing the case that individual rank-one updates of the weight matrix fit too closely to the training samples. It has been empirically shown that smaller values of ν favor better generalization performance and require correspondingly larger values of M [Friedman, 2001]. In practice, we use cross-validation to determine the value of ν.

Bootstrap **Ag**gregat**ing** (bagging) has been demonstrated to improve the performance of a noisy classifier by averaging over weakly correlated classifiers [Hastie *et al.*, 2009]. Correlations between classifiers are weakened by random subsampling. In our gradient boosting algorithm, we use the same technique of randomly sampling a fraction η[4], $0 < \eta \leq 1$, of the training observations to build each weak learner for learning the rank-one update. This idea has been well exploited in [Friedman, 2002] with tree classifiers, and it is shown that both accuracy and execution speed of the gradient boosting can be substantially improved by incorporating randomization into the procedure. The value of η is usually taken to be 0.5 or smaller if the sample size is large, which is tuned by cross-validation in our numerical experiments. In particular to our algorithm, bagging substantially reduces the training set size for individual rank-one updates so that D_i can be computed on the fly much more quickly without being pre-calculated, avoiding the need of computational memory. As a result, in applications with large sample sizes, bagging not only benefits the test error but also improves better computational efficiency.

In high-dimensional applications, it is likely that the input variables are correlated, which translates to high variance in the estimation. As *sDist* can be viewed as learning an ensemble of nonlinear classifiers, high correlation among features can deteriorate the performance of the aggregated classifier. To resolve this, we employ the same idea as in random forests [Breiman, 2001] of random subsampling on features to reduce the correlation among weak learners without greatly increasing the variance. At each boosting step m, we randomly select a subset of features of size $\tilde{p}_m$ from the candidate set C_m, where $\kappa < \tilde{p}_m \leq p_m$, on which D_i's is constructed with dimension $\tilde{p}_m \times \tilde{p}_m$. The optimization in (2.15) is then executed on a much smaller scale and select κ significant features from the random subset. As with bagging, feature subsampling enables fast computation of D_i's without pre-calculation. As suggested in [Breiman, 2001], we use $\tilde{p}_m = \sqrt{p_m}$ at the m^{th} boosting step. Although feature subsampling

[4]The parameter η is referred as the "bagging fraction" in the following.

will reduce the chance of selecting out the significant features at each boosting step, it shall be emphasized that bagging on training samples and feature subsampling should be accompanied by shrinkage and thus more boosting steps correspondingly. It is shown in [Hastie *et al.*, 2009] that subsampling without shrinkage leads to poor performance in test samples. With sufficient number of boosting steps, the algorithm manages to select out significant features from a dominant number of irrelevant ones. Since the computational complexity of the proposed algorithm is linearly scalable in the number of boosting steps M while quadratic in the feature dimension p, feature subsampling is more computationally efficient even with large M. Hence, in high-dimensional setting, reducing the dimension of feature set to $\sqrt{p}$ makes the algorithm sufficiently faster.

However, there is no closed rule for choosing the value of M in advance. Since each application has different underlying structure of its significant feature subspace as well as involving with different level of noise, the actual value of M varies case by case. However in general, we suggest a large number of M, from 500 to 2000, that is proportional the number of features p. When feature subsampling is applied, M should be increase in an order of $\sqrt{p}$ to cover all significant features in the random subsampling. Since the sparse boosting process is implemented, overfitting is effectively controlled even with large M and thus it is recommended to start with considerably large value of M. Otherwise, we use cross-validation to evaluate different choices of M's.

2.7 Numerical Experiments

In this section, we present both simulation studies and real-data applications to illustrate the proposed *sDist* algorithm. The algorithm is implemented with the following specifications. We use 5-fold cross-validations to determine the degree of sparsity for each rank-one update ρ, choosing from candidate values $\{0.05, 0.1, 0.2\}$. The same

cross-validation is also applied to the tune overall complexity regularizing parameter $\lambda_C \in \{0.001, 0.01, 0.1, 1, 10\}$. In order to control the computation cost and to ensure interpretability of the selected variables and polynomial features, we impose an upper limit on the maximum order of polynomial of the expanded features. That is, when the polynomial has an order greater than a cap value, we stop adding it to the candidate feature set. For our experiments, the cap order is set to be 4. Namely, we expect to see maximally four-way interactions. The total number of boosting steps M is set to be 2000 for all simulation experiments. While by sparse boosting, the actual numbers of weak learners used vary from case to case. Throughout the numerical experiments, the reported test errors are estimated using the k-Nearest Neighbor classifier with $k = 3$ under the tuned parameter configuration.

The performance of *sDist* is compared with several other distance metric learning methods, with the k-Nearest Neighbor (kNN) representing the baseline method with no metric learning, Probability Global Distance Metric (PGDM)[Xing *et al.*, 2002], Large Margin Nearest Neighbor (LMNN) [Blitzer *et al.*, 2005], Sparse Metric Learning via Linear Programming (SMLlp) [Rosales and Fung, 2006], and Sparse Metric Learning via Smooth Optimization (SMLsm) [Ying *et al.*, 2009]. PGDM [5] [Xing *et al.*, 2002] is a global distance metric learning method that solves the optimization problem:

$$\begin{aligned} \min_{W \succeq 0} \quad & \sum_{y_i = y_j} (\mathbf{x}_i - \mathbf{x}_j)^T W (\mathbf{x}_i - \mathbf{x}_j) \\ \text{s.t.} \quad & \sum_{y_i \neq y_l} (\mathbf{x}_i - \mathbf{x}_l)^T W (\mathbf{x}_i - \mathbf{x}_l) \geq 1. \end{aligned}$$

LMNN [6] learns the weight matrix W by maximizing the margin between classes in local neighborhoods with a semi-definite programming. That is, W is obtained by

[5]Source of Matlab codes: `http://www.cs.cmu.edu/%7Eepxing/papers/Old_papers/code_Metric_online.tar.gz`

[6]Source of Matlab codes: `http://www.cse.wustl.edu/~kilian/code/code.html`

solving:

$$\begin{aligned}\min_{W\succeq 0,\ \xi_{ijl}\geq 0} \quad & (1-\mu)\sum_{i=1}^{n}\sum_{j\in\mathcal{S}_k^+(\mathbf{x}_i)}(\mathbf{x}_i-\mathbf{x}_j)^T W(\mathbf{x}_i-\mathbf{x}_j)+\mu\sum_{i=1}^{n}\sum_{j\in\mathcal{S}_k^+(\mathbf{x}_i)}\sum_{l\in\tilde{\mathcal{S}}^-(\mathbf{x}_i)}\xi_{ijl},\\ \text{s.t.} \quad & (\mathbf{x}_i-\mathbf{x}_l)^T W(\mathbf{x}_i-\mathbf{x}_l)-(\mathbf{x}_i-\mathbf{x}_j)^T W(\mathbf{x}_i-\mathbf{x}_j)\geq 1-\xi_{ijl},\end{aligned}$$

where ξ_{ijl}'s are slack variables and $\tilde{\mathcal{S}}^-(\mathbf{x}_i)\triangleq\{l|y_l\neq y_i \text{ and } d_I(\mathbf{x}_i,\mathbf{x}_l)\leq \max_{j\in\mathcal{S}_k^+(\mathbf{x}_i)} d_I(\mathbf{x}_i,\mathbf{x}_j)\}$. In the experiments, we use $\mu=0.5$ as suggested in [Blitzer *et al.*, 2005]. SMLlp aims at learning a low rank weight matrix W by optimizing over the linear projection $L\in\mathbb{R}^{p\times D}$ with $D\leq p$ in (2.2):

$$\begin{aligned}\min_{L\in\mathbb{R}^{p\times D},\ \xi_{ijl}\geq 0} \quad & \sum_{(i,j,l)\in\mathcal{T}}\xi_{ijl}+\mu\sum_{r=1}^{p}\sum_{s=1}^{D}|L_{rs}|,\\ \text{s.t.} \quad & \|L\mathbf{x}_i-L\mathbf{x}_j\|_2^2\leq\|L\mathbf{x}_i-L\mathbf{x}_l\|_2^2+\xi_{ijl},\quad \forall\ (i,j,l)\in\mathcal{T},\end{aligned}$$

where $\mathcal{T}\in\{(i,j,l)\mid j=\mathcal{S}_1^+(\mathbf{x}_i),\ l=\mathcal{S}_1^-(\mathbf{x}_i)\}$. In a similar manner, SMLsm[7] learns a low-rank weight matrix W by employing a $l_{(2,1)}$ norm on the weight matrix W to enforce column-wise sparsity. It is cast into the minimization problem:

$$\begin{aligned}\min_{U\in\mathcal{O}^p}\min_{W\succeq 0,\ \xi_{ijl}\geq 0} \quad & \sum_{(i,j,l)\in\mathcal{T}}\xi_{ijl}+\mu\sum_{r=1}^{p}\left(\sum_{s=1}^{D}W_{rs}^2\right)^{\frac{1}{2}},\\ \text{s.t.} \quad & 1+(\mathbf{x}_i-\mathbf{x}_j)^T U^T W U(\mathbf{x}_i-\mathbf{x}_j)\leq(\mathbf{x}_i-\mathbf{x}_l)^T U^T W U(\mathbf{x}_i-\mathbf{x}_l)+\xi_{ijl},\\ & \forall(i,j,l)\in\mathcal{T},\end{aligned}$$

where $\mathcal{O}^p$ is the set of $p-$dimensional orthonormal matrices.

The effectiveness of distance metric learning in high-dimensional datasets heavily depends on the computational complexity of the learning method. PGDM deploys a semi-definite programming in the optimization for W which is in the order of $O(p^2+p^3+n^2p^2)$ for each gradient update. LMNN requires a computation complexity of $O(p^4)$ for optimization. SMLsm converges in $O(p^3/\epsilon)$, where ϵ is the stopping criterion for convergence. In comparison, *sDist* runs with a computational complexity

[7]Source of Matlab codes: `http://www.albany.edu/~yy298919/software.html`

of approximately $O(M[(\kappa p + p)\kappa \log p + np^2])$ where M is the number of boosting iterations and κ is the number of nonzero entries in rank-one updates. In practice, *sDist* can be significantly accelerated by applying the modifications in the Section 5, in which p is substituted by $\tilde{p}$ and n is substituted by ηn.

We construct two simulations settings that are commonly used as classical examples for nonlinear classification problems in the literature, the "double ring" case and the "XOR" case. In Figure 2, the left most column of the figures indicates the contour plots of class probability for generating sample points in a 3-dimensional surface, whereas the input variable space is expanded to a much greater space of $p = 50$, where irrelevant input variables represent pure noises. Figure 2.3 (*top row*) shows a simulation study in which sample points with opposite class labels interwine in a double rings, and Figure 2.3 (*bottom row*) borrows the illustrative example of "XOR" classification in Section 2. The columns 2-4 in Figure 2.3 illustrate the transformed subspaces learned in *sDist* algorithm at selected iterations. Since the optimal number of iterations is not static and due to the space limit, we show only the first iteration, the last iteration determined by sparse boosting, and the middle iteration, which is rounded half of the optimal number of iterations. It is clearly shown in Figure 2.3 that the surfaces transformed by the learned distance metric correctly capture the structures of the generative distributions. In the "double ring" example (*top row*), the learned surface sinks in the center of the plane while the rim bends upward so that sample points in the "outer ring" are drawn closer in the transformed surface. The particular shape owes to the quadratic polynomial of the two informative variables chosen in constructing W, shown as the parabola in cross-sectional grid lines. In the "XOR" example (*bottom row*), the diagonal corners are curved toward the same directions. The interaction between the two informative variables is selected in additional to original input variable, which is essential in describing this particular crossing nonlinear decision boundary. *sDist* also proves highly computationally efficient, achieving approximate optimality within a few iterations.

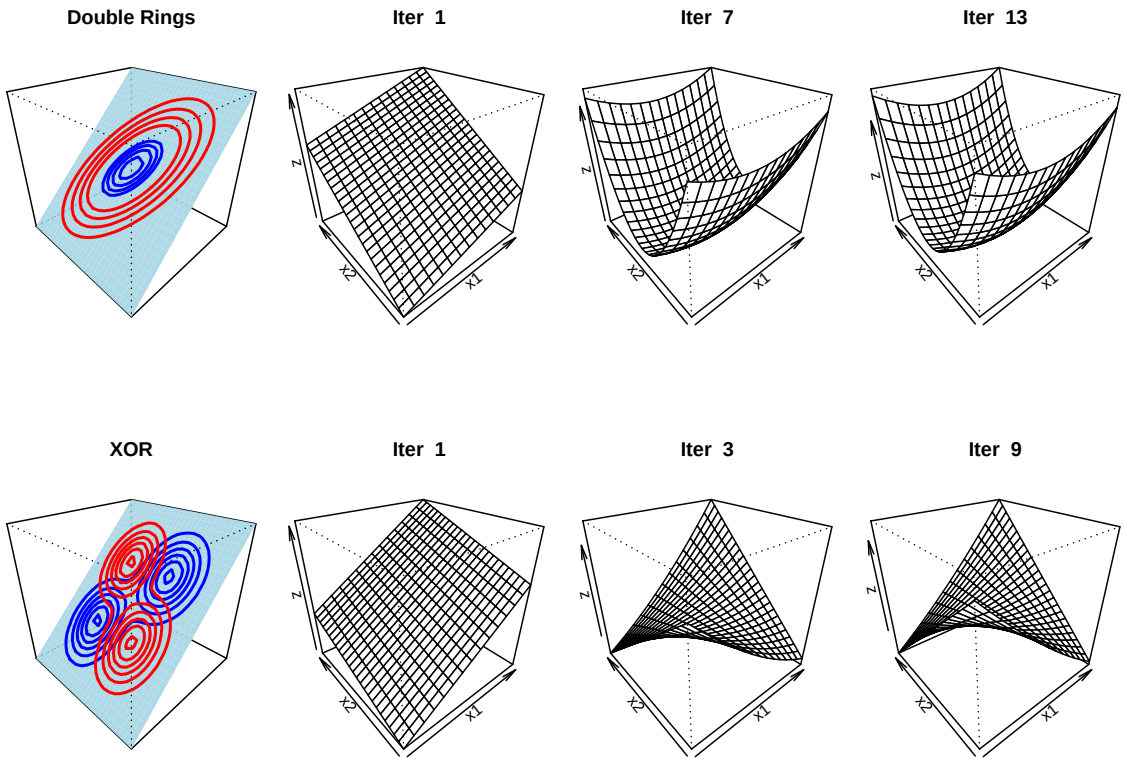


Figure 2.3: Transformed subspaces corresponding to metrics learned for nonlinear binary classification problems. The first column shows the simulations setups. *Upper*: Sample points are drawn from a "double rings" distribution. Shown are the contour plot of the generative class probability on a 3-dimensional surface. *Lower*: Sample points are drawn from the classical XOR scenario. Columns 2-4 demonstrate how the metric learning algorithm transofrm the feature space at selected iterations. The vertical dimensions is computed as the first principle components of the transformed feature space $L\phi(\mathbf{x})$, where $LL^T = W$. Note that the solid lines in the contour plots only show the geodesic lines of high probabilities in the class generation probability distributions. The generated class labels are not separable.

We also compare the performance of *sDist* with other metric learning methods under different values of dimensions p and sample sizes N to demonstrate its scal-

"Double Ring" Scenario									
	N=100			N=500			N=5000		
p	50	500	1000	50	500	1000	50	500	1000
kNN	0.310	0.40	0.488	0.311	0.426	0.478	0.308	0.475	0.489
PGDM	0.320	0.355	0.389	0.312	0.356	0.377	0.337	0.340	0.412
LMNN	0.230	0.280	0.290	0.245	0.291	0.289	0.246	0.303	0.315
SMLsm	0.222	0.289	0.250	**0.169**	0.200	0.249	0.199	0.276	0.330
sDist	**0.143**	**0.189**	**0.192**	0.177	**0.183**	**0.191**	**0.168**	**0.179**	**0.202**
Bayes Rate	0.130	0.150	0.160	0.154	0.156	0.144	0.160	0.154	0.156

Table 2.1: Comparison of distance metric learning methods in the simulated scenario of "Double Rings" as illustrated in Figure 2.3 (*Upper* panel). Recorded are average test error over 20 simulations with varying sample size (N) and different total number of variables (p). Averaged Bayes rates are also given for reference.

ability and its strength in obtaining essentially sparse solution in high-dimensional datasets. In this case, we generate the sample points from the "double ring " example and the "XOR" example with the numbers of informative variables being 10% of the total dimensions, ranging from 100 to 5000. The results of these two cases are shown in Table 2.1 and Table 2.2 respectively. It is noted that *sDist* achieves relatively low test errors as compared to the competing methods, especially in high dimensional settings. *sDist* is also proved to be scalable to datasets with large sample sizes and with high-dimensional inputs.

The performance of *sDist* is also evaluated on three public datasets, presented in Table 2.3. For each dataset, we randomly split the original data into a 70% training set and a 30% testing set, and repeat for 20 times. Parameter values are tuned by cross-validation similarly as the simulation studies. The reported test errors in Table 2.4 are the averages over 20 random splits on the datasets. The reported running

"XOR" Scenario									
	N=100			N=500			N=5000		
p	50	500	1000	50	500	1000	50	500	1000
kNN	0.355	0.410	0.491	0.420	0.446	0.499	0.397	0.500	0.500
PGDM	0.221	0.355	0.383	0.289	0.356	0.360	0.354	0.350	0.403
LMNN	**0.145**	0.280	0.274	0.188	0.213	0.239	0.198	0.231	0.299
SMLsm	0.207	0.307	0.333	0.277	0.291	0.337	0.242	0.378	0.420
sDist	0.157	**0.199**	**0.192**	**0.169**	**0.183**	**0.225**	**0.193**	**0.187**	**0.221**
Bayes Rate	0.130	0.160	0.160	0.133	0.177	0.181	0.155	0.144	0.138

Table 2.2: Comparison of distance metric learning methods in the simulated scenario of "XOR" as illustrated in Figure 2.3 (*Lower* panel). Average test error is evaluated over 20 simulations with varying sample size (N) and different total number of variables (p). Averaged Bayes rates are also given for reference.

Data Statistics	Ionosphere	SECOM	Madelon
Input Dimension (p)	33	590	500
Training Size (N)	351	1567	2600

Table 2.3: Data Statistics of 3 public real datasets.

times are the average CPU times for one execution[8]. We also obtain the average percentage of features selected by various sparse metric learning methods in Figure 2.4.

We first compare various distance metric learning methods on the Iononsphere

[8]Running time of *sDist* for datasets ionosphere, SECOM, Madelon are based on $M = 100$, 500, and 500 respectively with the configurations that achieve the best predictive performance. The *sDist* algorithm is implemented on R (version 3.1.3) on x86_64 Redhat Linux GNU system. Other competing algorithms are implemented on Matlab ($R2014a$) on the same operating system.

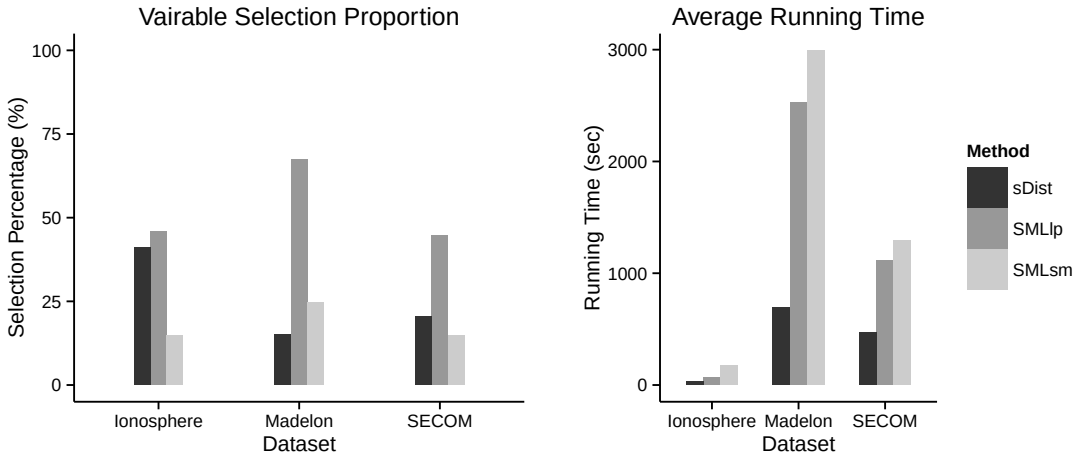


Figure 2.4: The average percentages of variables (features) selected in the final metrics learned by different algorithms as well as the average running times. The percentage of *sDist* is calculated as the ratio of the total number of selected features over p_{m*}, where p_{m*} is the dimension of the candidate set defined in (2.16) at the optimal stopping iteration $m*$ selected by the sparse boosting method in Section 3.3.

Data	Ionosphere		SECOM		Madelon	
Methods	Test Error	Running Time (sec)	Test Error	Running Time (sec)	Test Error	Running Time (sec)
kNN	0.13	0.01	0.14	2.07	0.46	9.05
PGDM	0.07	37.80	0.09	960.47	0.31	2527.82
LMNN	0.06	20.06	0.08	1960.94	0.39	1323.64
SMLsm	0.09	173.19	0.09	1293.97	0.41	2993.97
sDist	**0.05**	27.49	**0.07**	473.07	**0.09**	689.64

Table 2.4: Comparison of distance metric learning methods on three real public datasets. The test errors are computed using k-Nearest Neighbor classifier with $k = 3$ based on the learned metrics from the methods under comparisons averaged over 20 random cross-validations. The recorded running times are the average CPU time for one execution.

dataset [Lichman, 2013] [9]. This radar dataset represents a typical small dataset. It contains mixed data types, which poses a challenge to most of the distance-based classifiers. From Table 2.4, we see that *sDist* and other metric learning methods significantly reduce the test errors by learning a nonlinear transformation of the input space, as compared to the ordinary kNN classifier. *sDist* particularly achieves the best performance by screening out a large proportion of noises. The marginal features selected by different methods are compared in Figure 2.4. Features selected by *sDist* are mostly interactions within a single group of variables, suggesting an interesting underlying structure of the data for better interpretation.

SECOM [Lichman, 2013] [10] contains measurements from sensors for monitoring

[9] Available at `https://archive.ics.uci.edu/ml/datasets/Ionosphere`

[10] The data is available at `https://archive.ics.uci.edu/ml/datasets/SECOM`. The original data is trimmed by taking out variables with constant values and variables with more than 10% of missing values so that the dimension is reduced from 591 to 414. Observations with missing value

the function of a modern semi-conductor manufacturing process. This dataset is a representative real-data application in which not all input variables are equally valuable. The measured signals from the sensors contain irrelevant information and high noise which mask the true information from being discovered. Under such scenario, accurate feature selection methods are proven to be effective in reducing test error significantly as well as identifying the most relevant signals [Lichman, 2013]. As shown in Table 2.4, *sDist* [11] demonstrates dominant performance over the other three methods with an improvement about 33% over the original kNN using the Euclidean distance. As compared to SMLsm, another sparse metric learning method, *sDist* shows much better scalability with a large number of input variables in terms of CPU time.

MADELON is an artificial dataset used in the NIPS 2003 challenge on feature selection [12] [Lichman, 2013] [Guyon *et al.*, 2006][Guyon *et al.*, 2007]. It contains sample points with binary labels that are clustered on the vertices of a five dimensional hypercube, in which these five dimensions constitute 5 informative variables. Fifteen linear combinations of these five variables were added to form a set of 20 (redundant) informative variables while the other 480 variables have no predictive power on class label. In Table 2.4, *sDist* shows excellent performance compared to the other competing methods in terms of both predictive accuracy and computational efficiency. The test error achieved by *sDist* also outperforms states-of-the-art methods beyond the distance metric learning literature on the Madelon dataset [Kursa *et al.*, 2010] [Suarez *et al.*, 2014] [Turki and Roshan, 2014]. *sDist* also attains the sparsest solution

after the trimming on variables are discarded in this experiment, which reduces the sample size to 1436.

[11] Due to the heterogeneity in the input variables, we standardized the input variable matrix before implementing the *sDist* algorithm. In the nonlinear expansions, selected interaction terms are also scaled before being added to the candidate set $\mathcal{C}$.

[12] The data is available at `https://archive.ics.uci.edu/ml/datasets/Madelon`. We use both the train data and the validation data. The 5-fold cross-validation is performed on the combined dataset.

as shown in Figure 2.4, with 15.2% of features selected in the final weight matrix. Its outstanding performance indicates the importance of learning the low-dimensional manifold in high-dimensional data, particularly for the cases with low signal-to-noise ratio.

We also experimented different configurations of the tuning parameters introduced in the algorithm and the practical remarks on the Madelon dataset, including the frequency of local neighborhood updates, bagging fraction η, and the degree of sparsity for rank-one updates ρ. The performances in terms of both training error and validation error are shown in Figure 2.5 for both the kNN classifier and the base classifier f_W defined in (2.3). Particularly, it is evident that updating neighborhood more frequently seems to reduce validation error. The gain in performance diminishes as the frequency increases beyond a certain level. In practice, we suggest updating the local neighborhood every 50 steps as a tradeoff between the accuracy and the computational cost. In this example, the best performances of both classifiers are achieved at the bagging fraction 0.3 or 0.5 when the degree of sparsity ρ is small. While as ρ is large, the errors monotonically decrease as the bagging fraction increases. In practice, we suggest a bagging fraction 0.5 for moderate-size datasets and 0.3 for large datasets. When the true informative subspace is of relatively low-dimensional, as in the case of the Madelon dataset, both training errors and validation errors are reduced with small values of ρ. Sparse rank-one updates benefit the most from the boosting algorithm for progressive learning and prevent overfitting at each single step, while in other cases, the optimal value of ρ depends on the underlying sparsity structure.

2.8 Conclusion

In this chapter, we propose an adaptive learning algorithm for finding a sparse Mahalanobis distance metric on a nonlinear feature space via a gradient boosting algorithm for binary classification. We especially introduced sparsity control that results in au-

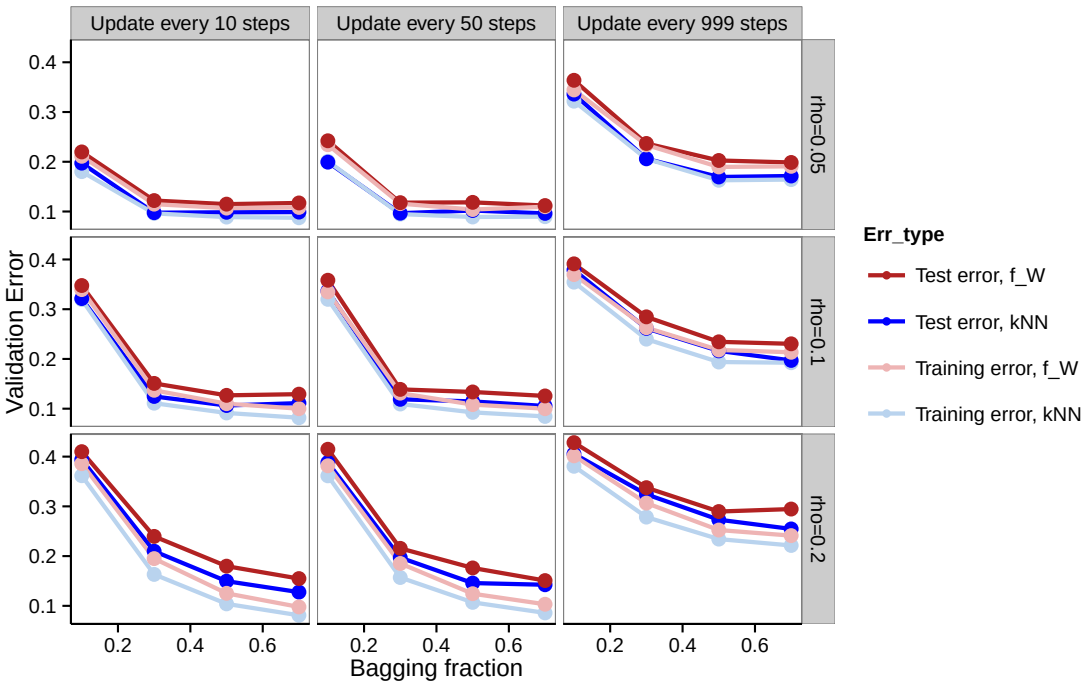


Figure 2.5: Sensitivity analysis of different configurations of the tuning parameters in the *sDist* algorithm: frequency of local neighborhood updates, the bagging fraction η, and the degree of sparsity ρ using the Madelon dataset. Training errors and testing errors are reported for both kNN classifier and the base classifier f_W in (2.3) based on 20 randomly partitioned 5-fold cross-validations.

tomatic feature selection. The *sDist* framework can be further extended in several directions. First, our framework can be generalized to multiclass problems. The base discriminant function in (2.3) can be extended for a multi-class response variable as in [Zhu *et al.*, 2009] for multi-class AdaBoost. More specifically, the class label c_i is recoded as a K-dimensional vector $\mathbf{y}_i = \{y_{i,1}, y_{i,2}, \ldots, y_{i,K}\}^T$ with K being the number of classes. Here $y_{ij} = 1$ if $c_i = j$ and $-\frac{1}{K-1}$ otherwise. Then a natural generalization of loss function in (2.5) is given by:

$$L(\mathbf{y}, f_W^{\phi}) = \sum_{i=1}^{n} \exp\left(-\frac{1}{K}\mathbf{y}_i^T f_W^{\phi}(\mathbf{x}_i)\right).$$

The other way is to redefine the local positive/negative neighborhood as the local similar/dissimilar neighborhood as in [Blitzer *et al.*, 2005], where the *similar* points refer to sample points with the same class label and the *dissimilar* ones are those with different class labels. However, a rigorous discussion on extension to muticlass problems requires extensive analysis. It is not straightforward in how to exactly address multiclass labels in metric learning, or whether the learning goal is to determine a common metric for all classes or to construct different metrics between pairs of classes. Due to the limited scope of this chapter, we will leave these questions in future studies.

Furthermore, in the proposed *sDist* algorithm, we approached the fitting of nonlinear decision boundary through interaction expansion and local neighborhoods. It has been noted that distance measures have close connections with kernel functions, which is commonly used for nonlinear learning methods in the literature. Integrating the nonlinear distance metric learning with kernel methods will lead to more flexible and powerful classifiers.

Chapter 3

Stabilized Sparse Online Learning for Sparse Data

3.1 Introduction

Although the *sDist* algorithm in Chapter 2 attains favorable results for learning informative feature subspaces for high-dimensional data, it is not yet effective if the data is not only of large dimenaionality but also highly sparse.Modern datasets pose many challenges for existing learning algorithms due to their unprecedented large scales in both sample sizes and input dimensions. It demands both efficient processing of massive data and effective extraction of crucial information from an enormous pool of heterogeneous features. In response to these challenges, a promising approach is to exploit online learning methodologies that performs incremental learning over the training samples in a sequential manner. In an online learning algorithm, one sample instance is processed at a time to obtain a simple update, and the process is repeated via multiple passes over the entire training set. In comparison with batch learning algorithms in which all sample points are scrutinized at every single step, online learning algorithms have been shown to be more efficient and scalable for data of large size that cannot fit into the limited memory of a single computer. As a result,

online learning algorithms have been widely adopted for solving large-scale machine learning tasks [Bottou, 1998].

In this chapter, we focus on first-order subgradient-based online learning algorithms, which have been studied extensively in the literature for dense data.[1] Among these algorithms, popular methods include the Stochastic Gradient Descent (SGD) algorithm [Zhang, 2004] [Bottou, 2010], the mirror descent algorithm [Beck and Teboulle, 2003] and the dual averaging algorithm [Nesterov, 2009]. Since these methods only require the computation of a (sub)gradient for each incoming sample, they can be scaled efficiently to high-dimensional inputs by taking advantage of the finiteness of the training sample. In particular, the stochastic gradient descent algorithm is the most commonly used algorithm in the literature of subgradient-based online learning. It enjoys an exceptionally low computational complexity while attaining steady convergence under mild conditions [Bottou, 1998], even for cases where the loss function is not everywhere differentiable.

Despite of their computational efficiency, online learning algorithms without further constraint on the parameter space suffers the "curse of dimensionality" to the same extent as their non-online counterparts. Embedded in a dense high-dimensional parameter space, not only does the resulted model lack interpretability, its variance is also inflated. As a solution, *sparse online learning* was introduced to induce sparsity in the parameter space under the online learning framework [Langford *et al.*, 2009]. It aims at learning a linear classifier with a sparse weight vector, which has been an active topic in this area. For most efforts in the literature, sparsity is introduced by applying L_1 regularization on a loss function as in the classific LASSO method [Tibshirani, 1996b] [Shalev-Shwartz and Tewari, 2011]. For example, [Duchi and Singer, 2009] extend the framework of Forward-Backward splitting [Lions and Mercier, 1979]

[1] *Dense data* is defined as a dataset in which the number of nonzero entries in all columns of its design matrix are in the order of $O(n)$ while the ones of *sparse data* are in the order of $O(\log(n))$ or less.

by alternating between an unconstrained truncation step on the sample gradient and an optimization step on the loss function with a penalty on the distance from the truncated weight vector. [Langford *et al.*, 2009] and [Carpenter, 2008] both explore the idea of imposing a *soft-threshold* on the weight vector $\mathbf{w} \in \mathbb{R}^p$ updated by the stochastic gradient descent algorithm:

$$w_j = \text{sign}(w_j) \max(|w_j| - \lambda, 0), \quad j = 1, \ldots, p.$$

This class of methods is known as the *truncated gradient algorithm*. For every K standard SGD updates, the weight vector is shrunk by a fixed amount to induce sparsity. In the work of [Duchi *et al.*, 2010], the same strategy has also been combined with a variant of the mirror descent algorithm [Beck and Teboulle, 2003]. [Wang *et al.*, 2015] further extends the truncated gradient framework to adjust for cost-effectiveness. This simple yet efficient method of truncated gradients particularly motivates the algorithm proposed in this chapter. Strategies different from the truncation-based algorithm have also been proposed. For example, [Xiao, 2009] proposes the Regularized Dual-Averaging (RDA) algorithm which builds upon the primal-dual subgradient method by [Nesterov, 2009]. The RDA algorithm learns a sparse weight vector by solving an optimization problem using the running average over all preceding gradients, instead of a single gradient at each iteration.

Closely related to sparse online learning is another area of active research, *online feature selection*. Instead of enforcing just a shrinkage on the weight vectors via L_1 regularization, online feature selection algorithms explicitly invoke feature selection by imposing a hard L_0 constraint on the weight vector, such as [Wang *et al.*, 2014] [Wu *et al.*, 2014]. In other words, online feature selection algorithms focus on generating a resulted weight vector that has a high sparsity level by directly shrinking a large proportion of the weights directly to zero (also referred to as a *hard thresholding*). In practice, L_0 regularization is computationally expensive to solve due to its non-differentiability. The set of selected features also suffers from high variability as the decisions of hard-thresholding are based on single random samples in an online

learning setting. Therefore, important features can be discarded simply owing to random perturbations.

Most current subgradient-based online learning algorithms do not consider potential structures or heterogeneity in the input features. As pointed out by [Duchi *et al.*, 2011], current methods largely follow a predetermined procedural scheme that is oblivious to the characteristics of data being used at each iteration. In large-scale applications, a common and important structure is heterogeneity in sparsity levels of the input features, i.e., the variability in the number of nonzero entries among features. For instance, consider the bag-of-word features in text mining applications.[2] For a learning task, the importance of a feature is not necessarily associated with the frequencies of its values. In genetics, for example, rare variants ($\leq 1\%$ in the population) have been found to be associated with disease risks [ref]. Both dense and sparse features may contain important information for the learning task. However, in the presence of heterogeneity in sparsity levels, using a simple L_1 regularization in an online setting will predispose rare features to be truncated more than necessary. The resulted sparse weight vectors usually exhibit high variance in terms of both weight values and the membership in the set of features with nonzero weights. As a result, the convergence of the standard truncation-based framework may also be hampered by this high variability. When the amount of information is scarce due to sparsity at each iteration, the convergence of the weight vector would understandably take a large number of iterations to approach the optimum. In two recent papers, [Oiwa *et al.*, 2011] and [Oiwa *et al.*, 2012] tackle this problem via L_1 penalty weighted by the accumulated norm of subgradients for extending several basic frameworks in sparse online learning. Their results suggest that, by acknowledging the sparsity structure in the features, both prediction accuracy and sparsity are improved over the origi-

[2]Here, by *sparse features*, we refer to features for which most samples assume a constant value (e.g., 0) and a few samples take on other values. Without loss of generality, we assume the majority constant is 0 throughout this chapter.

nal algorithms while maintaining the same convergence rate. However, their resulted weight vectors are unstable as the imposed subgradient-based regularization are excessively noisy due to the randomness of incoming samples in online learning. The membership in the set of selected features with nonzero weights is also very sensitive to the orderings of the training samples.

In this chapter, we propose a *stabilized truncated stochastic gradient descent* algorithm for high-dimensional sparse data. The learning framework is motivated by that of the Truncated Gradient SGD algorithm proposed by [Langford *et al.*, 2009]. To deal with the aforementioned issues with sparse online learning methods applied to high-dimensional sparse data, we introduce three innovative components to reduce variability in the learned weight vector and stabilize the selected features. First, when applying the soft-thresholding, instead of a uniform truncation on all features, we perform only *informative truncations*, based on actual information from individual features during the preceding computation window of K updates. By doing so, we eliminate the heterogeneous truncation bias associated with feature sparsity. The key idea here is to ensure that each truncation for each feature is based on sufficient information, and the amount of shrinkage is adjusted for the information available on each feature. Second, beyond the soft-thresholding corresponding to the ordinary L_1 regularization, the resulted weight vector is *stabilized* by staged purges of irrelevant features permanently from the active set of features. Here, *irrelevant features* are defined as features whose weights have been repeatedly truncated. Motivated by *stability selection* introduced in [Meinshausen and Bühlmann, 2010], these permanent purges prevent irrelevant features from oscillating between the active and non-active set of features, The "*purging*" process also resembles hard-thresholding in online feature selection and results in a stabler sparse solution than other sparse online learning algorithms. Results on the theoretical regret bound in 3.4 show that this stabilization step helps improve over the original truncated gradient algorithm, especially when the target weight vector is notably sparse. To attune the proposed

learning algorithm to the sparsity of the remaining active features, the third component of our algorithm is adjusting the amount of shrinkage progressively instead of fixing it at a predetermined value across all stages of the learning process. A novel hyperparameter, *rejection rate*, is introduced to balance between *exploration* of different sparse combinations of features at the beginning and the *exploitation* of the selected features to construct accurate estimate at a later stage. Our method gradually anneal the rejection rate to acquire the necessary amount of shrinkage on the fly for achieving the desired balance.

The rest of paper is organized as follows. Section 3.2 reviews the truncated Stochastic Gradient Descent (SGD) framework for sparse learning proposed in [Langford *et al.*, 2009]. In Section 3.3, we introduce, in details, the three novel components of our proposed algorithm. Theoretical analysis of the expected online regret bound is given in Section 3.4, along with the computational complexity. Section 3.5 gives practical remarks for efficient implementation. In Section 3.6, we evaluate the performance of the proposed algorithm on several real-world high-dimensional datasets with varying sparsity levels. We illustrate that the proposed method leads to improved stability and prediction performance for both sparse and dense data, with the most improvement observed in data with the highest average sparsity level. Section 3.7 concludes with further discussion on the proposed algorithm.

3.2 Truncated Stochastic Gradient Descent for Sparse Learning

Assume that we have a set of training data $\mathcal{D} = \{z_i = (\mathbf{x}_i, y_i), i = 1, \ldots, n\}$, where the feature vector $\mathbf{x}_i \in \mathbb{R}^p$ and the scalar output $y_i \in \mathbb{R}$. In the following, we use $\mathbf{x}_i$ to represent the vector of the i^{th} sample of length p and $\mathbf{x}_{\cdot,j}$ for the j^{th} feature vector of all samples of length n. In this chapter, we are interested in the case that both p and n are large and the feature vectors $\mathbf{x}_{\cdot,j}$'s, $j = 1, \ldots, p$, are sparse. We consider a

loss function $l(\hat{y}, y)$ that measures the cost of predicting $\hat{y}$ when the truth is y. The prediction $\hat{y}$ is given by function $f_{\mathbf{w}}(\mathbf{x})$ from a family $\mathcal{F}$ parametrized by a weight vector $\mathbf{w}$. Denote $L(\mathbf{w}, \mathbf{z}) \stackrel{\text{def}}{=\!=} l(f_{\mathbf{w}}(\mathbf{x}), y)$. The learning goal is to obtain an optimal weight vector $\hat{\mathbf{w}}$ that minimize the loss function $\sum_{i=1}^{n} L(\mathbf{w}, z_i)$ over the training data, with sparsity in the weight vector induced by a regularization term $\Psi(\mathbf{w})$. We can then formulate the learning task as a regularized minimization problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w} \in \mathbb{R}^p} \sum_{i=1}^{n} L(\mathbf{w}, z_i) + \Psi(\mathbf{w}). \tag{3.1}$$

The above optimization problem is often solved using some version of *gradient descent*. When both p and n are large, the computation becomes very demanding. To address this computational complexity, the Stochastic Gradient Descent (SGD) algorithm was proposed [Bottou, 1998] as a stochastic approximation of the full gradient algorithm. Instead of computing the gradient over the entire training set as under the batch setting, the stochastic gradient descent algorithm uses approximate gradients based on subsets of the training data. This is particularly attractive to large scale problems as it leads to a substantial reduction in computing complexity and potentially distributed implementation.

For applications with large data sets or streaming data feeds, SGD has also been used as a subgradient-based *online learning* method. Online learning and stochastic optimization are closely related and interchangeable most of the time [Cesa-Bianchi *et al.*, 2004]. For simplicity, in the following, we focus our discussion and algorithmic description under the online learning framework with regret bound models. Nonetheless, our results can be readily generalized to stochastic optimization as well.

In online learning, the algorithm receives a training sample $z_t = (\mathbf{x}_t, y_t)$ at a time from a continuous feed. Without sparsity regularization, at time t, the weight vector is updated in an online fashion with a single training sample $z_t \in \mathcal{D}$ drawn randomly,

$$\mathbf{h}_t = \mathbf{w}_t - \eta L'(\mathbf{w}_t, z_t), \tag{3.2}$$

$$\mathbf{w}_{t+1} = \mathbf{h}_t, \tag{3.3}$$

where $\eta > 0$ is the learning rate and $L'(\mathbf{w}_t, z_t) \in \partial_{\mathbf{w}_t} L(\mathbf{w}_t, z_t)$ is a subgradient of the loss function $L(\mathbf{w}_t, z_t)$ with respect to $\mathbf{w}_t$. The set of subgradients of f at the point x is called the subdifferential of f at x, and is denoted $\partial f(x)$. A function f is called subdifferentiable if it is subdifferentiable at all $x \in \mathbf{dom}\ f$. When $L(\mathbf{w}, \cdot)$ is differentiable at $\mathbf{w}$, $\partial_{\mathbf{w}} L(\mathbf{w}, \cdot) = \{\nabla_{\mathbf{w}} L(\mathbf{w}, \cdot)\}$. At the same time, a sequence of decisions $\mathbf{w}_t$ is generated at $t = 1, 2, \ldots$, that encounters a loss $L(\mathbf{w}_t, z_t)$ respectively.

Given an optimal decision $\mathbf{w} \in \mathbb{R}^p$, the goal of online learning with regularization is to achieve low *regret* defined as

$$R_T(\mathbf{w}) \triangleq \sum_{t=1}^{T} \left(L(\mathbf{w}_t, z_t) + \Psi(\mathbf{w}_t)\right) - \sum_{t=1}^{T} \left(L(\mathbf{w}, z_t) + \Psi(\mathbf{w})\right). \tag{3.4}$$

In this chapter, we focus on the L_1 regularization where $\Psi(\mathbf{w}) = g||\mathbf{w}||_1$ and g is the regularizing parameter. When adopted in an online learning framework, standard SGD algorithm does not work well in addressing (3.1) with L_1 penalty. Firstly, a simple online update requires the projection of the weight vector $\mathbf{w}$ onto a L_1-ball at each step, which is computationally expensive with a large number of features. Secondly, with noisy approximate subgradient computed using a single sample, the weights can easily deviate from zero due to the random fluctuations in z_t's. Such a scheme is therefore inefficient to maintain a sufficiently sparse weight vector.

To address this issue, [Langford *et al.*, 2009] induced sparsity in $\mathbf{w}$ by subjecting the stochastic gradient descent algorithm to soft-thresholding. For every $K \in \mathbb{N}^+$ iterations at step t, each of which is as defined in (3.2), the weight vector is shrunk by a soft-threshold operator T with a *gravity parameter* $\mathbf{g} \in \mathbb{R}^p$ with $g_j \geq 0$ for $j = 1, \ldots, p$. For a vector $\mathbf{h} = [h_1, \ldots, h_p] \in \mathbb{R}^p$,

$$\hat{\mathbf{w}}_{t+1} = T(\mathbf{h}_t, \mathbf{g}), \tag{3.5}$$

where $T(\mathbf{h}, \mathbf{g}) = [T(h_1, g_1), \ldots, T(h_p, g_p)]$ with the operator T defined by

$$T(h_j, g_j) \triangleq \begin{cases} \max(h_j - g_j, 0), & \text{if } h_j > 0; \\ \min(h_j + g_j, 0), & \text{if } h_j \leq 0. \end{cases} \tag{3.6}$$
$$= \text{sign}(h_j) \max(|h_j| - g_j, 0).$$

As one can see, the sequence of K SGD updates can be treated as a unit computational block, which will be referred to as a *burst* hereafter. Here the word *burst* indicates that it is a sequence of repetitive actions, e.g., the standard SGD updates as defined in (3.2), without interruption. Each burst is followed by a *soft-thresholding truncation* defined in (3.5), which puts a shrinkage on the learned weight vector.

A burst can be viewed as a base feature selection realized on a set of random samples with L_1 regularization as in the classical LASSO [Tibshirani, 1996b]. Within a burst, let $\mathcal{X}_K$ be the set of K random samples on which the weight vector $\hat{\mathbf{w}}$ is stochastically learned. We define the set of features with nonzero weights in $\hat{\mathbf{w}}$ as its *active (feature) set*:

$$\hat{\mathcal{S}}^g(\hat{\mathbf{w}}; \mathcal{X}_K) = \{j : |\hat{w}_j| > 0\}, \tag{3.7}$$

with a corresponding gravity $\mathbf{g}$. The steps within a truncated burst are summarized in Algorithm 3.1.

In the truncated gradient algorithm of [Langford *et al.*, 2009], the gravity parameter is a constant across all dimensions as $\mathbf{g} = g_0 K \mathbf{1}_p$, where $g_0 \in \mathbb{R} \geq 0$ is a *base gravity* for each update in a burst and $\mathbf{1}_p \triangleq (1, \ldots, 1) \in \mathbb{R}^p$. In general, with greater parameter g_0 and smaller burst size K, more sparsity is attained. When $g_0 = 0$, the update in (3.5) becomes identical to the standard stochastic gradient descent update in (3.2). [Langford *et al.*, 2009] showed that this updating process can be regarded as an online counterpart of L_1 regularization in the sense that it approximately solves (3.1) in the limit as as $K \to \infty$ and $\eta \to 0$.

Algorithm 3.1 $B_0(w_0, g_0)$: Truncated burst of K updates with universal gravity.

Input: $\mathbf{w}_0$ at initialization and the base gravity g_0.

Parameters: K, η.

for $t = 1$ **to** K **do**

Draw $z_t \in \mathcal{D}$ uniformly at random.

$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta L'(\mathbf{w}_{t-1}, z_t)$, where $L'(\mathbf{w}_{t-1}, z_t) \in \partial_{\mathbf{w}_{t-1}} L(\mathbf{w}_{t-1}, z_t)$.

end for

$\hat{\mathbf{w}} = T(\mathbf{w}_K, g_0 K \mathbf{1}_p)$.

Return: $\hat{\mathbf{w}}$.

3.3 Stabilized Truncated SGD for Sparse Learning

Truncated SGD [Langford *et al.*, 2009] works well for dense data. When it comes to high-dimensional *sparse* inputs, however, it suffers from a number of issues. [Shalev-Shwartz and Tewari, 2011] observe that the truncated gradient algorithm is incapable of maintaining sparsity of the weight vector as it iterates. Recall that, under the online learning setting, the weight vector $\mathbf{w}$ is updated with a noisy approximation of the true expected gradient using one sample at a time, from a random ordering of the data. With sparse inputs, it is highly probable that an important feature does not have a nonzero entry for many consequent samples, and is meaningfully updated for only a few times out of the K updates in a burst. As a result, it would be truncated after a few iterations and brought back to nonzero after another few updates. At the same time, sparsity in inputs will also give rise to sporadic large nonzero updates for irrelevant features, which cannot be fully resolved by the soft-threshold operator. The derived weight vector $\mathbf{w}$'s are of high variance, inadequate sparsity and poor generalizability. As an example, the number of nonzero variables in the weight vector during the last 1000 stochastic updates from the truncated gradient algorithm implemented on a high-dimensional sparse dataset (Dexter text mining data set; see Section 3.6 for details.) are shown in Figure 3.1. It can be seen that

the numbers of nonzero features in the weight vectors learned by the truncated SGD algorithm ($K = 5$) remain large and highly unstable throughout these 1000 iterations, oscillating within 10% of the total number of features. As a comparison, also in Figure 3.1, we plot the results from our proposed stabilized truncated SGD applied to the same data. During these last 1000 updates, the proposed algorithm is using a less frequent truncation schedule due to our *annealed reject rate*. It attains both high sparsity in the weight vector and high stability with high-dimensional sparse data.

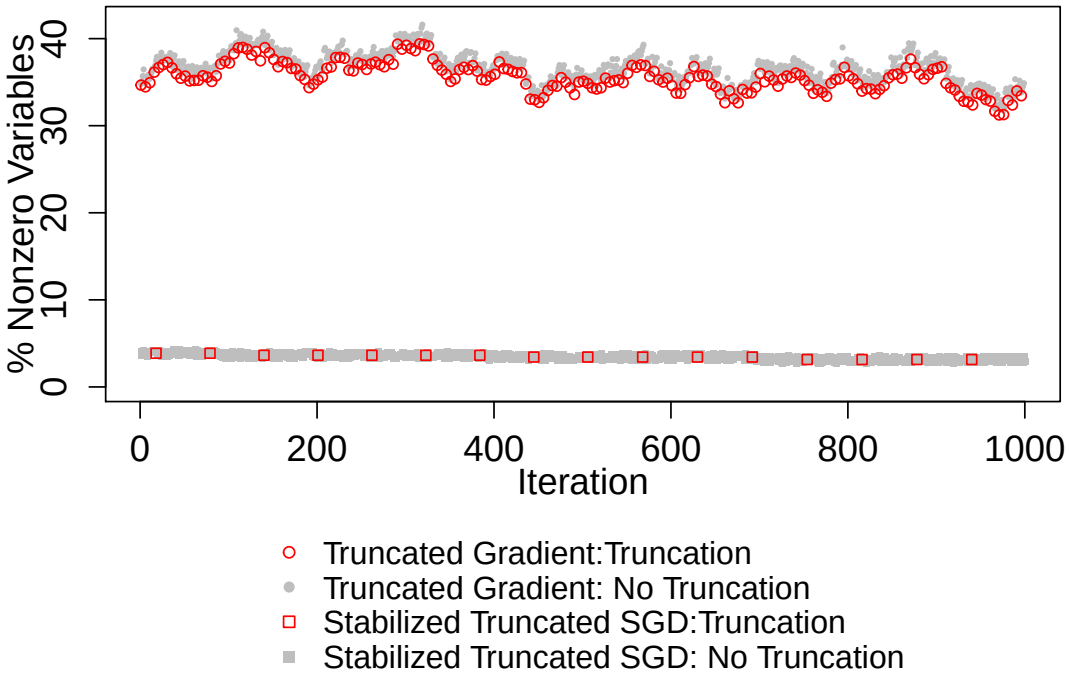


Figure 3.1: An example of the truncated gradient algorithm and the proposed stabilized truncated SGD algorithm applied to a high-dimensional sparse data set. Here we compare the percentage of nonzero variables in the resulted weight vector at each iterration during the last 1000 iterations of both algorithms. The underlying data set is the text mining dataset, *Dexter*, with $10,000$ features and 0.48% of sparsity, which is described in details in Section 3.6.

In this section, we introduce the *stabilized truncated Stochastic Gradient Descent (SGD)* algorithm. It attains a truly sparse weight vector that is stable and gives generalizable performance. Our proposed method attunes to the sparsity of each feature and adopts *informative truncation*. The algorithm keeps track of whether individual features have had enough information to be confidently subject to soft-thresholding. Based on the truncation results, we systematically reduce the active feature set by permanently discarding features that are truncated to zero with high probability via *stability selection*. We further improve the efficiency of our algorithm by adapting gravity to the sparsity of the current active feature set as the algorithm proceeds.

3.3.1 Informative Truncation

For the truncated SGD algorithm, [Langford *et al.*, 2009] suggest a general guideline for determining gravity in the batch mode by scaling a base gravity g_0 by K, the number of updates, for a single truncation after a burst. A direct online adaptation of a L_1 regularization would shrinks the weight vector at every iteration. The above batch mode operation is to delay the shrinkage for K iterations so that the truncation is executed based on information collected from K random samples instead of from a single instance. This guideline implicitly assumes that the K SGD updates in a burst are equally informative, which is in general true for dense features. For sparse features, however, under the online learning setting, not every update is informative about every feature due to the scarcity of nonzero entries. The original uniform formula, $\mathbf{g} = Kg_0\mathbf{1}_p$, for gravity would then create an undesirable differential treatment for features with different levels of sparsity. With a relatively small K, it is very likely that a substantial proportion of features would have no non-zero values on a size-K subsample used in a particular burst. The weights for these features remain unchanged after K updates. Consequently, the set of sparse features run the risk of being truncated to zero based on very few informative updates. The truncation

Algorithm 3.2 $B_1(w_0, g_0)$: A burst of K updates with informative truncation.

Input: $\mathbf{w}_0$ at initialization and the base gravity g_0.

Initialization: $\tilde{\mathbf{k}} = \mathbf{0}_p \in \mathbb{R}^p$.

Parameters: K, η.

for $t = 1$ **to** K **do**

Draw $z_t = (\mathbf{x}_t, y_t) \in \mathcal{D}$ uniformly at random.

$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta L'(\mathbf{w}_{t-1}, z_t)$, where $L'(\mathbf{w}_{t-1}, z_t) \in \partial_{\mathbf{w}_{t-1}} L(\mathbf{w}_{t-1}, z_t)$.

$\tilde{\mathbf{k}} \leftarrow \tilde{\mathbf{k}} + \mathbb{1}(|\mathbf{x}_t| > 0)$.

end for

$\hat{\mathbf{w}} = T(\mathbf{w}_K, g_0\tilde{\mathbf{k}})$.

Return: $\hat{\mathbf{w}}$, $\tilde{\mathbf{k}}$.

decision is therefore mostly determined by a feature's sparsity level, rather than its relevance to the class boundary.

To make the learning be informed of the heterogeneity in sparsity level among features, we introduce the *informative truncation* step, extended from the idea of base gradient used in Algorithm 3.1. Instead of applying a universal gravity proportional to K to all features, the amount of shrinkage is set proportional to the number of times that a feature is actually updated with nonzero values in the size-K subsample, i.e., the number of *informative updates*. Specifically, within each burst, the algorithm keeps a vector of *counters*, $\tilde{\mathbf{k}} \in \mathbb{R}^p$, of the numbers of informative updates for the features $\mathbf{x}_{\cdot,j}$, $j = 1, \ldots, p$. Let $g_0 \in \mathbb{R} \geq 0$ be the base gravity parameter that serves as the unit amount of shrinkage for each informative update on each feature. At the end of each burst, we shrink feature $\mathbf{x}_{\cdot,j}$ by $g_0\tilde{k}_j$. In other words, here we set $\mathbf{g} = g_0\tilde{\mathbf{k}}$. The computational steps for a burst with informative truncation in summarized in Algorithm 3.2.

A theoretical justification of informative truncation can be found in Lemma 2 (Section 3.4). This feature-specific gravity attunes to the sparsity structure incurred at each burst without *ad-hoc* adjustment. It also avoids data pre-processing for

locating sparse entries, which can be computationally expensive and compromises the advantage of online computation. In comparison to the truncated gradient algorithm in [Langford *et al.*, 2009] that quickly shrinks many features to zero indiscriminately, informative truncation keeps sparse features until enough evaluation is conducted. In doing so, sparse yet important features will be retained. The proposed approach also reduce the variability in the resulted sparse weight vector during the training process. [Duchi *et al.*, 2011] uses a similar strategy that allows the learning algorithm to adaptively adjust its learning rates for different features based on cumulative update history. They use the L_2 norm of accumulated gradients to regulate the learning rate. By adapting the gravity with the counter $\tilde{\mathbf{k}}$ within each burst, our proposed strategy here can be viewed as applying the L_0 norm to the accumulated gradients that is refreshed every K steps.

3.3.2 Stability Selection

Despite of its scalability, subgradient-based online learning algorithms commonly suffer from instability. It has been shown both theoretically and empirically that stochastic gradient descent algorithms are sensitive to random perturbations in training data as well as specifications of learning rate [Toulis *et al.*, 2015] [Hardt *et al.*, 2015]. This instability is particularly pronounced in sparse online learning with sparse data, as discussed in Section 1. Under an online learning setting, using random ordering of the training sample as inputs, the algorithm would produce distinct weight vectors and unstable memberships of the final active feature set. Moreover, there has been a lot of discussion, in the literature, on the link between the instability of an learning algorithm and its deteriorated generalizability [Bousquet and Elisseeff, 2002], [Kutin and Niyogi, 2002], [Rakhlin *et al.*, 2005], [Shalev-Shwartz *et al.*, 2010].

To tackle this instability issue, in the proposed algorithm, we exploit the method of *stability selection* to improve its robustness to random perturbation in the training data. Stability selection [Meinshausen and Bühlmann, 2010] does not launch a new

feature selection method. Rather, its aim is to enhance and improve a sparse learning method via subsampling. The key idea of stability selection is similar to the generic bootstrap [Meinshausen and Bühlmann, 2010]. It feeds the base feature selection procedure with multiple random subsamples to derive an empirical selection probability. Based on aggregated results from subsamples, a subset of features is selected with low variability across different subsamples. With proven consistency in variable selection, stability selection helps remove noisy irrelevant features and thus reduce the variability in learning a sparse weight vector.

Incorporating stability selection into our proposed framework, each truncated burst with gravity parameter $\mathbf{g}$ is treated as an individual sparse learning engine. It takes K random samples and carries out a feature selection to obtain a sparse weight vector. In the following, we define first the notion of *selection probability* for the stability selection step in our proposed algorithm.

Definition 3.1 (Selection Probability)**.** *Let $\mathcal{X}_K$ be a random subsample of $\{1, \ldots, n\}$ of size K, drawn without placement. Parametrized by the gravity parameter $\mathbf{g}$, the probability of the feature $\mathbf{x}_{\cdot,j}$ being in the active set of a truncated burst that returns $\hat{\mathbf{w}}$ is*

$$\Pi_j^{\mathbf{g}} = P^*\left(j \in \hat{S}^{\mathbf{g}}(\hat{\mathbf{w}}; \mathcal{X}_K)\right) = \mathbb{E}_{\mathcal{D}}\left[\mathbb{1}(|\hat{w}_j| > 0)\right],$$

where the probability P^* is with respect to the random subsampling of $\mathcal{X}_K$. Let $\Pi^{\mathbf{g}} = [\Pi_1^{\mathbf{g}}, \ldots, \Pi_p^{\mathbf{g}}]$.

For simplicity, we drop the superscript $\mathbf{g}$ of $\Pi^{\mathbf{g}}$ in later discussions. For the rest of the paper, the selection probability Π always refers to $\Pi^{\mathbf{g}}$ that corresponds to weight vector $\hat{\mathbf{w}}$ with gravity parameter $\mathbf{g}$.

Under unknown data distribution, the selection probabilities cannot be computed explicitly. Instead, they are estimated empirically. Since each truncation burst performs a screening on all features, the frequency of each feature being selected by a sequence of bursts can be used to derive an estimator of the selection probability. We

denote a sequence of $n_K > 0$ truncated bursts as a *stage*. A preliminary empirical estimate of the selection probability is given by

$$\hat{\Pi}_j = \begin{cases} \frac{\sum_{\tau:\tilde{k}_{j,\tau}>0} \mathbb{1}(|\hat{\mathbf{w}}_{j,\tau}|>0)}{\sum_{\tau=1}^{n_k} \mathbb{1}(\tilde{k}_{j,\tau}>0)}, & \text{for } j \text{ s.t. } \sum_{\tau=1}^{n_K} \tilde{k}_{j,\tau} > 0 \\ 1, & \text{otherwise} \end{cases}, \tag{3.8}$$

where $\tilde{\mathbf{k}}_\tau$ are returned counters of informative updates by each bursts for $\tau = 1, \ldots, n_K$.

Different from the conventional stability selection setting, $\hat{\mathbf{w}}_\tau$'s are obtained sequentially and thus are dependent with each other. When n_K is small, different subsamples produce selection probability estimates using (3.8) exhibit high variability, even when initialized with the same weight vector at $\tau = 1$. On the other hand, a large value of n_K requires a prohibitively large number of iterations for convergence. To resolve the issues of estimating selection probability using a single sequence of SGD updates, we introduce a multi-thread framework of updating paths. Multiple threads of sequential SGD updates are executed in a distributed fashion, which readily utilizes modern multi-core computer architecture. With M processors, we initialize the algorithm on each path of SGD updates with a random permutation of the training data, $\mathcal{D}$, denoted as $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(m)}$. Then independently, M stages of bursts run in parallel along M paths, which return with $\hat{\mathbf{w}}_\tau^{(m)}$, $\tau = 1, \ldots, n_K$, $m = 1, \ldots, M$. The joint estimate of selection probability with gravity $\mathbf{g}$ is obtained as

$$\hat{\Pi}_j = \begin{cases} \frac{\sum_{m=1}^{M} \sum_{\tau:\tilde{k}_{j,\tau}^{(m)}>0} \mathbb{1}(|\hat{\mathbf{w}}_{j,\tau}^{(m)}|>0)}{\sum_{m=1}^{M} \sum_{\tau=1}^{n_k} \mathbb{1}(\tilde{k}_{j,\tau}^{(m)}>0)}, & \text{for } j \text{ s.t. } \sum_{m=1}^{M} \sum_{\tau=1}^{n_K} \tilde{k}_{j,\tau}^{(m)} > 0 \\ 1, & \text{otherwise} \end{cases}, \tag{3.9}$$

When more processors are available, a smaller n_K is required for the algorithm to obtain a stable estimate of selection probability. The dependence among $\hat{\mathbf{w}}_\tau$'s is also attenuated when M random subsets of samples are used for the estimation. This strategy falls under parallelized stochastic gradient descent methods, which is discussed in detail by [Zinkevich *et al.*, 2010].

Under the framework of stability selection, each stage on every path uses a random subsample. The estimated selection probability quantifies the chance that a feature is found to have high relevance to class differences given a random subsample. At the end of each stage, *stable features* are identified as those that belong to a large fraction of active sets incurred during this stage of bursts.

Definition 3.2 (Stable Features)**.** *For a purging threshold* $\pi_0 \in [0, 1]$*, the set of stable features with gravity parameter* $\mathbf{g}$ *is defined as*

$$\hat{\Omega}^{\mathbf{g}} = \{j : \Pi_j^{\mathbf{g}} \geq \pi_0\}.$$

For simplicity, we write the stable set $\hat{\Omega}^{\mathbf{g}}$ as $\hat{\Omega}$ when there is no ambiguity.

Stability selection retains features that have high selection probabilities and discard those with low selection probabilities. At the end of a stage of M paths, we *purge* the features that are not in the set of stable features by permanently setting their corresponding weights to zero, and remove them from subsequent updates. We define the *stabilized* weight vector as

$$\tilde{\mathbf{w}} = \hat{\mathbf{w}} \cdot \mathbb{1}_{\hat{\Omega}}. \tag{3.10}$$

As discussed earlier, due to the nature of online learning with sparse data, there are two undesirable learning setbacks in a single truncated burst. The first occurs when an important feature has its weight stuck at zero due to inadequate information in the subsample used, while the second case is when a noise feature's weight gets sporadic large updates by chance. Using informative bursts (counted by), we can avert the first type of setbacks and using selection probability based on multiple bursts, we can spot noisy features more easily. In the presence of a large number of noisy features, the learned weights for important features suffer from high variance. Via stability selection, we systematically remove noisy features permanently from the feature pool. Furthermore, the choice of a proper regularization parameter is crucial yet known to be difficult for sparse learning, especially due to the unknown

Algorithm 3.3 $B_2(\mathbf{w}_0, g_0, \hat{\Omega}, \mathcal{D}^{(m)})$: Informative truncated burst with stability selection in thread m.

Input: $\mathbf{w}_0$, g_0, the input data $\mathcal{D}^{(m)}$ and the current set of stable features $\hat{\Omega}$, which is the output of equation (3.7) using (3.9) with predetermined threshold π_0.

Parameters: K, η.

Initialize $\mathbf{v}_0 = (w_{0,j})_{j\in\hat{\Omega}}$, $\tilde{\mathbf{k}} = \mathbf{0}_{|\hat{\Omega}|}$.

for $t = 1$ **to** K **do**

Draw $z_t = (\mathbf{x}_t, y_t)$ sequentially drawn from $\mathcal{D}^{(m)}$.

$\tilde{z}_t = (z_{t,j})_{j\in\hat{\Omega}}$.

$\mathbf{v}_t = \mathbf{v}_{t-1} - \eta L'(\mathbf{v}_{t-1}, \tilde{z}_t)$, , where $L'(\mathbf{v}_{t-1}, z_t) \in \partial_{\mathbf{v}_{t-1}} L(\mathbf{v}_{t-1}, z_t)$.

$\tilde{\mathbf{k}} \leftarrow \tilde{\mathbf{k}} + \mathbb{1}(|\mathbf{x}_t| > 0)$.

end for

$\hat{\mathbf{u}} = T(\mathbf{v}_K, g_0\tilde{\mathbf{k}})$.

$$\hat{w}_j = \begin{cases} \hat{u}_{j'}, & \text{if } j \in \hat{\Omega} \text{ and } \hat{\Omega}_{j'} = j \\ 0, & \text{if } j \notin \hat{\Omega} \end{cases}.$$

Return: $\hat{\mathbf{w}}$, $\tilde{\mathbf{k}}$.

noise level. Applying stability selection renders the algorithm less sensitive to choice of the base gravity parameter g_0 in learning a sparse weight vector via truncated gradient. As we will show using results from our numerical experiments, this purging by stability selection leads to a notable reduction in the estimation variance of the weight vector. Here, π_0 is a tuning parameter in practice. We have found that the learning results in the numerical experiments are not sensitive to different values of π_0 within a reasonable range. Under mild assumptions discussed in Section 3.4, we derive a lower bound of the expected improvement in convergence by employing stability selection in the learning process in Lemma 1.

3.3.3 Adaptive Gravity with Annealed Rejection Rate

The truncated gradient algorithm [Langford *et al.*, 2009] adopts a universal and fixed base gravity parameter at all truncations. As pointed out in [Langford *et al.*, 2009] , a large value of the base gravity g_0 achieves more sparsity but the accuracy is compromised, while a small value of g_0 leads to less sparse weight vector yet attaining better performance. In other words, different extents of shrinkage serve different purposes of a learning algorithm. The needs for shrinkage also changes as the weight vector (and the stable set) evolves. Intuitively, the truncation is expected to be greedy at the beginning so that the number of nonzero feature can be quickly reduced for better computational efficiency and learning performance. As the algorithm proceeds, fewer features remain in the stable set. We should then be careful not to shrink important features with a truncation that is too harsh.

A large base gravity g_0 is effective in inducing sparsity at the beginning of the algorithm when the weight vector $\hat{\mathbf{w}}$ is dense. As the algorithm proceeds, the same value of gravity is likely to impose too much shrinkage when the learned weight vector $\hat{\mathbf{w}}$ becomes very sparse, exposing some truly important features at the risk of being purged. On the other hand, a small fixed gravity is over-conservative so that the algorithm will not shrink irrelevant features effectively, leading to slow convergence and a dense weight vector overridden by noise. Tuning a reasonable fixed base gravity parameter for a particular data set does not only creates additional computational burden, but also inadequate in addressing different learning needs during different stages of the algorithm.

As the role of *gravity* in a learning algorithm is to induce sparse estimates, in this paper, we propose an *adaptive gravity* scheme that delivers the right amount of shrinkage at each stage of the algorithm towards a desirable level of sparsity for the learned weight vector. We propose to control sparsity by a target *rejection rate* β, that is, the proportion of updates that are expected to be truncated. Guided by this target rejection rate, we derive the necessary shrinkage amount and the corre-

sponding gravity. As we discussed in Section 3.3.1, a base gravity g_0 is used in our learning algorithms to create gravity values for individual features that are attuned to their data sparsity levels. Therefore our adaptive gravity scheme is carried out by adjusting g_0. At the beginning of a particular stage, we examine the truncation carried out during the previous stage. The base gravity g_0 is then adjusted to project the target rejection rate during the current stage. Specifically, at stage s, we look at the pooled set of non-truncated weight vectors and informative truncation counters $\left\{\mathbf{w}_\tau^{(m)}, \tilde{\mathbf{k}}_\tau^{(m)}, \tau = 1, \ldots, n_K, m = 1, \ldots, M\right\}$ from all the bursts conducted in the previous stages on multiple threads. The adaptive base gravity g_0 for a target rejection rate $\beta_s \in [0, 1]$ is then obtained as

$$g_{0,s}(\beta_s) \triangleq \sup\{g_0 \geq 0 : \hat{p}_s(g_0) \leq \beta_s\}. \tag{3.11}$$

Here $\hat{p}_s(g_0)$ is the empirical probability, i.e.,

$$\hat{p}_s(g_0) \triangleq \frac{\sum_{m=1}^{M}\sum_{\tau=1}^{n_k}\sum_{\{j:j\in\hat{\Omega}_s,\tilde{k}_{j,\tau}^{(m)}>0\}} \mathbb{1}\left(\left|\frac{\Delta w_{j,\tau}^{(m)}}{\tilde{k}_{j,\tau}^{(m)}}\right| > g_0\right)}{\sum_{m=1}^{M}\sum_{\tau=1}^{n_k}\sum_{j\in\hat{\Omega}_s} \mathbb{1}\left(\tilde{k}_{j,\tau}^{(m)} > 0\right)},$$

where $\Delta w_{j,\tau}^{(m)} \triangleq w_{j,\tau}^{(m)} - w_{j,\tau-1}^{(m)}$ is the amount of updates on feature $\mathbf{x}_{\cdot,j}$ during the τ^{th} burst.

In other words, we pool the updates in weight vectors learned from previous iterations within one stage and concatenate the scaled weights in the stable set with nonzero counter values as a single vector $\tilde{\mathbf{v}}_s$ of length $d_{\tilde{\mathbf{v}}_s}$, where

$$\tilde{\mathbf{v}}_s = \left(\frac{\Delta w_{j,\tau}^{(m)}}{\tilde{k}_{j,\tau}^{(m)}}\right)_{\left\{j\in\{j':j'\in\hat{\Omega}_s \text{ and } \tilde{k}_{j,\tau}^{(m)}>0\},\tau=1,\ldots,n_K,m=1,\ldots,M\right\}}. \tag{3.12}$$

The base gravity is set to be the β_s percentile of $\tilde{\mathbf{v}}_s$. Since the vector $\tilde{\mathbf{v}}_s$ is composed of discrete values, the adaptive base gravity $g_{0,s}$ can also be written as the l^{th} order statistics of $\tilde{\mathbf{v}}_s$ where $l = \beta_s d_{\tilde{\mathbf{v}}_s}$.

In other words, we pool the updates in weight vectors learned from previous

iterations within one stage and concatenate the scaled weights in the stable set with nonzero counter values as a single vector $\tilde{\mathbf{v}}_s$ of length $d_{\tilde{\mathbf{v}}_s}$, where

$$\tilde{\mathbf{v}}_s = \left(\frac{\Delta w_{j,\tau}^{(m)}}{\tilde{k}_{j,\tau}^{(m)}} \right)_{\left\{ j \in \{j' : j' \in \hat{\Omega}_s \text{ and } \tilde{k}_{j,\tau}^{(m)} > 0\}, \tau = 1, \ldots, n_K, m = 1, \ldots, M \right\}}. \tag{3.13}$$

The base gravity is set to be the β_s percentile of $\tilde{\mathbf{v}}_s$. Since the vector $\tilde{\mathbf{v}}_s$ is composed of discrete values, the adaptive base gravity $g_{0,s}$ can also be written as the l^{th} order statistics of $\tilde{\mathbf{v}}_s$ where $l = \beta_s d_{\tilde{\mathbf{v}}_s}$.

We initialize the algorithm with a high rejection rate so that a large proportion of the weight vector can be reduced to zero at the end of each burst during the early stage of the algorithm. It allows the algorithm to explore as many sparse combination of features as possible at the early stage of the learning process. Along with the stability selection, the set of stable features can be quickly reduced to a manageable size by removing the majority of noises. When the weight vector becomes sparse, we decrease the rejection rate proportionally. With a lower rejection rate, and consequently a lower gravity, the algorithm can better exploit the subsequent standard SGD updates for a more accurate estimate of the true weight vector. As the rejection rate decreases to 0, the algorithm converges to the standard stochastic gradient descent algorithm on a small subset of stable features.

To achieve the balance between exploration and exploitation, we construct an annealing function for the rejection rate that decreases monotonically as the level of sparsity decreases. Let $\beta_0 \in [0, 1]$ be the *maximum rejection rate* at initialization and let γ be the *annealing rate*. The annealing function ϕ for the rejection rate at stage $s + 1$ is given by

$$\begin{aligned} \beta_{s+1} &= \phi(d_s; \beta_0, \gamma) \\ &= \begin{cases} \beta_0 \left[\exp\left(-\gamma d_s\right) - d_s e^{-\gamma} \right] & \gamma \geq 0 \\ \beta_0 \frac{\log(1-\gamma(1-d_s))}{\log(1-\gamma)}, & \gamma < 0 \end{cases}, \end{aligned} \tag{3.14}$$

where $d_s = \frac{|\hat{\Omega}_s|}{p}$ assesses the level of sparsity at the end of stage s. The greater the

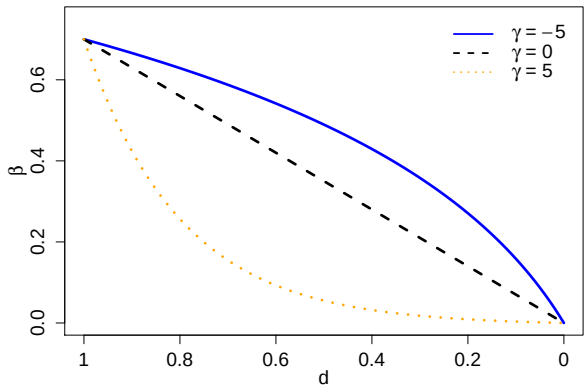


Figure 3.2: Examples of the rejection rate annealing function with different values of γ as defined in (3.14). Here $\gamma = -5, 0$, and 5 respectively. A positive annealing rate would reduce the rejection rate quickly as the proportion of non-zeros weight values, d_s, decreases, whereas a negative annealing rate would maintain it at a relatively high level.

value γ is, the faster the rejection rate is annealed to zero as the number of stable features decreases.

A positive, zero and negative value of γ corresponds to exponential decay, linear decay and logarithmic decay of the rejection rate, respectively. Figure 3.2 presents examples of the rejection rate anneal function with $\gamma = -5, 0$, and 5 respectively.

By using adaptive gravity (3.11) with annealed rejection rate (3.14), the amount of shrinkage is adjusted to the current level of sparsity of the weight vector quantified by the size of the stable set $|\hat{\Omega}_s|$ or the L_0 norm of the purged $\tilde{\mathbf{w}}$. Instead of tuning a fixed gravity parameter as in [Langford *et al.*, 2009], for our proposed algorithm, we tune the annealing rate γ and the maximum rejection rate β_0. Here γ balances the trade-off between exploration and exploitation and β_0 determines the initial intensity of truncation. It enables the tuning process to be tailored to the data at hand as well as being comparable across different datasets. In Section 3.6, the tuning results instantiate that a negative annealing rate is preferred for highly sparse data, such

as the RCV1 dataset, since the a high rejection rate needs to be maintained longer allowing sufficient information of sparse features to be evaluated by the learning process. On the other hand, a positive annealing rate is chosen for relatively dense data, such as the Arcene dataset, where the high frequency of nonzero values permit fast reduction of the active set. The complete algorithm of the stabilized truncated stochastic gradient descent algorithm is summarized in Algorithm 3.4.

3.4 Properties of the Stabilized Truncated Stochastic Gradient Descent Algorithm

The learning goal of sparse online learning is to achieve a low regret as defined in (3.4). In this section, we analyze the online regret bound of the proposed stabilized truncated SGD algorithm in Algorithm 3.4 with convex loss. For simplicity, the effect of adaptive gravity with annealed rejection rate is not considered here. To achieve viable result, we make the following assumptions.

Assumption 1. *The absolute values of the weight vector* $\mathbf{w}$ *are bounded above, that is,* $|w_j| \leq C$ *for some* $C \in [0, \infty)$*,* $j = 1, \ldots, p$*.*

Assumption 2. *The loss function* $L(\mathbf{w}, z)$ *is convex in* $\mathbf{w}$*, and there exist non-negative constants* A *and* B *such that, for all* $\mathbf{w} \in \mathbb{R}^p$ *and* $z \in \mathbb{R}^{p+1}$*,* $||\nabla_{\mathbf{w}} L(\mathbf{w}, z)||^2 \leq AL(\mathbf{w}, z) + B$*.*

For linear prediction problems, the class of loss function that satisfies Assumption 2 includes some common loss functions used in machine learning problems, such as the L_2 loss, the hinge loss and the logistic loss, with the condition that $\sup_{\mathbf{x}} ||x|| \leq C_x$ for some constant $C_x > 0$.

Assumption 3. *For* $\tau = 1, 2, \ldots,$ *the average number of active selected features from each truncated bursts,* $q^{\mathbf{g}}$*, given a gravity* $\mathbf{g}$*, is greater than or equal to the number of nonzero weights in the target weight vector* $\mathbf{w}^*$*, denoted as* d^**.*

Assumption 3 posits that the true parameter space of interest is substantially sparse, which is the main focus of sparse learning and of this chapter. Nevertheless, this condition does not confine the applicable scenarios of the proposed method to a fixed subclass of problems. It suggests a balance between the model sparsity and the value of the gravity parameter that is implicitly embedded within the parameter tuning process.

Lemma 1. *Let $\tilde{\mathbf{w}}$ be the stabilized weight vector after the non-stabilized dense weight vector $\hat{\mathbf{w}}$ being purged by the stability selection in* (3.10) *with a set of stable features $\hat{\Omega}$. Let $\mathbf{w}^*$ be the target sparse weight vector and d^* be the number of nonzero weights in $\mathbf{w}^*$. Let $q^{\mathbf{g}}$ be the average number of selected features from the truncated bursts on which the set of stability selection is constructed with the gravity parameter $\mathbf{g}$. Then, if Assumption 1 holds, there exists an $\varepsilon \in \left(0, \frac{\pi_0 C(p-|\hat{\Omega}|)}{|\hat{\Omega}|}\right)$ with $\hat{S}_\varepsilon = \{j : \mathbb{E}(|\hat{w}_j|) > \varepsilon\}$ such that the bound on the expected difference between the distance from the non-stabilized weight vector to the target and the distance from the stabilized weight vector to the target is given by*

$$\begin{aligned} &\mathbb{E}\left(||\hat{\mathbf{w}} - \mathbf{w}^*||^2 - ||\tilde{\mathbf{w}} - \mathbf{w}^*||^2\right) \\ &\geq \varepsilon^2(|\hat{S}_\varepsilon| - |\hat{\Omega}|) + 2\pi_0 C^2 \left[\left(1 - \frac{q^{\mathbf{g}}}{2\pi_0 p - p}\right) q^{\mathbf{g}} - d\right] \geq 0. \end{aligned} \tag{3.15}$$

Proof. Let $\Omega^* = \{j : |w_j^*| > 0\}$ be the set of nonzero features in the target weight vector $\mathbf{w}^*$. So $\mathbf{w}^*$ can also be written as $\mathbf{w}^* \mathbb{1}_{\Omega^*}$. Based on the stability selection in (3.10), we have $\tilde{\mathbf{w}} = \hat{\mathbf{w}} \mathbb{1}_{\hat{\Omega}}$.

Let $\Omega = \{1, \ldots, p\}$. The full set Ω can be further divided into four disjoint sets: $S_1 = \Omega \setminus (\Omega^* \cup \hat{\Omega})$, $S_2 = \Omega^* \setminus \hat{\Omega}$, $S_3 = \hat{\Omega} \setminus \Omega^*$, and $S_4 = \Omega^* \cap \hat{\Omega}$, respectively.

Firstly, we consider the L_2 distance from the non-stabilized weight vector $\hat{\mathbf{w}}$ to the target weight vector $\mathbf{w}^*$ and the L_2 distance from the stabilized weight vector $\tilde{\mathbf{w}}$

to $\mathbf{w}^*$:

$$\begin{aligned}||\hat{\mathbf{w}} - \mathbf{w}^*|| &= ||\hat{\mathbf{w}} - \mathbf{w}^* \mathbb{1}_{\Omega^*}||^2 \\ &= \sum_{j \in \Omega^*} (\hat{w}_j - w_j^*)^2 + \sum_{j \notin \Omega^*} \hat{w}_j^2, \\ &= \sum_{j \in S_2 \cup S_4} (\hat{w}_j - w_j^*)^2 + \sum_{j \in S_1 \cup S_3} \hat{w}_j^2.\end{aligned}$$

Since

$$(\hat{\mathbf{w}} \mathbb{1}_{\hat{\Omega}} - \mathbf{w}^* \mathbb{1}_{\Omega^*})_j^2 = \begin{cases} 0, & \text{if } j \notin \Omega^* \text{ and } j \notin \hat{\Omega}; \\ (w_j^*)^2, & \text{if } j \in \Omega^* \text{ and } j \notin \hat{\Omega}; \\ (\hat{w}_j)^2, & \text{if } j \notin \Omega^* \text{ and } j \in \hat{\Omega}; \\ (\hat{w}_j - w_j^*)^2, & \text{if } j \in \Omega^* \text{ and } j \in \hat{\Omega}, \end{cases}$$

$$\begin{aligned}||\tilde{\mathbf{w}} - \mathbf{w}^*||^2 &= ||\hat{\mathbf{w}} \mathbb{1}_{\hat{\Omega}} - \mathbf{w}^* \mathbb{1}_{\Omega^*}||^2 \\ &= \sum_{j \in S_2} (w_j^*)^2 + \sum_{j \in S_3} (\hat{w}_j)^2 + \sum_{j \in S_4} (\hat{w}_j - w_j^*)^2.\end{aligned}$$

Then

$$\begin{aligned}||\hat{\mathbf{w}} - \mathbf{w}^*||^2 - ||\tilde{\mathbf{w}} - \mathbf{w}^*|| &= \sum_{j \in S_2} (\hat{w}_j - w_j^*)^2 + \sum_{j \in S_1} \hat{w}_j^2 - \sum_{j \in S_2} (w_j^*)^2 \\ &= \sum_{j \in S_2} \hat{w}_j^2 + \sum_{j \in S_2} (w_j^*)^2 - 2 \sum_{j \in S_2} w_j^* \hat{w}_j - \sum_{j \in S_2} (w_j^*)^2 + \sum_{j \in S_1} \hat{w}_j^2 \\ &= \sum_{j \in S_1 \cup S_2} \hat{w}_j^2 - 2 \sum_{j \in S_2} w_j^* \hat{w}_j \\ &= ||\hat{\mathbf{w}}(1 - \mathbb{1}_{\hat{\Omega}})||^2 - 2 \sum_{j \in S_2} w_j^* \hat{w}_j \\ &\geq \hat{\mathbf{w}}_{\min}^2 (p - |\hat{\Omega}|) - 2 \sum_{j \in S_2} w_j^* \hat{w}_j. \end{aligned} \tag{3.16}$$

For the first part in (3.16), there exists some small $\varepsilon \in \left(0, \frac{\pi_0 C(p - |\hat{\Omega}|}{|\hat{\Omega}|}\right)$ with $\hat{S}_\varepsilon = \{j : \mathbb{E}(|\hat{w}_j|) > \varepsilon\}$ such that

$$\begin{aligned}
\mathbb{E}\left(||\hat{\mathbf{w}}(1-\mathbb{1}_{\hat{\Omega}})||^2\right) =& \mathbb{E}\left(\sum_{j=1}^{p}\hat{w}_j^2\mathbb{1}(\hat{p}_j \leq \pi_0)\right) \\
=& \mathbb{E}\left(\sum_{j=1}^{p}\hat{w}_j^2\mathbb{1}(\hat{p}_j \leq \pi_0)\mathbb{1}(|\hat{w}_j| \leq \varepsilon)\right) + \mathbb{E}\left(\sum_{j=1}^{p}\hat{w}_j^2\mathbb{1}(\hat{p}_j \leq \pi_0)\mathbb{1}(|\hat{w}_j| > \varepsilon)\right) \\
\geq& 0 + \varepsilon^2\mathbb{E}\left(\sum_{j=1}^{p}\mathbb{1}(\hat{p}_j \leq \pi_0)\mathbb{1}(|\hat{w}_j| \leq \varepsilon)\right) \\
\geq& \varepsilon^2(|\hat{S}_\varepsilon| - |\hat{\Omega}|) > 0,
\end{aligned}$$

where the last inequality is due to $|\hat{S}_\varepsilon| > |\hat{\Omega}|$ with the specified range of ε.

For the second part in (3.16), from on Theorem 1 of [Meinshausen and Bühlmann, 2010] in stability selection, we have

$$\mathbb{E}(|S_2|) \leq \left(\frac{q^g}{2\pi_0 p - p} - 1\right) q^g + d.$$

And, since, for any $j \in \Omega \setminus \hat{\Omega}$, $\Pr(|\hat{w}_j| > 0) \leq \pi_0$,

$$\mathbb{E}(|\hat{w}_j|) = \int_0^C \Pr(|\hat{w}_j| \geq u)du \leq \Pr(|\hat{w}_j| > 0)C \leq \pi_0 C.$$

Then,

$$\mathbb{E}\left(\sum_{j\in S_2} w_j^*\hat{w}_j\right) \leq \pi_0 C^2|S_2| \leq \pi_0 C^2|S_2|\left[\left(\frac{q^g}{2\pi_0 p - p} - 1\right) q^g + d\right].$$

Thus we have

$$\mathbb{E}\left(||\hat{\mathbf{w}} - \mathbf{w}^*||^2 - ||\tilde{\mathbf{w}} - \mathbf{w}^*||^2\right) \geq \varepsilon^2(|\hat{S}_\varepsilon| - |\hat{\Omega}|) + 2\pi_0 C^2\left[\left(1 - \frac{q^g}{2\pi_0 p - p}\right) q^g - d\right].$$

□

Lemma 1 quantifies the gain of using stabilization when the true weight vector is highly sparse. When the purging threshold π_0 is sufficiently large such that $\pi_0 \in \left(\frac{1}{2} + \frac{(q^{\mathbf{g}})^2}{2p(q^{\mathbf{g}} - d^*)}, 1\right)$, the lower bound achieved by (**??**) is guaranteed to be positive. Furthermore, this result also indicates that the expected difference in distances

to the sparse target weight vector between the non-stabilized and stabilized weight vector depends on the differences between the sizes of the temporary nonzero set of features before purging, $|\hat{S}_\varepsilon|$, and the size of the stable features after purging. In expectation, the stabilized weight vector is much closer to target sparse weight vector if the operation of purging can efficiently reduce the size of stable features. This suggests a much faster convergence with stabilization. Lemma 1 also provides an insight on the benefit from using adaptive gravity with annealed rejection rate. At the beginning of the algorithm, the gap between the size of $|\hat{S}_\varepsilon|$ and the size of the set of stable features $|\hat{\Omega}|$ is large when aiming for extensive exploration of different sparse combination of features. Hence, the improvement brought by stabilization is more substantial during the early state of learning period. As the algorithm proceeds and the set of stable features becomes smaller, it dwindles the leeway that allows the aforementioned two sets to be different. Consequently, deviation from the standard stochastic gradient descent algorithm is gradually reduced to facilitate better convergence at the later period of the learning process.

Lemma 2. *Let* $\mathbf{w}_0$ *be the weight vector at initialization. After the first burst, let* $\bar{\mathbf{w}}_1$ *be the truncated weight vector using universal gravity* $g_0 K$ *as in Algorithm* **??** *and let* $\hat{\mathbf{w}}_1$ *be the truncated weight vector with informative truncation as in Algorithm* **??**. *Let* $\mathbf{w}^* \in \mathbb{R}^p$ *be the target sparse weight vector. Assume features* $\mathbf{x}_1, \ldots, \mathbf{x}_p$ *has various sparsity distribution that* $\Pr(|x_{i,j}| > 0) = \lambda_j$, *where* $\lambda_j \in [0, 1]$, *for* $i = 1, \ldots, n$, *and* $\lambda_j = 0$ *if* $\mathbf{w}^*_j = 0$, *for* $j = 1, \ldots, p$. *Then,*

$$\mathbb{E}\left(||\bar{\mathbf{w}}_1 - \mathbf{w}^*||^2 - ||\hat{\mathbf{w}}_1 - \mathbf{w}^*||^2\right) \geq 2g_0 \sum_{t=1}^{K} \|\zeta_t \mathbf{w}^*\|_1 \geq 0 \tag{3.17}$$

where $\zeta_{t,j} = \mathbb{1}(|x_{t,j}| = 0)$ *for* $j = 1, \ldots, p$.

Proof. Based on Algorithm 3.1 and Algorithm 3.2, we have the following relationships:

$$\bar{\mathbf{w}}_1 = T(\mathbf{h}_1, Kg_0) = \text{sign}(\mathbf{h}_1) \max(|\mathbf{h}_1| - g_0 \mathbf{1}_p K, 0), \tag{3.18}$$

$$\hat{\mathbf{w}}_1 = T(\mathbf{h}_1, \tilde{\mathbf{k}} g_0) = \text{sign}(\mathbf{h}_1) \max(|\mathbf{h}_t| - g_0 \tilde{\mathbf{k}}_1, 0), \tag{3.19}$$

where

$$\mathbf{h}_1 = \mathbf{w}_0 - \sum_{t=1}^{K} \eta L'\left(\mathbf{w}_{t-1}, z_t\right).$$

Without loss of generality, we consider the difference in the squared errors to the optimal weight $\mathbf{w}^*$ between$\bar{\mathbf{w}}_1$ and $\hat{\mathbf{w}}_1$:

$$\begin{aligned}
&||\bar{\mathbf{w}}_1 - \mathbf{w}^*||^2 - ||\hat{\mathbf{w}}_1 - \mathbf{w}^*||^2 \\
=&||(\bar{\mathbf{w}}_1 - \mathbf{h}_1) - (\mathbf{w}^* - \mathbf{h}_1)||^2 - ||(\hat{\mathbf{w}}_1 - \mathbf{h}_1) - (\mathbf{w}^* - \mathbf{h}_1)||^2 \\
=&\left(||\bar{\mathbf{w}}_1 - \mathbf{h}_1||^2 - ||\hat{\mathbf{w}}_1 - \mathbf{h}_1||^2\right) - 2\left[(\bar{\mathbf{w}}_1 - \mathbf{h}_1)^T(\mathbf{w}^* - \mathbf{h}_1) - (\hat{\mathbf{w}}_1 - \mathbf{h}_1)^T(\mathbf{w}^* - \mathbf{h}_1)\right]
\end{aligned} \tag{3.20}$$

In the first part of (3.20), based on (3.18) and (3.19), we have

$$\begin{aligned}
||\bar{\mathbf{w}}_1 - \mathbf{h}_1||^2 &= pK^2 g_0^2, \\
||\hat{\mathbf{w}}_1 - \mathbf{h}_1||^2 &= ||\tilde{\mathbf{k}}_1||^2 g_0^2,
\end{aligned}$$

where $\tilde{\mathbf{k}}_1$ is the counter of informative updates in the first burst with $\tilde{k}_{1,j} \leq K$ for $j = 1, \ldots, p$. Hence,

$$||\bar{\mathbf{w}}_1 - \mathbf{h}_1||^2 - ||\hat{\mathbf{w}}_1 - \mathbf{h}_1||^2 \geq 0 \tag{3.21}$$

Based on (3.18) and (3.19),

$$\begin{aligned}
(\bar{\mathbf{w}}_1 - \mathbf{h}_t)^T\mathbf{h}_1 =& - g_0 \sum_{j=1}^{p} \text{sign}(h_{1,j}) K \mathbf{h}_{1,j} = -g_0 K ||\mathbf{h}_t||_1, \\
(\hat{\mathbf{w}}_1 - \mathbf{h}_t)^T\mathbf{h}_1 =& - g_0 \sum_{j=1}^{p} \text{sign}(h_{1,j}) \tilde{k}_{1,j} \mathbf{h}_{1,j} = -g_0 ||\tilde{\mathbf{k}}_1 \mathbf{h}_t||_1.
\end{aligned}$$

Thus in the second part of (3.20), we have

$$\begin{aligned}
&- 2\left[(\bar{\mathbf{w}}_1 - \mathbf{h}_1)^T(\mathbf{w}^* - \mathbf{h}_1) - (\hat{\mathbf{w}}_1 - \mathbf{h}_1)^T(\mathbf{w}^* - \mathbf{h}_1)\right] \\
=& \left[(\bar{\mathbf{w}}_1 - \mathbf{h}_1)^T\mathbf{h}_1 - (\hat{\mathbf{w}}_1 - \mathbf{h}_1)^T\mathbf{h}_1\right] - \left[(\bar{\mathbf{w}}_1 - \mathbf{h}_1)^T\mathbf{w}^* - (\hat{\mathbf{w}}_1 - \mathbf{h}_1)^T\mathbf{w}^*\right] \\
=& 2g_0 \left[\left(K\|\mathbf{h}_1\|_1 - \|\tilde{\mathbf{k}}_1\mathbf{h}_1\|_1\right) - (\bar{\mathbf{w}}_1 - \hat{\mathbf{w}}_1)^T\mathbf{w}^*\right]
\end{aligned} \tag{3.22}$$

Since $K \geq \tilde{k}_{1,j}$ for $j = 1, \ldots, p$, the first part of (3.22) is guaranteed to be nonnegative. In the second part of (3.22) , let $\hat{w}_{1,j} = \hat{v}_{1,j}\mathbb{1}(|h_{1,j}| > \tilde{k}_{1,j}g_0)$ and $\bar{w}_{1,j} = \bar{v}_{1,j}\mathbb{1}(|h_{1,j}| > Kg_0)$. Denote $\delta_t = L(w_{t-1}, z_t)$ to be the gradient of the loss function at a certain iteration. Since in this chapter we consider linear prediction model,

$$\delta_t = L'(\mathbf{w}_{t-1}, z_t) = G(f_{\mathbf{w_{t-1}}}(\mathbf{x}_t))\mathbf{x}_t.$$

Hence, $\delta_{t,j} = \delta_{t,j}\mathbb{1}(|x_{t,j}| > 0)$.

Again based on relationships in (3.18) and (3.19), we consider the following two scenarios:

- When $h_{1,j} > 0$:

$$\hat{v}_{1,j} = \sum_{t=1}^{K} \left[\mathbb{1}(|x_{t,j}| > 0)(\delta_{t,j} - g_0)\right],$$
$$\bar{v}_{1,j} = \sum_{t=1}^{K} \left[\mathbb{1}(|x_{t,j}| > 0)\delta_{t,j} - g_0\right].$$

  Thus,

$$\begin{aligned}
\bar{v}_{1,j} - \hat{v}_{1,j} &= -\sum_{t=1}^{K} \mathbb{1}(|x_{t,j}| = 0)g_0, \\
\bar{w}_{1,j} - \hat{w}_{1,j} &= -\sum_{t=1}^{K} \mathbb{1}(|x_{t,j}| = 0)g_0\mathbb{1}(h_{1,j} > Kg_0) - \sum_{t=1}^{K}(h_{1,j} - \tilde{k}_{1,j}g_0)\mathbb{1}(\tilde{k}_{1,j}g_0 < h_{1,j} < Kg_0) \\
&\leq -\sum_{t=1}^{K} \mathbb{1}(|x_{t,j}| = 0)g_0\mathbb{1}(h_{1,j} > Kg_0). \qquad (3.23)
\end{aligned}$$

- When $h_{1,j} < 0$, similarly, we get

$$\begin{aligned}
\bar{v}_{1,j} - \hat{v}_{1,j} &= \sum_{t=1}^{K} \mathbb{1}(|x_{t,j}| = 0)g_0, \\
\bar{w}_{1,j} - \hat{w}_{1,j} &= \sum_{t=1}^{K} \mathbb{1}(|x_{t,j}| = 0)g_0\mathbb{1}(h_{1,j} < -Kg_0) + \sum_{t=1}^{K}(h_{1,j} + \tilde{k}_{1,j}g_0)\mathbb{1}(-Kg_0 < h_{1,j} < -\tilde{k}_{1,j}g_0) \\
&\geq \sum_{t=1}^{K} \mathbb{1}(|x_{t,j}| = 0)g_0\mathbb{1}(h_{1,j} < -Kg_0). \qquad (3.24)
\end{aligned}$$

Hence, based on jointly we have

$$\begin{aligned}
&\mathbb{E}\left((\bar{\mathbf{w}}_1 - \hat{\mathbf{w}}_1)^T \mathbf{w}^*\right) \\
=&\sum_{j=1}^{p} \mathbb{E}((\bar{w}_{1,j} - \hat{w}_{1,j}) w_j^* | h_{1,j} > 0) \Pr(h_{1,j} > 0) + \mathbb{E}((\bar{w}_{1,j} - \hat{w}_{1,j}) w_j^* | h_{1,j} < 0) \Pr(h_{1,j} < 0) \\
\leq& - g_0 \sum_{t=1}^{K} \sum_{j=1}^{p} \mathbb{1}(|x_{t,j}| = 0) |w_j^*| \leq 0
\end{aligned}$$

The second inequality is derived from the condition that $\lambda_j = 0$ if $\mathbf{w}_j^* = 0$ and the following fact. Since $\mathbf{h}_1$ is the sum of K stochastic gradients descent steps and each one of the stochastic gradient which is an unbiased estimator of the true gradient [Bottou, 1998], we have

$$\mathbb{E}(\mathbb{1}(h_{1,j} < 0) w_j^*) < 0,$$
$$\mathbb{E}(\mathbb{1}(h_{1,j} > 0) w_j^*) > 0.$$

Hence, together with (3.21) and (3.22), we have

$$||\bar{\mathbf{w}}_1 - \mathbf{w}^*||^2 - ||\hat{\mathbf{w}}_1 - \mathbf{w}^*||^2 \geq 0.$$

□

In Lemma 2, we compare the distances towards the optimal weight vector $\mathbf{w}^*$ from 1) the weight vector with uniform gravity and 2) the weight vector with informative truncation that depends on the number of zero entries occurred in a burst. Such a gap suggests the effectiveness of informative truncation on sparse data in which feature sparsity is highly heterogeneous. In the scenarios that very few nonzero entries appear in a burst, the informative truncation imposes gravity that is proportional to the information presented in a burst. It is a fairer treatment than uniform truncation and leads to a large improvement in expectation. When features are all considerably dense in a burst, the informative truncation is equivalent to the uniform truncation.

In short, Lemma 1 demonstrates the improvement in expected squared error due to stabilization on the weight vector. Lemma 2, on the other hand, quantifies the

improvements in reduce truncation bias when implementing informative truncation on sparse features with heterogeneous sparsity levels.

Given Lemma 1 and Lemma 2, we have the expected regret bound of the proposed Algorithm 3.4 in Theorem 1.

Theorem 3.1. *Consider the updating rules for the weight vector in Algorithm 3. On an arbitrary path, with* $\underline{\mathbf{w}}_0 = 0$ *and* $\eta > 0$*, let* $\{\underline{\mathbf{w}}_t\}_{t=1}^T$ *be the resulted weight vector and* $\left\{\underline{g}_t\right\}_{t=1}^T$ *be the gravity values applied to the weight vectors generated by Algorithm* **??***, along with the base gravity parameters* $\left\{\underline{g}_{0,t}\right\}_{t=1}^T$*. Let* $\mathbf{w}^* \in \mathbb{R}^p$ *be the target sparse weight vector with* $d^* = ||w^*||_0$*. Set the purging threshold* π_0 *to be sufficiently large such that* $\pi_0 \in \left(\frac{1}{2} + \frac{(q^{\mathbf{g}})^2}{2p(q^{\mathbf{g}} - d^*)}, 1\right)$*. If Assumption 1, 2, and 3 hold, then there exists a sequence of* $\varepsilon_t \in \left(0, \frac{\pi_0 C(p - |\hat{\Omega}_t|)}{|\hat{\Omega}_t|}\right)$ *at each stability selection with the set of stable features* $\hat{\Omega}_t$ *such that*

$$\begin{aligned}
&\mathbb{E}\left(\sum_{t=1}^T \left[L(\underline{\mathbf{w}}_t, z_t) + K\underline{g}_t||\underline{\mathbf{w}}_t||_1\right] - \sum_{t=1}^T \left[L(\mathbf{w}^*, z_t) + K\underline{g}_t||\mathbf{w}^*||_1\right]\right) \\
\leq &\frac{\eta A}{2 - \eta A}\left(\mathbb{E}\left[\sum_{t=1}^T L(\mathbf{w}^*, z_t) + K\underline{g}_{0,t}(||\mathbf{w}^*||_1 - ||\underline{\mathbf{w}}_t||_1)\right]\right) + \frac{1}{2 - \eta A}\left(\eta TB + \frac{1}{\eta}||\mathbf{w}^*||^2\right) \\
&- \frac{1}{2\eta - \eta^2 A}\sum_{t=1}^T \varepsilon_t^2 \mathbb{1}\left(\frac{t}{Kn_K} \in \mathbb{Z}\right)(|\hat{S}_{\varepsilon_t,t}| - |\hat{\Omega}_t|) \qquad (3.25)
\end{aligned}$$

where $\hat{S}_{\varepsilon,t} = \{j : \mathbb{E}(|\hat{w}_{t,j}|) > \varepsilon_t\}$ *and* $\hat{\mathbf{w}}_t$ *is the weight vector at time* t *before stabilization.*

Proof. At a given time t when truncation is performed, let $\mathbf{h}_t = \mathbf{w}_{t-1} - \eta\nabla_{\mathbf{w}_{t-1}} L(\mathbf{w}_{t-1}, z)$ and let $\hat{\mathbf{w}}_t$ be the truncated but non-stabilized weight vector based on the truncation in (3.5) that $\hat{\mathbf{w}}_t = T(\mathbf{h}_t, g_{0,t}\tilde{\mathbf{k}}_t)$ with the base gravity $g_{0,t}$. Let $\hat{\Omega}$ be the current set of stable features and the stabilized weight vector is obtained as $\tilde{\mathbf{w}}_t = \hat{\mathbf{w}}_t \mathbb{1}_{\hat{\Omega}}$.

Firstly, we have

$$\begin{aligned}
||\hat{\mathbf{w}}_t - \mathbf{w}^*|| &\leq ||\mathbf{w}^* - \hat{\mathbf{w}}_t||^2 + ||\hat{\mathbf{w}}_t - \mathbf{h}_t||^2 \qquad (3.26) \\
&= ||\mathbf{w}^* - \mathbf{h}_t||^2 - 2(\mathbf{w}^* - \hat{\mathbf{w}}_t)^T(\hat{\mathbf{w}}_t - \mathbf{h}_t).
\end{aligned}$$

In the first part of (3.26),

$$\begin{aligned}||\mathbf{w}^* - \mathbf{h}_t||^2 =& ||\mathbf{w}^* - \mathbf{w}_{t-1} + \mathbf{w}_{t-1} - \mathbf{h}_t||^2 \\ =& ||\mathbf{w}^* - \mathbf{w}_{t-1}||^2 + ||\mathbf{w}_{t-1} - \mathbf{h}_t||^2 + 2(\mathbf{w}^* - \mathbf{w}_{t-1})^T(\mathbf{w}_{t-1} - \mathbf{h}_t) \\ =& ||\mathbf{w}^* - \mathbf{w}_{t-1}||^2 + ||\mathbf{w}_{t-1} - \eta\nabla_{\mathbf{w}_{t-1}}L(\mathbf{w}_{t-1}, z) - \mathbf{w}_{t-1}||^2 \\ &+ 2(\mathbf{w}^* - \mathbf{w}_{t-1})^T(\mathbf{w}_{t-1} - \mathbf{w}_{t-1} + \eta\nabla_{\mathbf{w}_{t-1}}L(\mathbf{w}_{t-1}, z)) \\ =& ||\mathbf{w}^* - \mathbf{w}_{t-1}||^2 + \eta^2||\nabla_{\mathbf{w}_{t-1}}L(\mathbf{w}_{t-1}, z)||^2 + 2\eta(\mathbf{w}^* - \mathbf{w}_{t-1})^T\nabla_{\mathbf{w}_{t-1}}L(\mathbf{w}_{t-1}, z).\end{aligned} \tag{3.27}$$

Since $L(w, z)$ is convex,

$$(\mathbf{w}^* - \mathbf{w}_{t-1})^T\nabla_{\mathbf{w}_{t-1}}L(\mathbf{w}_{t-1}, z) \leq L(\mathbf{w}^*, z) - L(\mathbf{w}_{t-1}, z).$$

And based on Assumption 1,

$$||\nabla_{\mathbf{w}_{t-1}}L(\mathbf{w}_{t-1}, z)||^2 \leq AL(\mathbf{w}_{t-1}, z) + B.$$

Thus (3.27) has the upper bound

$$||\mathbf{w}^* - \mathbf{h}_t||^2 \leq ||\mathbf{w}^* - \mathbf{w}_{t-1}||^2 + \eta^2(AL(\mathbf{w}_{t-1}, z) + B) + 2\eta\left[L(\mathbf{w}^*, z) - L(\mathbf{w}_{t-1}, z)\right].$$

In the second part of (3.26), since $\hat{\mathbf{w}}_t = T(\mathbf{h}_t, g_t)$, $\hat{\mathbf{w}}_t = \text{sign}(\mathbf{h}_t)\max(|\mathbf{h}_t| - g_{0,t}\tilde{\mathbf{k}}_t, 0)$ and thus

$$(\hat{\mathbf{w}}_t - \mathbf{h}_t)^T\hat{\mathbf{w}}_t = -g_{0,t}\sum_{j=1}^{p}\text{sign}(h_{t,j})\tilde{k}_{t,j}\hat{\mathbf{w}}_{t,j} = -g_{0,t}||\tilde{\mathbf{k}}_t\hat{\mathbf{w}}_t||_1.$$

Then

$$\begin{aligned}-(\mathbf{w}^* - \hat{\mathbf{w}}_t)^T(\hat{\mathbf{w}}_t - \mathbf{h}_t) &= -(\mathbf{w}^*)^T(\hat{\mathbf{w}}_t - \mathbf{h}_t) + \hat{\mathbf{w}}_t^T(\hat{\mathbf{w}}_t - \mathbf{h}_t) \\ &\leq -\sum_{j=1}^{p}|w_j^*||\hat{w}_{t,j} - h_{t,j}| - g_{0,t}||\tilde{\mathbf{k}}_t\hat{\mathbf{w}}_t||_1 \\ &\leq -g_{0,t}\sum_{j=1}^{p}|\tilde{k}_{t,j}w_j^*| - g_{0,t}||\tilde{\mathbf{k}}_t\hat{\mathbf{w}}_t||_1 \\ &\leq -Kg_{0,t}\sum_{j=1}^{p}|w_j^*| - Kg_{0,t}||\hat{\mathbf{w}}_t||_1 \\ &= -Kg_{0,t}\left(||\mathbf{w}^*||_1 - ||\hat{\mathbf{w}}_t||_1\right).\end{aligned}$$

The second last inequality is due to $\tilde{k}_{j,t} \leq K$ for all $j = 1, \ldots, p$. Thus, in total, equation (3.26) has the upper bound

$$||\hat{\mathbf{w}}_t - \mathbf{w}^*||^2 \leq ||\mathbf{w}^* - \mathbf{w}_{t-1}||^2 + \eta^2(AL(\mathbf{w}_{t-1}, z) + B) + 2\eta\left[L(\mathbf{w}^*, z) - L(\mathbf{w}_{t-1}, z)\right] + 2Kg_{0,t}\left(||\mathbf{w}^*||_1 - ||\hat{\mathbf{w}}_t||_1\right). \quad (3.28)$$

On the other hand, from Lemma 1 we have

$$\mathbb{E}\left(||\hat{\mathbf{w}}_t - \mathbf{w}^*||^2 - ||\tilde{\mathbf{w}}_t - \mathbf{w}^*||^2\right) \geq \varepsilon^2(|\hat{S}_\varepsilon| - |\hat{\Omega}|) + 2\pi_0 C^2\left[\left(1 - \frac{q^{\mathbf{g_t}}}{2\pi_0 p - p}\right) q^{\mathbf{g}_t} - d\right], \quad (3.29)$$

where $\mathbf{g}_t = g_{0,t}\tilde{\mathbf{k}}_t$. Since $\pi_0 \in \left(\frac{1}{2} + \frac{(q^{\mathbf{g}})^2}{2p(q^{\mathbf{g}} - d^*)}, 1\right)$ and, by Assumption 3, $q^{\mathbf{g}} > d$ for $t = 1, \ldots, T$, we have

$$2\pi_0 C^2\left[\left(1 - \frac{q^{\mathbf{g}_t}}{2\pi_0 p - p}\right) q^{\mathbf{g}_t} - d\right] \geq 0, \quad \text{for } t = 1, \ldots, T.$$

Thus, we can further write (3.29) as

$$\mathbb{E}\left(||\tilde{\mathbf{w}}_t - \mathbf{w}^*||^2\right) \leq \mathbb{E}\left(||\hat{\mathbf{w}}_t - \mathbf{w}^*||^2\right) - \varepsilon^2(|\hat{S}_\varepsilon| - |\hat{\Omega}|). \quad (3.30)$$

By rearranging (3.30) and combining it with (3.28), we get

$$\begin{aligned}
&\mathbb{E}\left(\left(1 - \frac{1}{2}\eta A\right) L(\mathbf{w}_{t-1}, z_t) + \frac{Kg_{0,t}}{\eta}||\mathbf{w}_{t-1}||_1\right) \\
\leq &\mathbb{E}\left(L(\mathbf{w}^*, z_t) + \frac{Kg_{0,t}}{\eta}||\mathbf{w}^*||_1\right) + \frac{1}{2\eta}\mathbb{E}\left(||\mathbf{w}^* - \mathbf{w}_{t-1}||^2 - ||\mathbf{w}^* - \tilde{\mathbf{w}}_t||^2\right) + \frac{\eta B}{2} \\
&- \frac{\varepsilon^2}{2\eta}(|\hat{S}_{\varepsilon_t}| - |\hat{\Omega}|). \quad (3.31)
\end{aligned}$$

Based on Algorithm 3, let $\underline{\mathbf{w}}_t$ be the output weight vector at the end of time t and let $\underline{g}_t$ be the applied gravity parameter at time t, where

$$\underline{g}_{0,t} = \begin{cases} \frac{g_{0,t}}{\eta}, & \text{if } t/K \text{ is an integer,} \\ 0, & \text{otherwise.} \end{cases},$$

$$\underline{g}_t = \begin{cases} \tilde{\mathbf{k}}\frac{g_{0,t}}{\eta}, & \text{if } t/K \text{ is an integer,} \\ 0, & \text{otherwise.} \end{cases},$$

and,

$$\underline{\mathbf{w}}_t = \begin{cases} \tilde{\mathbf{w}}_t, & \text{if } t/(Kn_K) \text{ is an integer,} \\ \hat{\mathbf{w}}_t, & \text{otherwise.} \end{cases}.$$

By initializing the weight vector as a vector of zeros of length p, we sum up (3.31) over $t = 1, \ldots, T$ with telescoping to obtain the following result:

$$\begin{aligned}
&\left(1 - \frac{1}{2}\eta A\right) \mathbb{E}\left(\sum_{t=1}^{T}\left[L(\underline{\mathbf{w}}_t, z_t) + \frac{K\underline{g}_{0,t}}{1 - 1/2\eta A}||\underline{\mathbf{w}}_t||_1\right]\right) \\
\leq &\mathbb{E}\left(\sum_{t=1}^{T} L(\mathbf{w}^*, z_t) + K\underline{g}_{0,t}||\mathbf{w}^*||_1\right) + \frac{1}{2\eta}\mathbb{E}\left(||\mathbf{w}^* - \underline{\mathbf{w}}_1||^2 - ||\mathbf{w}^* - \underline{\mathbf{w}}_T||^2\right) + \frac{\eta BT}{2} \\
&- \frac{1}{2\eta}\sum_{t=1}^{T} \varepsilon_t^2 \mathbb{1}(\frac{t}{Kn_K} \in \mathbb{Z})(|\hat{S}_{\varepsilon,t}| - |\hat{\Omega}_t|) \\
\leq &\mathbb{E}\left(\sum_{t=1}^{T} L(\mathbf{w}^*, z_t) + K\underline{g}_{0,t}||\mathbf{w}^*||_1\right) + \frac{||\mathbf{w}^*||^2}{2\eta} + \frac{\eta BT}{2} \\
&- \frac{1}{2\eta}\sum_{t=1}^{T} \varepsilon_t^2 \mathbb{1}(\frac{t}{Kn_K} \in \mathbb{Z})(|\hat{S}_{\varepsilon,t}| - |\hat{\Omega}_t|). \qquad (3.32)
\end{aligned}$$

The theorem follows by rearrange terms in (3.32) and that $\tilde{k}_{t,j} \leq K$ for $j = 1, \ldots, p$ and $t = 1, \ldots, T$.

□

In the result of Theorem 1, the first two parts of the right-hand-side of the expected regret bound (3.25) is similar to the bound obtained in [Langford *et al.*, 2009]. It implies the trade-off between attained sparsity in the resulted weight vector and the regret performance. When the applied gravity is small under the joint effect of the base gravity g_0 and the size of each burst K, the sparsity is less but the expected regret bound is lower. On the other hand, when the applied gravity is large, the resulted weight vector is more sparse but at the risk of higher regret. Based on Lemma 1, the proposed algorithm is guaranteed to achieve lower regret bound in expectation when the target weight vector is highly sparse. As quantified in the third term of the right-hand-side of (3.25), the improvement comes from the reduction of the active

set at each purging. By its virtue, noisy features are removed from the set of stable features and thus are absent in later SGD updates and truncations.

Theorem 1 is stated with a constant learning rate η. It is possible to obtain a lower regret bound in expectation with adaptive learning rate η_t decaying with t, such as $\eta_t = \frac{1}{\sqrt{t}}$, which is commonly used in the literature of online learning and stochastic optimization. However, the discussion of using an varying learning rate is not a main focus of this paper and adds extra complexity of the analysis. Without knowing T in advance, this may lead to a no-regret bound as suggested in [Langford *et al.*, 2009]. Instead, in Corollary 1, we show that the convergence rate of the proposed algorithm is $O(\sqrt{T})$ with $\eta = \frac{1}{\sqrt{T}}$.

Corollary 1. *Assume that all conditions of Theorem 1 are satisfied. Let the learning rate η be $\frac{1}{\sqrt{T}}$. The upper bound of the expected regret is*

$$\mathbb{E}\left(\sum_{t=1}^{T}\left[L(\underline{\mathbf{w}}_t, z_t) + \underline{g}_t||\underline{\mathbf{w}}_t||_1\right] - \sum_{t=1}^{T}\left[L(\mathbf{w}^*, z_t) + \underline{g}_t||\mathbf{w}^*||_1\right]\right) \leq O(\sqrt{T}),$$

where $\underline{g}_t = K\underline{g}_{0,t}$.

Proof. By plugging in $\eta = \frac{1}{\sqrt{T}}$ to the result from Theorem 1, we get

$$\begin{aligned}
&\mathbb{E}\left(\sum_{t=1}^{T}\left[L(\underline{\mathbf{w}}_t, z_t) + \underline{g}_t||\underline{\mathbf{w}}_t||_1\right] - \sum_{t=1}^{T}\left[L(\mathbf{w}^*, z_t) + \underline{g}_t||\mathbf{w}^*||_1\right]\right) \\
\leq &\frac{A}{2\sqrt{T} - A}\left(\mathbb{E}\left[\sum_{t=1}^{T} L(\mathbf{w}^*, z_t) + \underline{g}_t(||\mathbf{w}^*||_1 - ||\underline{\mathbf{w}}_t||_1\right]\right) \\
&+ \frac{T}{2\sqrt{T} - A}\left(\eta TB + \frac{1}{\eta}||\mathbf{w}^*||^2\right) - \frac{T}{2\sqrt{T} - A}\sum_{t=1}^{T}\varepsilon_t^2 \mathbb{1}(\frac{t}{Kn_K} \in \mathbb{Z})(|\hat{S}_{\varepsilon,t}| - |\hat{\Omega}_t|).
\end{aligned}$$

The result is then straightforward. □

Assume that the input features have d nonzero entries on average. With linear prediction model $f_{\mathbf{w}}(x) = \mathbf{w}^T x$, the computational complexity at each iteration is

$O(d)$. Leveraging the sparse structure, the informative truncation only requires an additional $O(Kd)$ space for recording the counters. The purging process of stability selection consumes $O(\delta)$, $\delta = Kn_KMd$, space for storing the generated intermediate weight vectors and $O(\delta \log(\delta))$ computational complexity. Both storage and computational cost decrease when the set of stable features diminishes as the algorithm proceeds. Since the parameters K, n_K, and M is normally set to be small values, the complexity mostly depends on $O(d \log(d))$. In summary, the proposed algorithm scales with the number of nonzero entries instead of the total dimensions, making it appealing to high-dimensional applications.

3.5 Practical Remarks

When implementing Algorithm 3.4 in practice, the performance can be further improved in terms of both accuracy and computational efficiency by employing a couple of practical techniques. It includes applying *informative purging* and attenuating the truncation frequency to achieve more accurate sparse learning and steadier convergence.

The first improvement can be implemented by better addressing the issue of scarcity of incoming samples. For computing selection probabilities, instead of using only information from the current stage, we can inherit information from previous stages for features that are too scarce to accumulate enough updates during one stage. Specifically, we introduce an *accumulated counter* κ_s at stage s as the total number of times that a feature is updated within a burst during this stage:

$$\kappa_{j,s} = \sum_{m=1}^{M} \sum_{\tau=1}^{n_K} \tilde{k}_{j,\tau}^{(m)}, \quad j = 1, \ldots, p,$$

which is essentially the denominator of the selection probability in (3.9). Similarly, we define an *accumulated truncation indicator* $\mathbf{b}_s$ at stage s as the total number of

times that a feature is truncated to zero given valid update(s):

$$\mathbf{b}_{j,s} = \sum_{m=1}^{M} \sum_{\tau:\tilde{k}_{j,\tau}^{(m)}>0} \mathbb{1}(|\hat{\mathbf{w}}_{j,\tau}^{(m)}| > 0), \quad j = 1, \ldots, p.$$

A feature is then evaluated in the stability selection *only* if there are enough updates from the present stage and from any unused information carried over from previous stages. Given a threshold $\delta_K \geq 0$, let $\tilde{\kappa}_{j,s} \triangleq \tilde{\kappa}_{j,s-1}\mathbb{1}(\tilde{\kappa}_{j,s-1} < \delta_K) + \kappa_{j,s}$ and $\tilde{\mathbf{b}}_{j,s} \triangleq \tilde{\mathbf{b}}_{j,s-1}\mathbb{1}(\tilde{\kappa}_{j,s-1} < \delta_K) + \mathbf{b}_{j,s}$. The selection probability is modified as

$$\hat{\Pi}_{j,s} = \begin{cases} \frac{\tilde{\mathbf{b}}_{j,s}}{\tilde{\kappa}_s}, & \text{for } j \text{ s.t.} \tilde{\kappa}_s > \delta_K \\ 1, & \text{otherwise} \end{cases}, \quad \text{for } j = 1, \ldots, p. \tag{3.33}$$

This strategy extends the key idea in Section 3.3.1 that, with sparse data, each decision need to be based on sufficient evidence. Using the "carried-over" information allows the algorithm to utilize information available in a sequence of SGD updates while attuned to the needs of features with different levels of sparsity. In practice, this modification facilitates faster convergence especially for ultra-sparse data.

The second practical strategy is that the size of each burst, K, can be adaptively adjusted in a similar fashion as the rejection rate β in (3.14). At the end of each stage, the burst size K_s is updated as

$$K_s = \left\lceil K_0 \log\left(\frac{1}{\alpha d_{s-1}}\right)\right\rceil,$$

where $K_0 > 0$ is the initial burst size and, as in (3.14), $d_s = \frac{|\hat{\Omega}_s|}{p}$. The tuning parameter $\alpha > 0$ adjusts the annealing rate of the truncation frequency. Although the result in Theorem 1 is based on a fixed K, it can be easily shown that the same upper bound can also be attained with an increasing K_s. By increasing K in the later stage of the algorithm, when the majority of irrelevant features have been removed from the stable set, the chance of erroneous truncation is reduced. Such scheme further steers the algorithm from the mode of exploring potential sparse combination of features in the early stage toward the fine tuning of the weight vector by exploiting information

from more samples in a sequence. It also facilitates faster convergence as the size of the stable set approaches to a sufficiently small number, as the algorithm converges to the standard stochastic gradient descent approximately.

3.6 Results

In this section, we present experimental results evaluating the performance of the proposed stabilized truncated SGD algorithm in high-dimensional classification problems with sparsity regularization. In this paper, we focus on linear prediction model for binary classification where $f_{\mathbf{w}}(x) = \mathbf{w}^T x$ and $\hat{y} = \text{sign}(f_w(x))$ with the observed class label $y \in \{-1, 1\}$. We consider two commonly used convex loss functions in machine learning tasks that both satisfy Assumption 1:

- Hinge loss: $l(f, y) = \max(1 - fy, 0)$
- Logistic loss: $l(f, y) = \log(1 + \exp(-fy))$

Using five datasets from different domains, the performance of our algorithm and other algorithms for comparison are evaluated on classification performance and feature selection stability and sparsity. We first define measure of feature stability in Section 3.6.1.

3.6.1 Feature Selection Stability

The goal of sparse learning is to select a subset of truly informative features with stabilized estimation variance as well as increased classification accuracy and model interpretability. Subgradient-based online learning methods depend heavily by the random ordering of samples on which they are fed to the algorithm. Such dependence leads to much deteriorated performance when it comes to high-dimensional sparse inputs. For a particular feature, the positions of its nonzero occurrences in a random ordering of samples greatly affect its learning outcome, in terms of learnt weight and

membership in the set of selected features. Therefore, in addition to attaining a low generalization error, a desirable sparse online learning method should also produce an informative feature subset that is stable and robust to random permutations of input data. To evaluate feature selection stability of subgradient-based sparse learning methods, we define in the following a numerical measure of similarity between selected feature subsets resulted from different random permutations of data. Given an output weight vector $\underline{\mathbf{w}}$ from a subgradient-based algorithm with input data $\mathcal{D}$, similarly as in (3.7), we denote the selected feature subset as

$$\mathcal{S}(\underline{\mathbf{w}}; \mathcal{D}) = \{j : |\underline{w}_j| > 0, \underline{\mathbf{w}} = \Psi(\mathcal{D}\}.$$

Given two random permutations of the training data $\mathcal{D}$, $\mathcal{D}^{(1)}$ and $\mathcal{D}^{(2)}$, the similarity between the two sets of selected feature subsets $\mathcal{S}_1 = \mathcal{S}(\underline{\mathbf{w}}_1; \mathcal{D}^{(1)})$ and $\mathcal{S}_2 = \mathcal{S}(\underline{\mathbf{w}}_2; \mathcal{D}^{(2)})$ is measured by the Cohen's kappa coefficient [Cohen, 1960],

$$\kappa(\mathcal{S}_1, \mathcal{S}_2) = \frac{q_o - q_e}{1 - q_e},$$

where q_o is the *relative observed agreement* between $\mathcal{S}_1$ and $\mathcal{S}_2$:

$$q_o = \frac{p_{11} + p_{22}}{p},$$

and q_e is the *hypothetical probability of change agreement*: $\mathcal{S}_1$ and $\mathcal{S}_2$

$$q_e = \frac{(p_{11} + p_{12})(p_{11} + p_{21})}{p^2} + \frac{(p_{12} + p_{22})(p_{21} + p_{22})}{p^2}$$

with $p_{11} = |\mathcal{S}_1 \cap \mathcal{S}_2|$, $p_{12} = |\mathcal{S}_1 \cap \mathcal{S}_2^C|$, $p_{21} = |\mathcal{S}_1^C \cap \mathcal{S}_2|$, $p_{22} = |\mathcal{S}_1^C \cap \mathcal{S}_2^C|$, and $p = p_{11} + p_{12} + p_{21} + p_{22}$ is the size of variable pool.

Note that $\kappa(\mathcal{S}_1, \mathcal{S}_2) \in [-1, 1]$, where $\kappa(\mathcal{S}_1, \mathcal{S}_2) = 1$ if $\mathcal{S}_1$ and $\mathcal{S}_2$ completely overlap with each other and $\kappa(\mathcal{S}_1, \mathcal{S}_2) = -1$ when $\mathcal{S}_1$ and $\mathcal{S}_2$ are in complete disagreement with $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$ and $\mathcal{S}_1^C \cap \mathcal{S}_2^C = \emptyset$.

Based on Cohen's kappa coefficient, we define the measure of feature selection stability of $\mathcal{S}(\underline{\mathbf{w}}; \cdot)$ returned by a procedure Ψ using randomly ordered data $\mathcal{D}^{(1)}$ and $\mathcal{D}^{(2)}$ as

$$s(\Psi) = \mathbb{E}_{\mathcal{D}^{(1)}, \mathcal{D}^{(2)}} \left[\kappa(\mathcal{S}(\underline{\mathbf{w}}_1; \mathcal{D}^{(1)}), \mathcal{S}(\underline{\mathbf{w}}_2; \mathcal{D}^{(2)}) \right],$$

which is motivated by [Sun *et al.*, 2013].

In practice, we use the empirical average $\hat{s}(\Psi)$ over B random permutations of the training data to measure the stability of the a subgradient-based online learning algorithm $\Psi(\cdot)$:

$$\hat{s}(\Psi) = \frac{1}{B(B-1)} \sum_{i=1}^{D} \sum_{j \neq i} \left[\kappa(\mathcal{S}\underline{\mathbf{w}}_i; \mathcal{D}^{(i)}), \mathcal{S}(\underline{\mathbf{w}}_j; \mathcal{D}^{(j)}) \right]. \tag{3.34}$$

3.6.2 Experiment Setup

We evaluate the performance of our algorithm on several real-world classification datasets with up to $100,000$ features. These datasets have different levels of sparsity with various sample sizes. The information of experiment datasets are summarized in Table 3.1. The first four datasets were constructed for NIPS 2003 Feature Selection Challenge[3] [Guyon *et al.*, 2004], which were preprocessed with added "probes" as random features distributed similarly to the real features. Thus, a good performance does not only lie in low generalization error rates, but also in sparse weight vectors that identify the truly important features. Reuters CV1 (RCV1) is a popular text classification dataset with a bag-of-words representation. We use the binary version from the LIBSVM dataset collection[4] introduced in [Cai and He, 2012]. We create the training and validation set using a 70-30 random splits. All datasets are normalized such that each feature has variance 1.

[3]Data source: `https://archive.ics.uci.edu/ml/index.html`

[4]Data source: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

Dataset	Domain	Dimensions	Data Density (Sparsity)	Training Size	Validation Size
RCV1	Text Mining	47,236	0.16%	14169	6073
Dexter	Text Mining	20,000	0.48%	300	300
Dorothea	Drug Discovery	100,000	0.91%	800	350
Gisette	Digits Recognition	5000	13%	6000	1000
Arcene	Mass-Spectrometry	10,000	50%	100	100

Table 3.1: Datasets used in the numerical experiments. Sparsity is defined as the average feature sparsity levels, which is the column-wise average percentage of nonzero entries, in the training data.

We compare the proposed algorithm with the standard stochastic gradient descent algorithm [Bottou, 1998] and another three other sparsity-inducing stochastic methods, including the truncated gradient algorithm [Langford *et al.*, 2009], the Regularized Dual Averaging (RDA) algorithm [Xiao, 2009], and the forward backward splitting (FOBOS) algorithm [Duchi and Singer, 2009]. The RDA algorithm updates the weight vector at each step based on a running average $\bar{\mathbf{g}}_t$ of all subgradients $\{\mathbf{g}_\tau = L'(\mathbf{w}_\tau, z_\tau) \in \partial_{\mathbf{w}_\tau} L(\mathbf{w}_\tau, z_\tau), \tau = 1, \ldots, t\}$ in previous iterations as

$$\bar{\mathbf{g}}_t = \frac{t-1}{t}\mathbf{g}_{t-1} + \frac{1}{t}\mathbf{g}_t$$

Given the average subgradient , the next weight vector is computed by solving the minimization problem

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left\{ \frac{1}{t}\sum_{\tau}^{t} \langle L'(\mathbf{w}_t, z_t), \mathbf{w}\rangle + \Psi(\mathbf{w}) + \frac{\beta_t}{t} h(\mathbf{w}) \right\}, \tag{3.35}$$

where $\Psi(\mathbf{w})$ is the regularizer, $h(w)$ is an auxiliary strongly convex function, and $\{\beta_t\}_{t\geq 1}$ is a nonnegative and nondecreasing input sequence, which determines the convergence properties of the algorithm.

In the context of L_1 regularization, the RDA algorithm is derived by setting $\Psi(\mathbf{w}) = \lambda||\mathbf{w}||_1$, $\beta_t = \gamma\sqrt{t}$, and replacing $h(w)$ with a parametrized version:

$$h_\rho = \frac{1}{2}||\mathbf{w}||_2^2 + \rho||\mathbf{w}||_1,$$

where $\rho \geq 0$ is a sparsity-enhancing parameter. Hence, the minimization problem in (3.35) has a explicit solution as, for $j = 1, \ldots, p$,

$$\mathbf{w}_{t+1}^{(j)} = \begin{cases} 0, & \text{if } |\bar{\mathbf{g}}_t^{(j)}| \leq \lambda_t^{\text{RDA}} \\ -\frac{\sqrt{t}}{\gamma}\left(\bar{\mathbf{g}}_t^{(j)} - \lambda_t^{\text{RDA}}\text{sgn}(\bar{\mathbf{g}}_t^{(j)})\right), & \text{otherwise,} \end{cases}$$

which is equivalent to

$$\mathbf{w}_{t+1} = T(\bar{\mathbf{g}}_t, \lambda_t^{\text{RDA}}),$$

where $\lambda_t^{\text{RDA}} = \lambda + \gamma\rho/\sqrt{t}$.

The FOBOS algorithm alternates between two phases. On each iteration, it first perform an unconstrained gradient descent step as in the standard SGD algorithm, whose output is denoted as $\mathbf{w}_{t+\frac{1}{2}}$. Then it cast and solve an instantaneous optimization problems that trades off between minimization of a regularization term and a close proximity to the result of the first phase:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \left\{\frac{1}{2}||\mathbf{w} - \mathbf{w}_{t+\frac{1}{2}}||^2 + \eta_{t+\frac{1}{2}}\Psi(\mathbf{w})\right\}, \tag{3.36}$$

where the regularization function is scaled by an interim step size $\eta_{t+\frac{1}{2}}$.

With L_1 regularization where $\Psi(w) = \lambda||\mathbf{w}||_1$, the second-phase update can be computed as

$$\mathbf{w}_{t+1} = T(\mathbf{w}_{t+\frac{1}{2}}, \eta_{t+\frac{1}{2}}\lambda).$$

To evaluate the stability of resulted weight vectors, we randomly permute the indices of the training samples for $B = 50$ times to produce stochastic samples that are fed to the algorithms. Such randomization helps identify the instability in learning results in terms of both error rate and the selected features.

For implementing all five stochastic methods, we allow the algorithms to run on the training data for $\{5, 10, 20, 30, 40, 50, 60\}$ passes. In the standard stochastic

gradient descent algorithm, we choose the optimal learning rate between 0.1 to 0.5. The base gravity parameter in the truncated gradient algorithm is chosen from the range $[0.001, 0.01]$. For RDA, we follow the suggestions in [Xiao, 2009] which sets $\gamma = 5000$ and $\rho = 0.005$ (effectively $\gamma\rho = 25$). We tune the parameter λ from the set of values $\{5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$ for both RDA and FOBOS. As instructed in [Duchi and Singer, 2009], η_t is set to be $\frac{1}{\sqrt{t}}$ in FOBOS. For the proposed stabilized truncated SGD algorithm, the size of each burst (the truncation frequency) is fixed at $K = 5$ and $n_K = 5$ for all datasets. In the proposed algorithm, we initiate the rejection rate at $\beta_0 = 0.7$ with annealing rate chosen from $\{-7, -5, -3, -1, 0, 1, 3\}$. The stability selection threshold π_0 is tuned within the range $[0.5, 0.6, \ldots, 0.9]$. For multi-thread implementation in Algorithm 3.4, $M = 16$ is used running in parallel on a high-performance computing cluster.

3.6.3 Results

3.6.3.1 Classification Performance

As shown in Table 3.2, the proposed algorithm shows improvement over the truncated gradient algorithm over all datasets and has better performances than RDA and FOBOS in most of the experiments. As data density increases, the truncated gradient algorithm performs better with hinge loss than with logistic loss. With hinge loss, the algorithm updates the weight vector only for samples within a small margin from the boundary. With logistic loss, the algorithm updates with continuous increments for all incoming samples and favors dense features in sparse learning. With highly sparse samples sequentially feed to the algorithm, the truncation in every K iterations is conducted with insufficient information about the true gradient due to the lack of nonzero entries in sparse features. Truncation with logistic loss leads to over selection of dense features and overfitting. The other two sparsity-inducing methods for comparison, RDA and FOBOS, also have large fluctuations across data sets with different levels of sparsity over these two loss functions, especially when data is highly

sparse. In comparison, the performance of the proposed algorithm is consistent for various dimensions and sparsity levels. Our algorithm is also shown to be robust to different choices of loss function. The comparatively lower test errors, especially for highly sparse data, mainly owes to the proposed algorithm's fairer treatments of features with heterogeneous sparsity.

From Table 3.2 we also observe substantially lower variances in test errors under random permutations of samples. The proposed algorithm has the lowest standard deviations of test errors across all datasets and under both loss functions. Such an improvement over truncated gradient algorithm and other comparing methods comes from both informative truncation with adaptive gravity and stability selection. Based on these selection probabilities we carry out feature purging. Our selection probability is computed based on multiple bursts whose truncation is guided by fair amounts of shrinkage. Hence, the removal decisions of features from the set of stable variables are grounded in reliable information on feature importance, which is more likely to be shared across different permutations of the data. The accumulations of sufficient information for all features help the proposed algorithm to be robust to random fluctuations in online setting.

3.6.3.2 Feature Selection and Sparsity

As far as sparse learning for feature selection goes, the proposed algorithm, as shown in Table 3.3, is the only method that delivers sparse weight vectors across five example datasets with various sparsity levels. It is shown in Table 3.3 that the resulted weight vectors are far more sparse than the truncated gradient algorithm in all datasets with both loss functions. When comparing these two algorithms, the distinction in attained sparsity levels is particularly striking with logistic loss function and when the sparsity level of the input data is low, such as the case of the Arcene dataset. These results also indicate that the truncate gradient algorithm fails to extract significant features and to obtain sparse solutions, which motivates the development of

techniques discussed in this paper. On the other hand, the variances in the percentage of nonzero features of the proposed algorithm are reduced by approximately a magnitude of 10. The contribution of stabilization is demonstrated again in terms of feature selection. Although RDA achieves very low sparsity in Dorothea, such a behavior is not observed in other datasets. It is shown to result in particularly denser weight vector with highly sparse data, such as RCV1, indicating its weakness in identifying truly informative features when information is scarce. FOBOS demonstrates overall poorer performance in terms of inducing sparse weight vector as compared to RDA and the proposed algorithm. Similar to its performance in terms of test error, the proposed algorithm delivers highly stable results in feature selection regardless of the choice of loss function and of random permutations of the data. We also achieve sufficient sparse result owing to stability selection which prevent noisy features from adding back to the stable set of variables in the online setting. On the other hand, the high sparsity in weight vector does not overshadow the generalizability performance as the informative truncation with unbiased shrinkage underlies a better estimation of the selection probability for the construction of the set of stable variables.

We further compare the data sparsity levels of the features selected by both sparse online learning algorithms. In Figure 3.3, using the Dorothea dataset as an example, we show the fraction of selected features at different data density levels. It is shown that truncated gradient algorithm is more likely to select dense features over sparse ones with both loss functions. The majority of the sparse features are truncated to zero regardless of their importance. Moreover, the fraction of selected feature exhibits a linear relationship with the level of feature density in truncated gradient algorithm. This pattern suggests that the amount of shrinkage applied to features should be approximately proportional to their data density, on the probability of having informative updates. This provides an independent justification of the informative truncation introduced in Section 3.3.1. In contrast, features selected by the proposed algorithm have approximately uniform distribution of sparsity levels.

This improvement over the truncated gradient algorithm mostly owes to the use of informative truncation. With the crucial treatment on the heterogeneity in feature sparsity, the proposed algorithm is effective in keeping rare but important features from premature truncations.

Based on the measure of feature selection stability defined in (3.34), the performance of the original truncated gradient algorithm [Langford *et al.*, 2009] and the proposed algorithm are summarized in Table 3.4. The proposed algorithm demonstrates its excellent stability as compared to the truncated gradient algorithm. It shows greater overlap in selected features among different permutations of the training data. This result also echos with the lower standard deviations in the test error (Table 3.2) and in the number of selected features (Table 3.3). As discussed in Section 3.6.3.1, the insensitivity to data perturbation is mainly due to the fair shrinkage applied on each feature by using informative truncation and to the sufficient accumulations of information in selection probabilities before applying stability selection. These measures suggest that the proposed algorithm is particularly favorable for high-dimensional sparse data.

3.6.3.3 Convergence

Our main motivation of the proposed method stems from observing the slow and unstable convergence of the truncated gradient algorithm as shown in Figure 3.1 in Section 3.2. In this sub-section, we show the proposed algorithm significantly improves the condition with fast and stable convergence in both test errors and numbers of nonzero weights. The test erros and number of selected variables over iteractions are shown in Figure 3.4 and Figure 3.5, respectively.

As compared to Figure 3.1, the test error of the stablized truncated SGD algorithm is quickly reduced after a few bursts and remains relatively stable as the algorithm proceeds around a low value. The path of test error in Figure 3.4 also implies that the stabilized algorithm resists over-fitting after convergence. Figure 3.5 indicates a fast

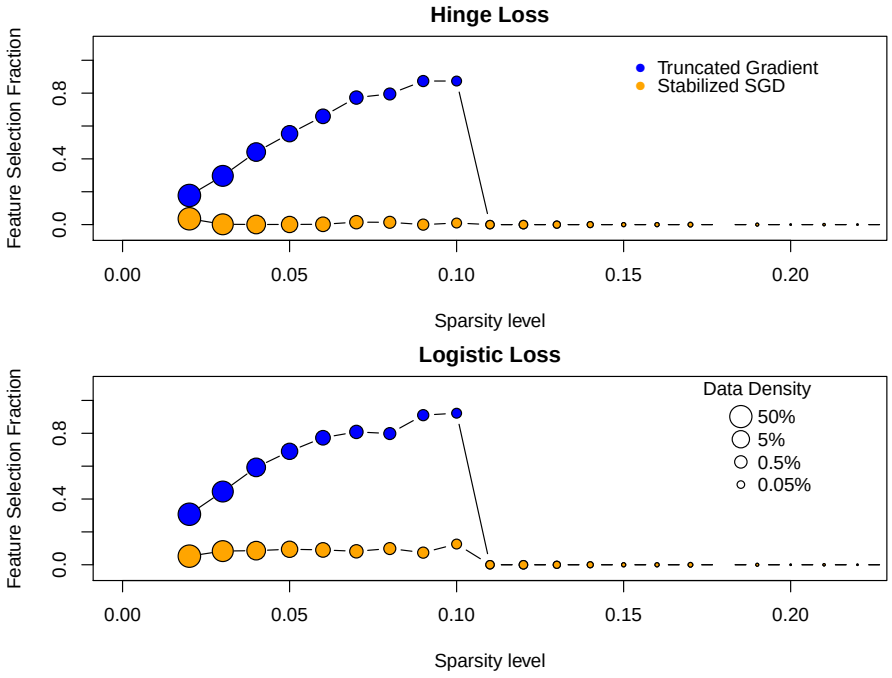


Figure 3.3: The fraction of features selection at different sparsity levels by the truncated gradient algorithm and the proposed algorithm respectively with the Dorothea dataset as an example. The selected features are extracted as the features with nonzero entries in the resulted weight vector from the algorithms with a single fixed ordering of the training data. The x-axis represents the sparsity level, which is the proportion of nonzero entries in a feature in the training sample. The y-axis indicates the fraction of selected features among features with a discretized sparsity level. The size of each point scales with the proportion of features that fall in level of sparsity. It can be seen that the truncated gradient algorithm over-selects denser features. The proposed algorithm does not have a pre-exposed preference towards any density level.

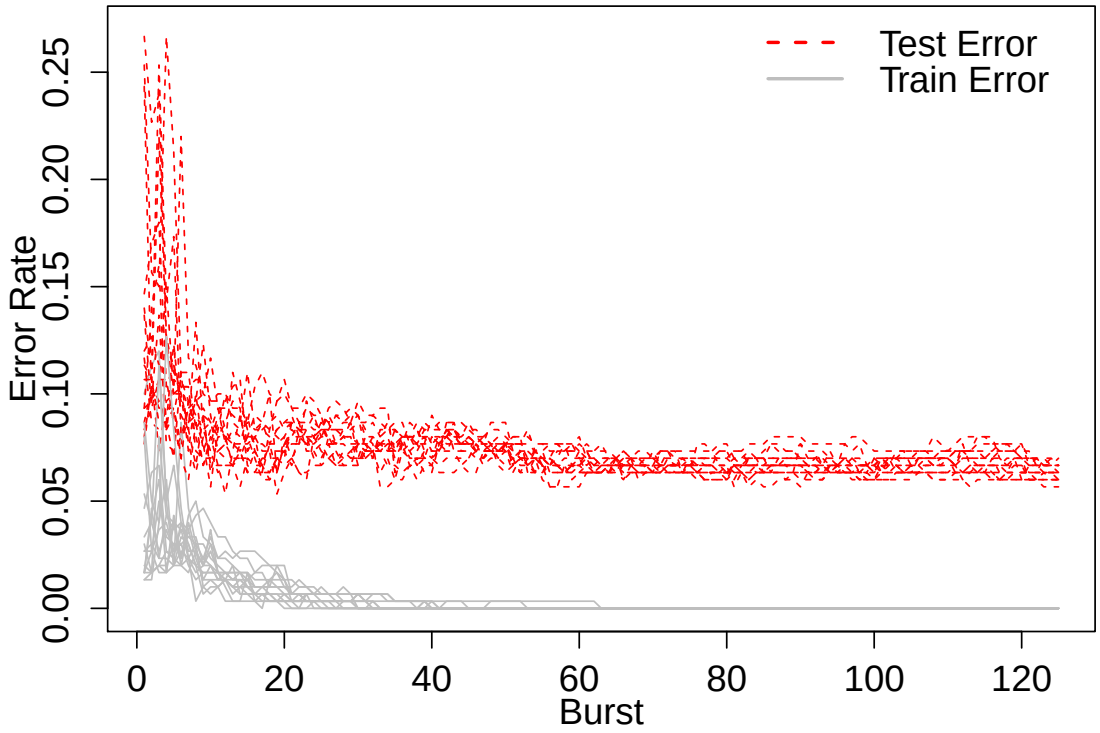


Figure 3.4: The error rates of the proposed stabilized truncated SGD algorithm along with the number of iterations on the example Dexter dataset with $M = 16$ threads of SGD updating paths.

and efficient elimination of most irrelevant features. The numbers of selected variable only have small variations between two stability selections. Unlike the truncated gradient algorithm, there is no large jump back to dense weight vector. The effectiveness of detecting and purging of irrelevant features also explains the good generalizability in Figure 3.4.

3.7 Conclusion

In this chapter, we address the problem of inducing sparsity in subgradient-based online learning algorithm when applied to high-dimensional sparse data. Based on

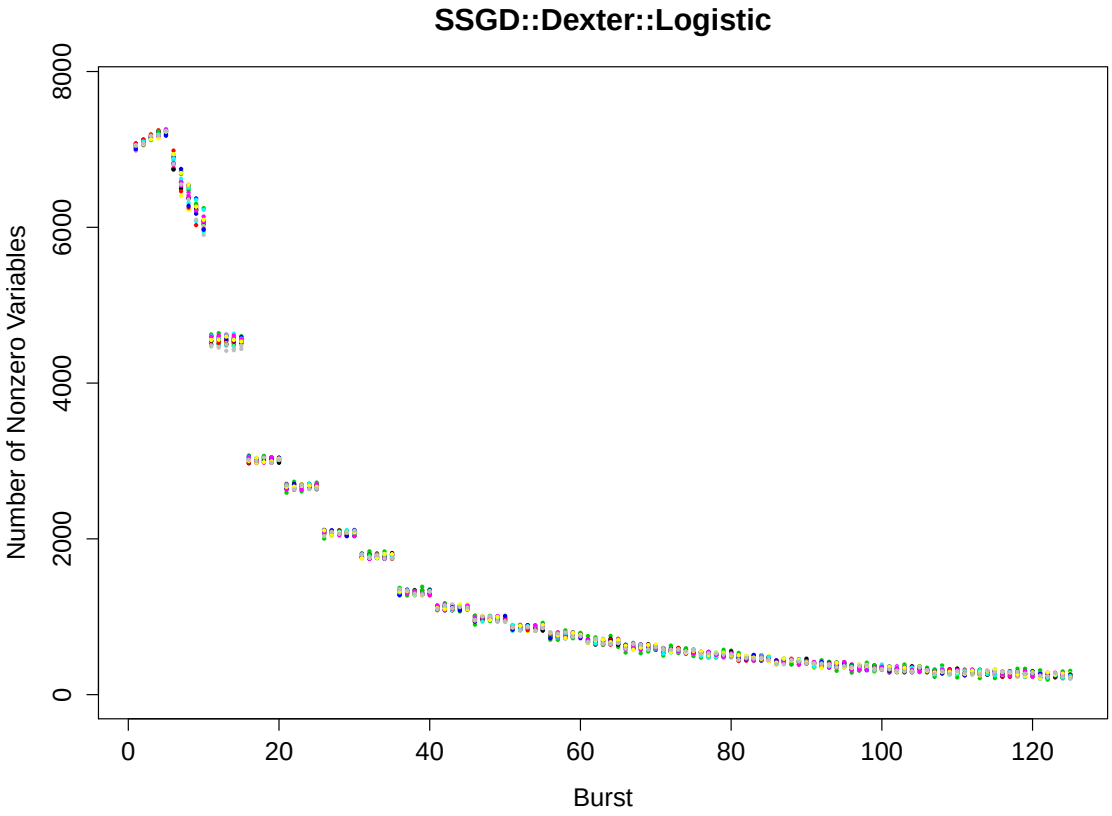


Figure 3.5: The number of nonzero features in the resulted weight vector learned by the proposed stabilized truncated SGD algorithm along with the number of iterations on the example Dexter dataset with $M = 16$ threads of SGD updating paths.

the truncated gradient framework, we reduce truncation bias due to heterogeneity in feature sparsity via informative truncation. Integrated with soft-thresholding truncation, stability selection helps eliminate irrelevant features along the learning process in order to achieve stable and sufficiently sparse results. The adaptive gravity method dynamically adjusts the shrinkage at different stages of the learning process to balance between the exploration of sparse combination of nonzero weights and the exploitation of finite updates for better convergence. This strategy also consolidates the tuning needs of the proposed algorithm into two major parameters, the annealing rate γ and the purging threshold π_0. We present a theoretical analysis of the expected regret

bound, which offered a quantification of the potential improvement attained by the proposed algorithm. At last, we use extensive experimental studies to evaluate the performance of the proposed algorithm. The results demonstrate that our algorithm achieve favorable results in terms of both prediction accuracy and feature selection, compared to the original truncated gradient algorithm. The proposed algorithm can serves as the computational engine for not only the sDist algorithm discussed in Chapter 2 but a broad class of learning methods, in both classification and regression, for inducing sparse solution in feature space.

Algorithm 3.4 $\Psi(\mathcal{D})$: Stabilized truncated stochastic gradient descent for sparse learning.

Input: the training data $\mathcal{D}$.

Parameters: β_0, γ, π_0, K, n_K, M, η.

Initialization: $\forall m \in \{1, \ldots, M\}$, initialize $\mathbf{w}_0^{(m)} = \mathbf{0}_p$, $\tilde{\mathbf{k}} = \mathbf{0}_p$ and $\mathcal{D}^{(m)}$ is a random permutation of $\mathcal{D}$; $\hat{\Omega}_0 = \{1, \ldots, p\}$.

For $s = 1, 2, \ldots,$

repeat

Obtain $g_{0,s}(\beta_s)$ as in (**??**).

for all $m \in \{1, \ldots, M\}$ **parallel do**

for $\tau = 1$ **to** n_K **do**

$\hat{\mathbf{w}}_\tau^{(m)} = B_2\left(\mathbf{w}_0^{(m)}, g_{0,s}(\beta_s), \hat{\Omega}_{s-1}, \mathcal{D}^{(m)}\right)$.

$\mathbf{w}_0^{(m)} = \hat{\mathbf{w}}_\tau^{(m)}$.

end for

end for

Compute $\hat{\Pi}_s$ as in(3.9) and update the set of stable features $\hat{\Omega}_s = \left\{j : \hat{\Pi}_{s,j} \geq \pi_0\right\}$.

$\tilde{\mathbf{w}}_s^{(m)} = \hat{\mathbf{w}}_{n_K}^{(m)} \mathbb{1}_{\hat{\Omega}_s}$, $m = 1, \ldots, M$.

$\mathbf{w}_0^{(m)} = \tilde{\mathbf{w}}_s^{(m)}$.

$\beta_{s+1} = \phi(d_s; \beta_0, \gamma)$ where $d_s = \frac{|\hat{\Omega}_s|}{p}$.

Aggregation: $\underline{\mathbf{w}} = \frac{1}{M} \sum_{m=1}^{M} \tilde{\mathbf{w}}_s^{(m)}$.

until $\underline{\mathbf{w}}$ converges.

Return: $\underline{\mathbf{w}}$.

Table 3.2: Mean test errors (%) and the corresponding standard deviations with hinge loss and logistic loss over 50 random ordering of the training samples.

	Dataset	Standard SGD	Truncated Gradient	RDA	FOBOS	Stabilized Truncated SGD
Hinge Loss	RCV1	**2.86**	6.59	4.62	8.44	3.01
		.08	*.09*	*.10*	*.15*	*.04*
	Dexter	7.77	8.37	11.42	10.91	**6.58**
		.96	*.91*	*.59*	*1.21*	*.57*
	Dorothea	6.11	6.83	6.83	8.51	**5.14**
		.60	*1.23*	*.18*	*1.96*	*0.43*
	Gisette	2.71	6.32	**2.40**	5.50	3.05
		.43	*3.02*	*.45*	*1.20*	*.30*
	Arcene	19.26	22.06	22.10	18.72	**17.60**
		3.67	*8.08*	*4.00*	*3.26*	*2.62*
Logistic Loss	RCV1	3.68	18.55	5.69	14.18	**3.19**
		.07	*.78*	*.08*	*.39*	*.05*
	Dexter	7.91	10.25	9.41	10.45	**6.41**
		.36	*.58*	*.68*	*.81*	*.43*
	Dorothea	6.26	7.29	6.87	7.15	**5.47**
		.62	*1.28*	*.31*	*1.35*	*.35*
	Gisette	2.52	6.35	2.21	5.98	**2.11**
		.29	*4.08*	*.20*	*.85*	*.26*
	Arcene	17.38	19.48	21.62	18.44	**13.38**
		2.93	*5.01*	*2.05*	*3.11*	*1.26*

Table 3.3: The average percentages (%) of nonzero features selected in the resulted weight vector and the corresponding standard deviations with hinge loss and logistic loss over 50 random permutations of the training samples.

	Dataset	Standard SGD	Truncated Gradient	RDA	FOBOS	Stabilized Truncated SGD
Hinge Loss	RCV1	63.68	6.95	11.38	5.01	0.86
		.34	*.42*	*.10*	*.28*	*.10*
	Dexter	29.46	14.74	1.61	37.69	1.98
		.35	*.82*	*.06*	*2.43*	*.15*
	Dorothea	39.84	18.62	0.55	14.89	6.68
		.62	*5.94*	*.02*	*3.37*	*.09*
	Gisette	90.88	63.8	17.45	61.81	3.84
		.21	*3.72*	*.43*	*3.94*	*.34*
	Arcene	98.22	61.08	12.15	54.5	3.67
		.07	*25.55*	*.90*	*13.37*	*.40*
Logistic Loss	RCV1	82.31	2.75	6.98	2.89	1.44
		.00	*.35*	*.05*	*.29*	*.12*
	Dexter	38.76	13.91	4.84	30.21	1.32
		.00	*.71*	*.14*	*1.87*	*.13*
	Dorothea	68.63	19.31	0.43	20.4	1.67
		.00	*3.00*	*.01*	*3.51*	*.47*
	Gisette	93.96	66.31	14.99	58.83	3.22
		.00	*3.31*	*.34*	*3.44*	*.49*
	Arcene	98.36	88.34	12.71	45.96	5.35
		.00	*4.38*	*.27*	*9.76*	*.24*

Table 3.4: Summary of feature selection stability performance of the resulted weight vector measured by (3.34) based on Cohen's kappa coefficient.

	Hinge Loss		Logistic Loss	
Dataset	Truncated Gradient	Stabilized Truncated SGD	Truncated Gradient	Stabilized Truncated SGD
RCV1	0.38	0.44	0.29	0.60
Dexter	0.46	0.61	0.23	0.58
Dorothea	0.19	0.52	0.19	0.33
Gisette	0.48	0.60	0.54	0.54
Arcene	0.26	0.55	0.35	0.69

Chapter 4

Extensions and Conclusion

In Chapter 2 and Chapter 3, we introduce two methods aiming at inducing sparsity in feature space. They provide vehicles for efficiently extracting low-dimensional feature subspaces on which the patterns embedded in the data are revealed with much better definition to facilitate building good classifiers. In this section, based on these proposed frameworks, we make further extensions, which provide future directions for both theoretical and application-oriented developments. The ability of handling high-dimensional data makes both of the proposed methods pertinent for some popular machine learning applications, such as computer vision. Unlike the state-of-art methods with formidable complexity, through the proposed method, the learned feature subspaces can help identify and visualize the key factors for distinguishing different class labels. Such an advantage can be availed to reduce computational costs and to interpret or diagnose observed patterns. On the other hand, we generalize the distance-based framework in Chapter 2 to the *kernel machine*. By recognizing kernel as a generalized distance measure and the duality between kernel functions and random process, we approximate kernels by random features to overcome the scalability issue of kernel machine. Such a technique not only aborts the reliance on the prohibitively large kernel matrices but also suggests a way to configure feature subspace learning via inducing sparsity on coefficients of random features. At last, we conclude

this thesis by summarizing the commonalities and distinctive characteristics of each method proposed.

4.1 Potential Applications in Machine Learning: Computer Vision

The capability of automating object recognition has always been a top goal of artificial intelligence research. In recent years, a panoply of sophisticated methods are developed to counter this challenge. As the methods become provably effective, they have evolved as giant models with hundreds of thousands parameters that are no longer fathomable. Despite of the good performances, people cannot understand the model's functionality and decipher what are the scientific interpretations of the results. What is advocated in the proposed sparse learning methods in this thesis is not only about good generalizability but about straightforward interpretation of models on high-dimensional data with complex structure. In this section we demonstrate how our methods of flexible feature subspace learning work on computer vision applications, particularly for image classification. Although our method cannot achieve test errors that match with complicated black-box methods, such as the popular deep learning methods, our models manage to identify and to visualize the driven forces behind computer vision applications, which might help better understanding of the problems at hand and fostering further improvements.

4.1.1 ZIP Code Digits Recognition

We first consider a toy example of digits recognition in ZIP code[1] [Le Cun *et al.*, 1990]. This problem captured the attention of the machine learning community for many years, and has remained a benchmark problem in the field. Since we mainly concern

[1]Data source: `http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/`

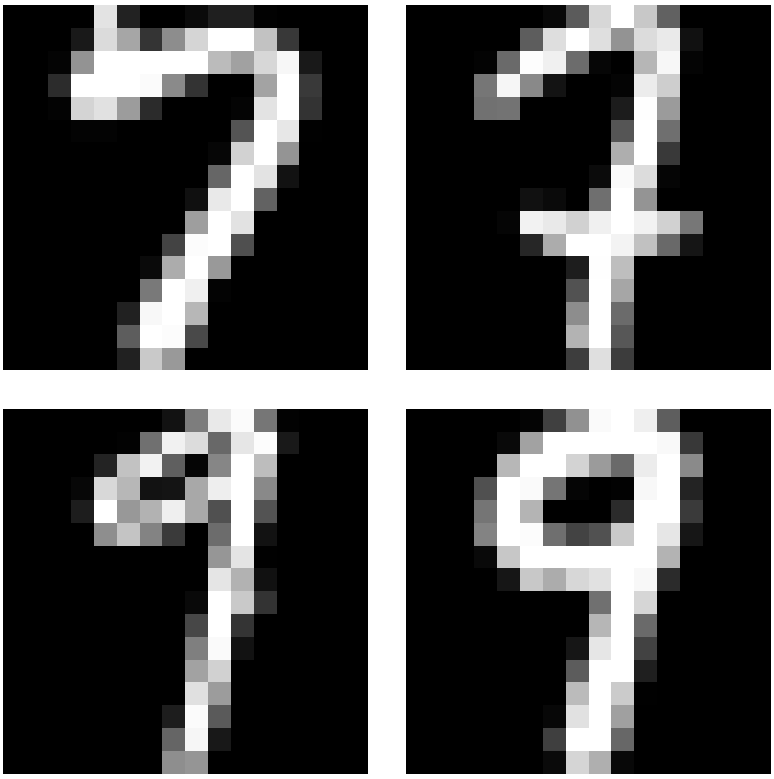


Figure 4.1: Examples of training cases of digit 7 and digit 9 from ZIP code data. Each image is a 16×16 8-bit grayscale representation of a handwritten digit.

about binary classification, we only take digit 7 and digit 9 in this example, which are generally considered as the most difficult pair to discern. Figure 4.1 shows some examples of handwritten digits, automatically scanned from envelopes by the U.S Postal Service. The scanned digits are normalized to a uniform size and orientation such that each pixel represents the same spatial feature across images, rendering in 16×16 grayscale images.

Because of the normalization of image sizes and orientations, the raw pixels can be directly used as input feature vectors since they basically describe the same visual meanings. We train the *sDist* algorithm in Chapter 2 on the subsample of digits 7 and 9, which results in a feature subspace defined by the sparse weight matrix W and the learned nonlinear feature mapping $\phi(\cdot)$. In Figure 4.2, we visualize the

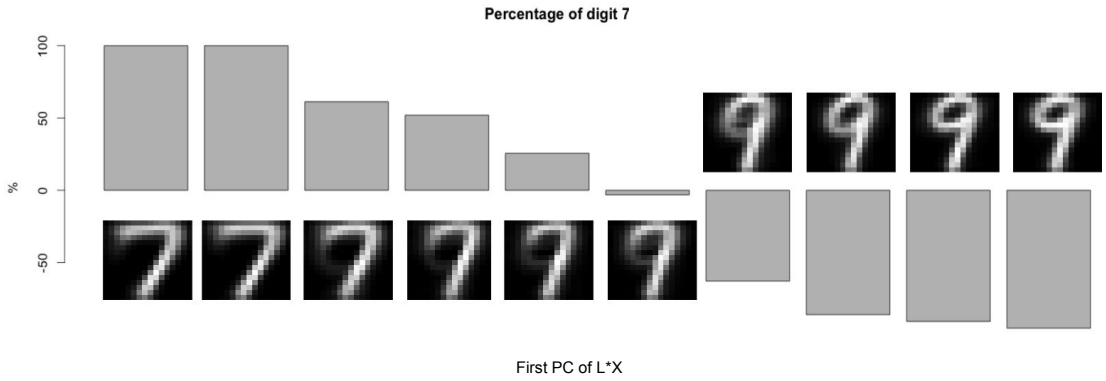


Figure 4.2: The percentage of digit 7 minus the percentage of digit 9 in each bin of the values the first principle component (PC) of the transformed feature subspace $L\tilde{X}$, where $W = L^T L$ and W is the resulted weight matrix from the $sDist$ algorithm proposed in Chapter 2. A positive percentage indicates a cluster of digit 7's and a negative percentage indicates a cluster of digit 9's. This figure implies that the first PC of the transformed feature vectors concisely summarizes the information of the class difference as an one-dimensional feature.

distributions of digit 7 and digit 9 on the one-dimensional subspace defined as the first principle component of the transformed matrix $L\tilde{X}$, where $W = L^T L$ and $\tilde{X}$ is the training data matrix transformed by $\phi(\cdot)$. As we can see from the figure, different digits are well separated on the learned nonlinear feature subspace. Digit 7's are mostly clustered on the left of the x-axis while digit 9's on the right of the x-axis. The first principle component of $L\tilde{X}$ captures the signal of class differences such that two classes of digits are almost linearly separable on this one-dimensional line. It can also be considered as a discriminant between classes.

Unlike other complex methods for object recognition, we can directly visualize which features make up the feature subspace on which the digits are mostly separable. In Figure 4.3, we highlight the selected features (pixels) in the sparse weight matrix W in randomly sampled image of digit 7. It is shown that the main signal are composed

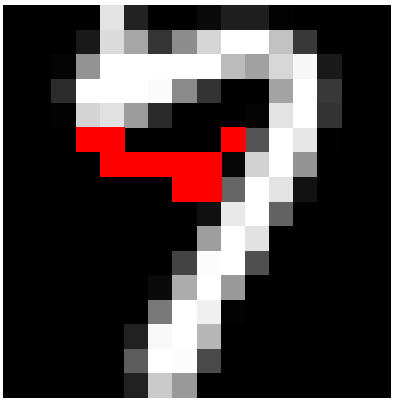


Figure 4.3: The selected pixels by the *sDist* algorithm highlighted on a randomly sampled digit 7.

of almost all the relevant pixels that visually distinguish digit 7 and 9. Our algorithm provides a short list of features that matters the most to the classification task without redundancy. Although this example is of moderate dimensionality, it instantiates a case that how our method can contribute to particular computer vision tasks by identifying a compact solution. Based on the sparse result, future applications may be saved by a considerable amount of computation time and may achieve better generalizability.

4.1.2 Cats Vs. Dog Classification

Another example is the classification of cats and dogs in pictures[2] [Parkhi *et al.*, 2012]. In this dataset, there are 7390 images, including 4990 of dogs and 2400 of cats, of different breeds. As an example, we extract a subsample that is known to be difficult to classify: all images of the *pomeranian* dogs, which are small in size

[2]Data source: `http://www.robots.ox.ac.uk/~vgg/data/pets/`

and are often misrecognized as cats, and a randomly sampled cat images of the same count. Figure 4.4 shows a small subset of the figures. We randomly split the original data into a 70% training set and a 30% testing set. Different from the zip code example, the pictures in this set are taken in real circumstances in which objects appear in various positions, gestures, light conditions with complicated backgrounds. Moreover, there is no uniform size for these pictures. Hence, pixels are not aligned to consistently represent the same visual meanings across images, making them invalid to be input features. Instead, to represent textures, we extract *visual bag-of-word features* from the training image set [Fei-Fei and Perona, 2005], which provides a histogram representation of the image based on independent features. Based on the visual bag-of-word model, visual words [Sivic and Zisserman, 2003] are computed densely on the image by extracting *scale-invariant feature transform* (SIFT) feature descriptor [Lowe, 2004]. An example of extracted SIFT keypoints along with the corresponding graphical descriptors is shown in Figure 4.5. The descriptors are then quantized to create a codebook by using the k-means clustering algorithm with a given vocabulary size. At last, the normalized frequency of the codebook elements in each image are generated as the final input features. The resulted features have an average feature-wise sparsity of 17.6%.

Given the head bounding boxes, we extract a total of 31,000 features by concatenating the following features:

- SIFT visual words with vocabulary size 5,000 using the *spatial histogram* [Lazebnik *et al.*, 2006]. The layout of spatial histogram consists of five spatial bins organized as a 1×1 and a 2×2 grids of uniform sizes;
- SIFT visual words with vocabulary size 5,000 on the head image along without using the spatial histogram'
- Color histogram features using 10 bins in each of the three color channels.

Using these features, we implement a modified *sDist* algorithm by substituting

Figure 4.4: An example of figures for cats and dogs classification: pomeranian dogs in the *upper panel* and randomly selected cats in the *lower panel*.

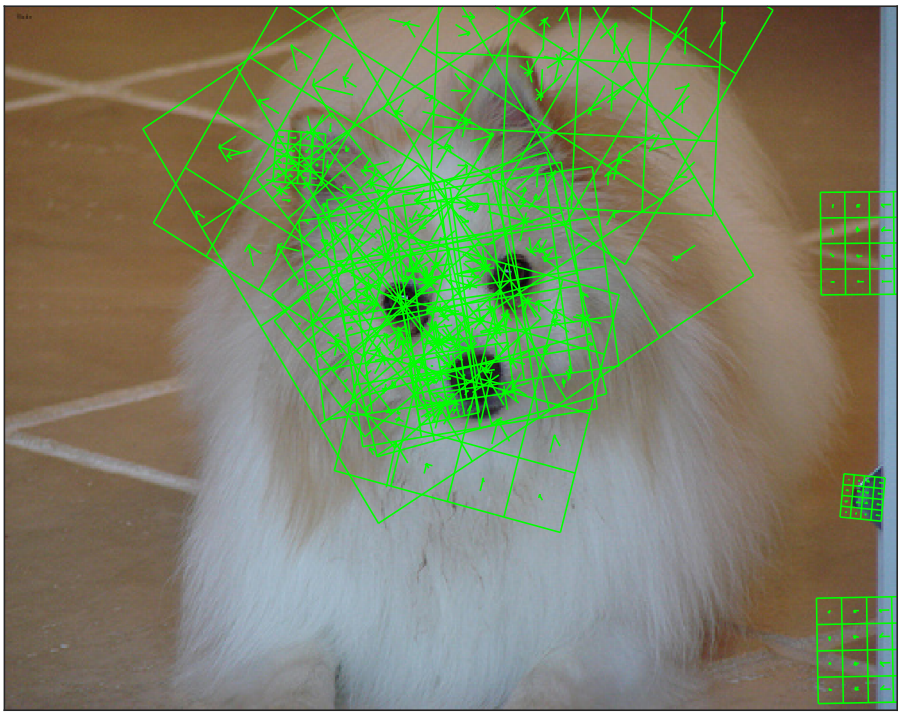
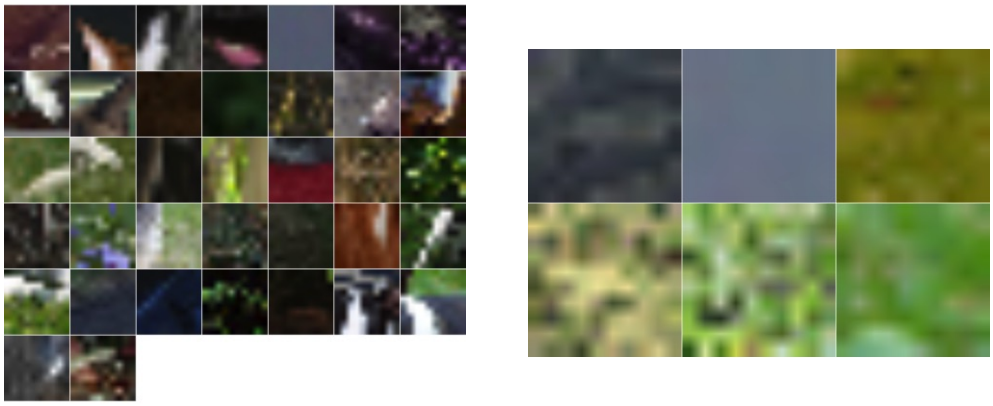


Figure 4.5: A example of extracted SIFT keypoints and along with the corresponding graphical descriptors on a sample pomeranian image in the training data.



(a) X_1 (b) X_1

Figure 4.6: An example of extracted feature by using the proposed algorithm. Besides of being identified as informative features as individuals, these two features compose of an interaction feature in the learned nonlinear feature mapping by the *sDist* algorithm.

step (c) of Algorithm 2.2 in Section 2.4 by the stabilized truncated stochastic gradient descent algorithm in Algorithm 3.4 in Section 3.3. It allows the *sDist* algorithm to be generalized to data with higher dimensionality, high degree of sparsity and large dimensionality by using the informative truncation and stability selection introduced in Chapter 3. We go through the same tuning procedure as introduced in Chapter 2 and constrain the maximum order of interaction as 2, that is, by considering potential pairwise interactions. Although the performance not yet outperforms that of the highly complex deep learning framework, such as CAFFE [Jia *et al.*, 2014]. Unlike the "blackbox" algorithms, the learned low-dimensional feature subspaces allow us to investigate and to visualize what kind of visual factors really make a difference between classes. In Figure 4.6, we show an example of extracted features that are selected by the proposed algorithm.

The visualization is generated as follows: since each feature corresponds to a code

in the codebook learned from the training data, that is, a cluster center in the result of the k-means clustering, we retrieve the key points in all training images that fall into the selected cluster center. Then, we plot the patches around all the relevant key points with widths equal to twice of the widths of the detected key points.

In Figure 4.6, we can recognize that feature (a) roughly represents the shape of an upright ear and feature (b) captures the texture of grass lawn. Besides of being identified as informative features as individuals, these two features compose of an interaction feature in the learned nonlinear feature mapping by the $sDist$ algorithm. So the co-appearance of these two features indicates that the visual components of an upright ear shape and of grass lawn jointly construct an informative feature subspace on which cats and dogs can be separated with larger margin. So besides of the ability of making good prediction, the proposed algorithm also provide valuable insights of the driving factors that helps researchers understand the underlying mechanism of computer vision applications. This functionality is one of the most crucial motivation of the proposed methods and also of this thesis for learning informative feature subspaces.

4.2 Extension to Kernel Machine

Kernel machines, such as Support Vector Machine (SVM), are widely used in machine learning to achieve nonlinear decision boundary [Hofmann *et al.*, 2008]. The kernel methods uses a positive-definite kernel function to induce an implicit nonlinear feature mapping, which could potentially be infinitely dimensional. The well-known *"kernel trick"* allows one to exploit these rich feature spaces without explicitly working on such high dimensions. Moreover, it is known that positive-definite translation-invariant kernels can be used to generalize the notion of distances in feature space [Schiilkopf, 2001]. Hence, the nonlinear feature space induced by kernel functions can be used to generalized distance metric learning for greater flexibility.

However, a caveat is that, despite of their popularity, kernel machines are not scalable with the number of training samples. At training time, they require the computations of the $n \times n$ kernel matrix, where n is the sample size. Another drawback of kernel machine is mentioned in Chapter 2 Section 2.4 that, for the purpose of obtaining sparse and interpretable results, one cannot rely on the "kernel trick" since evaluating each feature is too costly to be implemented.

A solution might be found by using *kernel approximation* with sparse learning. Kernel approximation via explicit nonlinear maps has became a popular strategy for speeding up kernel machines. Among these, *Random Features* has attracted considerable recent interest due to its simplicity and efficiency [Rahimi and Recht, 2007]. In this section, we first recognize the Mahalanobis distance on a nonlinear feature space as a positive-definite translation-invariant kernel. Therefore, we can extend the distance metric learning framework of *sDist* algorithm in Chapter 2 to the domain of kernel machine for improved flexibility. By using kernel approximation via random features, it enables scalable computations for data with both high dimensionality and large sample size.

4.2.1 Kernel as Generalized Distance Measure

We begin the discussion by reviewing the positive definite translation-invariant kernels.

Definition 4.1. *A kernel* $k(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *is translation-invariant if*

$$k(\mathbf{x}, \mathbf{x}') = k'(\mathbf{x} - \mathbf{x}').$$

Definition 4.2 (Positive-Definite Kernel)**.** *A symmetric kernel* $k(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *is called positive definite if, for all* $m > 1$*, and* $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{X}$*, and* $c_1, \ldots, c_m \in \mathbb{R}$*,*

$$\sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j k(\mathbf{x}_i, \mathbf{x}_j) > 0. \tag{4.1}$$

Beyond learning a linear transformation of the feature space using the Mahalanobis distance, we introduce a distance metric based on a nonlinear feature mapping $\phi(\mathbf{x}) : \mathcal{X} \to \mathcal{F}$ in Section 2.4. We limit the discussion within a class of polynomial feature mappings to retain tractable computation and interpretation. In this extension, we consider a broader class of nonlinear feature mapping from the kernel point of view. Given a nonlinear feature mapping ϕ, the squared Euclidean distance on the nonlinear feature space can be written as

$$\begin{aligned} d_\phi(\mathbf{x}, \mathbf{x}')^2 &\triangleq ||\phi(\mathbf{x}) - \phi(\mathbf{x}')||^2 \\ &= \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle + \langle \phi(\mathbf{x}'), \phi(\mathbf{x}') \rangle - 2\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle \\ &= k(\mathbf{x}, \mathbf{x}) + k(\mathbf{x}', \mathbf{x}') - 2k(\mathbf{x}, \mathbf{x}'), \end{aligned}$$

with the kernel trick $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$.

Hence if the kernel k is definite-positive and translation invariant, the distance can be computed from a kernel function:

$$d_\phi(\mathbf{x}, \mathbf{x}') = \sqrt{-2k(\mathbf{x}, \mathbf{x}')}. \tag{4.2}$$

The Mahalanobis distance on the nonlinear feature space with weight matrix W can be written in a equivalent form as (4.2) by using the kernel $\tilde{k}(\mathbf{x}, \mathbf{x}') = \langle \tilde{\phi}(\mathbf{x}), \tilde{\phi}(\mathbf{x}') \rangle$, where $\tilde{\phi}(\mathbf{x}) \triangleq L\phi(\mathbf{x})$ and $W = L^T L$.

On the other hand, it is shown that kernel can be treated as a generalized measure of similarity as distance metrics [Schiilkopf, 2001]:

Proposition 4.1 (Hilbert Space Representation of Positive Definite Kernels [Schiilkopf, 2001])**.** *Let k be a real-valued positive-definite kernel on $\mathcal{X}$, satisfying $k(\mathbf{x}, \mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$. Then there exists a Hilbert space $\mathcal{H}$ of real-valued functions on $\mathcal{X}$, and a mapping $\phi : \mathcal{X} \to \mathcal{H}$, such that*

$$||\phi(\mathbf{x}) - \phi(\mathbf{x}')||^2 = -k(\mathbf{x}, \mathbf{x}').$$

It can be shown that if $k(\mathbf{x}, \mathbf{x}) = 0$ for all $\mathbf{x} \in \mathcal{X}$, then

$$d_k(\mathbf{x}, \mathbf{x}') \triangleq ||\phi(\mathbf{x}) - \phi(\mathbf{x}')|| = \sqrt{-k(\mathbf{x}, \mathbf{x})}$$

is a semi-metric [Cowling, 1983].

Based on above observations, the discriminant function with nonlinear feature mapping ϕ is constructed in the similar fashion as in (2.3):

$$f_\phi(\mathbf{x}_i) = k(\mathbf{x}_i, \mathbf{x}_j) - k(\mathbf{x}_i, \mathbf{x}_k), \tag{4.3}$$

where, by using 1-nearest neighbor, $\mathbf{x}_j$ is the nearest sample point in positive class to $\mathbf{x}_i$ and $\mathbf{x}_j$ is the nearest sample point in negative class to $\mathbf{x}_i$. Hence, the local-distance-based discriminant function for distance metric learning problem in Chapter 2 is transformed into a kernel machine. The goal is then to search for an optimal kernel function that maximizes the margin in terms of kernel between two classes in local neighborhoods.

Existing works in the literature of distance metric learning have explored the possibilities of incorporating kernels into the Euclidean/Mahalanobis distance metric [Kedem *et al.*, 2012] [Wang *et al.*, 2011] [Jain *et al.*, 2012]. However, these methods exclusively rely on the "kernel trick" by which the entire $n \times n$ Gram matrix needs to be computed and stored. In such a case, kernelized distance metric learning cannot be applied on data with large sample size due to excessive computational cost. The "kernel trick" also prevents one to identify low-dimensional feature subspaces. Instead, a integral representation of kernel function provides an alternative to approximate kernel function. It enables tractable computation as well as an additive structure that facilitates sparse learning.

Theorem 4.1 (Integral Representation of Kernel Function)**.** *If $k(\mathbf{x}, \mathbf{x}')$ is a positive definite kernel, then there exists a set Ω, a measure $\mathbb{P}$ on Ω, and random feature $\phi_\omega(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ from $L_2(\mathbb{R})$, such that*

$$k(\mathbf{x}, \mathbf{x}') = \int_\Omega \phi_\omega(\mathbf{x})^T \phi_\omega(\mathbf{x}') d\mathbb{P}(\omega).$$

Essentially, the above integral representation relates kernel functions to a random process ω with measure $\mathbb{P}(\omega)$. Note that the integral representation of a kernel function may not be unique. If the kernel is also continuous and translation-invariant,

the integral representation specializes into a form characterized by inverse Fourier transform stated in the Bochner's Lemma [Rudin, 2011]:

Theorem 4.2 (Bochner's Lemma [Rudin, 2011]). *A continuous, real valued, symmetric and translation-invariant function $k(\mathbf{x}-\mathbf{x}')$ is a positive definite kernel if and only if $k(\mathbf{x}-\mathbf{x}')$ is the Fourier transform of a non-negative measure. That is, there exists a non-negative measure $\mathbb{P}(\omega)$ on Ω, such that*

$$\begin{aligned} k(\mathbf{x}-\mathbf{x}') &= \int_{\Omega} e^{i\omega(\mathbf{x}-\mathbf{x}')} d\mathbb{P}(\omega) \\ &= \int_{\Omega\times[0,2\pi]} 2\left[\cos(\omega^T\mathbf{x}+b)\cos(\omega^T\mathbf{x}'+b) + \sin(\omega^T\mathbf{x}+b)\sin(\omega^T\mathbf{x}'+b)\right] \\ &\qquad \cdot d(\mathbb{P}(\omega)\times\mathbb{P}(b)) \\ &= \int_{\Omega\times[0,2\pi]} 2\cos(\omega^T\mathbf{x}+b)\cos(\omega^T\mathbf{x}'+b) d(\mathbb{P}(\omega)\times\mathbb{P}(b)) \\ &= \mathbf{E}_w\left[\phi_\omega(\mathbf{x})^T\phi_\omega(\mathbf{x}')\right], \end{aligned}$$

where $\mathbb{P}(b)$ is a uniform distribution on $[0,2\pi]$, and $\phi_\omega(\mathbf{x}) = \sqrt{2}\cos(\omega^T\mathbf{x}+b)$. The imaginary part is dropped since the kernel function is real. The measure $\mathbb{P}(\omega)$ is also known as the spectral density of $k(\cdot)$.

Therefore, the kernel has an unbiased estimation using the inner product of the random feature vector:

$$\begin{aligned} \hat{\phi}(\mathbf{x}) &= \frac{1}{\sqrt{D}}(\phi_{\boldsymbol{\omega}_1}(\mathbf{x}),\ldots,\phi_{\boldsymbol{\omega}_D}(\mathbf{x}))^T \\ &= \sqrt{\frac{2}{D}}\left(\cos(\boldsymbol{\omega}_1^T\mathbf{x}),\ldots,\cos(\boldsymbol{\omega}_D^T\mathbf{x})\right)^T. \end{aligned}$$

One can use Monte Carlo methods to approximate the kernel $k(\mathbf{x},\mathbf{x}')$ by sampling D vectors ω's i.i.d from $\mathbb{P}(\omega)$ Hence,

$$k(\mathbf{x},\mathbf{x}') \approx \hat{\phi}(\mathbf{x})^T\hat{\phi}(\mathbf{x}) = \frac{1}{D}\sum_{l=1}^{D}\phi_{\boldsymbol{\omega}_l}(\mathbf{x})^T\phi_{\boldsymbol{\omega}_l}(\mathbf{x}'),$$

which has an additive structure.

The advantage of this approximation is that we may now approximate the discriminant function in (4.3) as

$$f(\mathbf{x}_i) \approx [\hat{\phi}(\mathbf{x}_j) - \hat{\phi}(\mathbf{x}_k)]^T \hat{\phi}(\mathbf{x}_i). \tag{4.4}$$

Hence, the objective becomes the search for an optimal random feature vector $\hat{\phi}(\cdot)$.

4.2.2 Duality between Kernel and Random Process

On the other hand, Bochner's Lemma also allows one to work in the other direction. Instead of finding the random process $\mathbb{P}(\omega)$ and the mapping $\phi_\omega(\mathbf{x})$, one can go the reverse direction by constructing kernels from given random processes and feature mappings.

Theorem 4.3 ([Dai *et al.*, 2014])**.** *If* $k(\mathbf{x}, \mathbf{x}') = \int_\Omega \phi_\omega(\mathbf{x})^T \phi_\omega(\mathbf{x}') d\mathbb{P}(\omega)$ *for a non-negative measure* $\mathbb{P}(\omega)$ *on* Ω *and* $\phi_w(\mathbf{x}) : \mathcal{X} \to \mathbb{R}^d$*, each component from* $L_2(\Omega, \mathbb{P})$*, then* $k(\mathbf{x}, \mathbf{x}')$ *is a positive definite kernel.*

Hence, we can learn a random process for a customized approximation for a kernel. Such a kernel would define a nonlinear transformation $\phi(\cdot)$ by which the margin between classes is locally maximized.

4.2.3 Hierarchical Sparse Learning Via First-Order Markov Process

To achieve sparsity in feature space, we propose to induce sparsity in ω_l of random features. Let $\mathbf{z}_l$ be the set of feature indices with nonzero values in ω_l, which we refer as to the "atoms". We treat $\mathbf{z}_l$ be an infinitely long stochastic process, where $z_{l,t} \in \mathcal{I} \bigcup \{0\}$ for $t = 1, 2, \ldots$ and $z_{l,0} = 0$. The set $\mathcal{I} = \{1, \ldots, p\}$ is the feature index set and 0 indexes the "absorbing state", or the "null state". The generative process sequentially picks features until the process returns to the null state 0.

In Section 2.4, we propose a hierarchical nonlinear expansion for polynomial features. In this extension, we hold a similar belief that higher-order interactions shall be considered if and only if lower-order features/interactions are included in the model. To generalize this idea to the kernel machine via random features, we impose a *first-order Markov chain prior* on the atoms $\mathbf{z}_l$:

$$\mathbb{P}(z_{l,t} = j' | z_{l,t-1} = j, \pi_0, \Theta) = \theta_{j,j'}, \quad j, j' \in \{0, 1, \ldots, p\}, \ t = 1, 2, \ldots \tag{4.5}$$

With this prior, features are sequentially selected such that the subsequent selection of needed feature depends on previous selections to form a interaction term that is informative to the learning task. The prior in (4.5) generalizes the stepwise hierarchical expansion in Section 2.4 to a random process. It also permits sparse learning that keeps the list of involved features to minimum, which makes it promising for nonlinear sparse learning problems.

We consider a logistic distribution on the binary class label Y. The generative process is summarized as follows:

- Global variables:
  - Draw an initial-state distribution $\pi_0 \sim \text{Dir}(\frac{\gamma}{p}\mathbf{1}_p)$ or [all start from the null state]
  - For each input variable index $j = 0, 1, \ldots, p$, draw transition distribution $\theta_j \sim \text{Dir}(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha} \in \mathbb{R}^{p+1}$ and $\sum_{j=0}^{p} \alpha_j = 1$. $\Theta = \left[(1, 0, \ldots, 0)^T; \theta_1^T; \ldots; \theta_p^T\right]$.
  - Draw the number of random features $D \sim \text{Unif}(1, 2, \ldots, \infty)$
- Local variables:

  For each random feature $l \in \{1, 2, \ldots, D\}$:
  - Draw a Markov chain of atoms for combination of variable indices $\mathbf{z}_l \sim \text{MC}(\pi_0, \Theta)$. That is,

$$\mathbb{P}(z_{l,t} = j' | z_{l,t-1} = j, \pi_0, \Theta) = \theta_{j,j'}, \quad j, j' \in \{0, 1, \ldots, p\}, \ t = 1, 2, \ldots$$

- The active feature set is expanded as $\mathcal{A}_t = \mathcal{A}_{t-1} \bigcup \{x_{z_{l,t}} \otimes \mathcal{A}_{t-1}\}$, that is, include all the interaction terms of the newly selected feature and the features in the existing candidate set. Expand the candidate set by including the interaction term
$$u_{l,t} = \prod_{\tau=1}^{t} x_{z_{l,\tau}}$$
- Draw the distribution for random frequencies (or embedding coefficients)
  - $\mu_l \sim \mathcal{N}(\mu_0, \Sigma_0)$
  - $\Sigma_l \sim \mathcal{W}^{-1}(\Psi_0, \nu_0)$
- Draw the corresponding random frequency vector $\boldsymbol{\omega}_{l,z_l} \sim \mathcal{N}(\mu_{l,z_l}, \Sigma_{l,z_l,z_l})$

- Response variable:
  - For each data point index $i = 1, \ldots, n$:
    - Construct the random features
$$\hat{\phi}(\mathbf{x}_i) = \sqrt{\frac{2}{D}} \left(\cos(\boldsymbol{\omega}_1^T \mathbf{x}_i), \ldots, \cos(\boldsymbol{\omega}_D^T \mathbf{x}_i)\right)$$
    - Construct discriminant function $f(\cdot)$ as in (4.3).
    - Draw the binary response variable $Y_i \sim \text{Bernoulli}\left(\frac{1}{1+\exp(-f(\mathbf{x}_i))}\right)$

We can learn the optimal random features, and thus the corresponding kernel function, by using Markov Chain Monte Carlo (MCMC) to obtain the posterior distribution $p(\omega_{1,\ldots,D}, D|X, Y)$. However, since the number of random features can be infinite, the dimensionality of feature space is indefinite. To resolve this difficulties in inference, we can apply the *Reversible-Jump MCMC* [Green, 1995].

4.2.4 Connection to Deep Learning

By collecting the randomly sampled ω's in a matrix $W = (\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_D)^T$, it is easy to see that $\phi(W\mathbf{x})$ is a neural network module, consisting of a linear layer $W\mathbf{x}$ and

entry-wise nonlinearities introduced by the cosine function. Such neural network corresponds to a specific kernel function.

4.3 Conclusion

In this thesis, we proposed two methods for flexible sparse learning of feature subspaces. In Chapter 2, the sparse learning is constructed based on the distance metric learning framework. In the pursuit of sparse distance metric, we are able to extract feature subspaces that maximize the margin between classes in local neighborhoods. In this method, flexibility is achieved by two approaches: the exploitation of local information and the exploration of potential polynomial features via nonlinear hierarchical expansion. In Chapter 3, we focus on the sparse learning in online setting based on the stochastic gradient descent algorithm. After identifying the unstable nature of SGD algorithm on highly sparse data, we improve its flexibility by employing informative truncation to accommodate heterogeneous sparsity levels of features. Stability selection and annealing rejection rate are applied to enhance efficiency. At last, an extension to kernel machine is briefly discussed in Chapter 4 for further possibilities in even more flexible solutions with tractable computations. In all proposed methods, we focus on obtaining generalizable sparse solutions while maintaining tractable computational cost. Flexibility is highly valued in our designs. We strive to devise algorithms that are nonparametric, nonlinear and adaptive to local variations. Our approaches are demonstrated to be effective and computationally efficient in several simulation studies and real data analysis.

Bibliography

[Aronszajn, 1950] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, pages 337–404, 1950.

[Beck and Teboulle, 2003] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[Belkin and Niyogi, 2002] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, pages 585–591, 2002.

[Bellet *et al.*, 2013] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[Bellman, 1961] R.E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton Legacy Library. Princeton University Press, 1961.

[Bi *et al.*, 2011] Jinbo Bi, Dijia Wu, Le Lu, Meizhu Liu, Yimo Tao, and Matthias Wolf. Adaboost on low-rank psd matrices for metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2617–2624. IEEE, 2011.

[Blitzer *et al.*, 2005] John Blitzer, Kilian Q Weinberger, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1473–1480, 2005.

[Bottou, 1998] Léon Bottou. Online learning and stochastic approximations. *Online Learning in Neural Networks*, 17(9):142, 1998.

[Bottou, 2010] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, page 177. Springer Science & Business Media, 2010.

[Bousquet and Elisseeff, 2002] Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

[Boyd and Vandenberghe, 2004] Stephen Poythress Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[Breiman, 2001] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[Bühlmann and Yu, 2006] Peter Bühlmann and Bin Yu. Sparse boosting. *The Journal of Machine Learning Research*, 7:1001–1024, 2006.

[Cai and He, 2012] Deng Cai and Xiaofei He. Manifold adaptive experimental design for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):707–719, 2012.

[Carpenter, 2008] Bob Carpenter. Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. *Alias-i, Inc., Technical Report*, pages 1–20, 2008.

[Cesa-Bianchi *et al.*, 2004] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

[Cohen, 1960] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[Cortes and Vapnik, 1995] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[Cover and Hart, 1967] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[Cowling, 1983] Michael G Cowling. Harmonic analysis on semigroups. *Annals of Mathematics*, pages 267–283, 1983.

[Craven and Wahba, 1978] Peter Craven and Grace Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, 31(4):377–403, 1978.

[Dai *et al.*, 2014] Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049, 2014.

[Ding and Peng, 2005] Chris Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, 3(02):185–205, 2005.

[Duchi and Singer, 2009] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

[Duchi *et al.*, 2010] John C Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In *Proceedings of the Twenty Third Annual Conference on Computational Learning Theory*, pages 14–26, 2010.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[Fei-Fei and Perona, 2005] Li Fei-Fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531. IEEE, 2005.

[Fisher, 1936] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[Freund, 1995] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[Friedman and others, 1991] Jerome H Friedman et al. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67, 1991.

[Friedman, 1994] Jerome H Friedman. Flexible metric nearest neighbor classification. *Technical Report*, 113, 1994.

[Friedman, 2001] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[Friedman, 2002] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[Goldberger *et al.*, 2005] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, pages 513–520. MIT Press, 2005.

[Golub *et al.*, 1999] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R

Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

[Green *et al.*, 1994] Peter J Green, Bernard W Silverman, Bernard W Silverman, and Bernard W Silverman. *Nonparametric regression and generalized linear models: a roughness penalty approach*. Chapman & Hall London, 1994.

[Green, 1995] Peter J Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[Guyon and Elisseeff, 2003] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(Mar):1157–1182, 2003.

[Guyon *et al.*, 2004] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in Neural Information Processing Systems*, pages 545–552, 2004.

[Guyon *et al.*, 2006] Isabelle Guyon, Jiwen Li, Theodor Mader, Patrick A Pletscher, Georg Schneider, and Markus Uhr. Feature selection with the clop package. Technical report, Technical Report, 2006.

[Guyon *et al.*, 2007] Isabelle Guyon, Jiwen Li, Theodor Mader, Patrick A Pletscher, Georg Schneider, and Markus Uhr. Competitive baseline methods set new standards for the nips 2003 feature selection benchmark. *Pattern recognition letters*, 28(12):1438–1444, 2007.

[Hardt *et al.*, 2015] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

[Hastie *et al.*, 2009] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The Elements of Statistical Learning*, volume 2. Springer, 2009.

[Haury *et al.*, 2011] Anne-Claire Haury, Pierre Gestraud, and Jean-Philippe Vert. The influence of feature selection methods on accuracy, stability and interpretability of molecular signatures. *PloS one*, 6(12):e28210, 2011.

[Hinton and Roweis, 2002] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, pages 833–840, 2002.

[Hofmann *et al.*, 2008] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The Annals of Statistics*, pages 1171–1220, 2008.

[Hong *et al.*, 2011] Yi Hong, Quannan Li, Jiayan Jiang, and Zhuowen Tu. Learning a mixture of sparse distance metrics for classification and dimensionality reduction. In *IEEE International Conference on Computer Vision (ICCV)*, pages 906–913. IEEE, 2011.

[Huang *et al.*, 2009] Kaizhu Huang, Yiming Ying, and Colin Campbell. Gsml: A unified framework for sparse metric learning. In *Ninth IEEE International Conference on Data Mining*, pages 189–198. IEEE, 2009.

[Hughes, 1968] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, Jan 1968.

[Jain *et al.*, 2012] Prateek Jain, Brian Kulis, Jason V Davis, and Inderjit S Dhillon. Metric and kernel learning using a linear transformation. *Journal of Machine Learning Research*, 13:519–547, 3 2012.

[Jia *et al.*, 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[Johnson and Lindenstrauss, 1984] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26(189-206):1, 1984.

[Kedem *et al.*, 2012] Dor Kedem, Stephen Tyree, Fei Sha, Gert R Lanckriet, and Kilian Q Weinberger. Non-linear metric learning. In *Advances in Neural Information Processing Systems*, pages 2573–2581, 2012.

[Kursa *et al.*, 2010] Miron B Kursa, Witold R Rudnicki, et al. Feature selection with the boruta package. *Journal of Statistical Software*, 36(i11), 2010.

[Kutin and Niyogi, 2002] Samuel Kutin and Partha Niyogi. Almost-everywhere algorithmic stability and generalization error. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 275–282. Morgan Kaufmann Publishers Inc., 2002.

[Langford *et al.*, 2009] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. In *Advances in Neural Information Processing Systems*, pages 905–912, 2009.

[Lazebnik *et al.*, 2006] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[Le Cun *et al.*, 1990] B Boser Le Cun, John S Denker, D Henderson, Richard E Howard, W Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a

back-propagation network. In *Advances in Neural Information Processing Systems*, 1990.

[Lichman, 2013] M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013. University of California, Irvine, School of Information and Computer Sciences.

[Lions and Mercier, 1979] Pierre-Louis Lions and Bertrand Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

[Liu *et al.*, 2010] Wei Liu, Shiqian Ma, Dacheng Tao, Jianzhuang Liu, and Peng Liu. Semi-supervised sparse metric learning using alternating linearization optimization. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1139–1148. ACM, 2010.

[Lowe, 2004] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[Marimont and Shapiro, 1979] R. B. Marimont and M. B. Shapiro. Nearest neighbour searches and the curse of dimensionality. *IMA Journal of Applied Mathematics*, 24(1):59–70, 1979.

[Meinshausen and Bühlmann, 2010] Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

[Nadaraya, 1964] Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

[Natarajan, 1995] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[Nesterov, 2009] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

[Oiwa *et al.*, 2011] Hidekazu Oiwa, Shin Matsushima, and Hiroshi Nakagawa. Frequency-aware truncated methods for sparse online learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 533–548. Springer, 2011.

[Oiwa *et al.*, 2012] Hidekazu Oiwa, Satoru Matsushima, and Hirotoshi Nakagawa. Healing truncation bias: self-weighted truncation framework for dual averaging. In *IEEE 12th International Conference on Data Mining (ICDM)*, pages 575–584. IEEE, 2012.

[Parkhi *et al.*, 2012] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[Pearson, 1901] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559–572, 1901.

[Qi *et al.*, 2009] Guo-Jun Qi, Jinhui Tang, Zheng-Jun Zha, Tat-Seng Chua, and Hong-Jiang Zhang. An efficient sparse metric learning in high-dimensional space via l 1-penalized log-determinant regularization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 841–848. ACM, 2009.

[Rahimi and Recht, 2007] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

[Rakhlin *et al.*, 2005] Alexander Rakhlin, Sayan Mukherjee, and Tomaso Poggio. Stability results in learning theory. *Analysis and Applications*, 3(04):397–417, 2005.

[Rosales and Fung, 2006] Rómer Rosales and Glenn Fung. Learning sparse metrics via linear programming. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 367–373. ACM, 2006.

[Roweis and Saul, 2000] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[Rudin, 2011] Walter Rudin. *Fourier analysis on groups*. John Wiley & Sons, 2011.

[Schiilkopf, 2001] Bernhard Schiilkopf. The kernel trick for distances. *Advances in Neural Information Processing Systems*, 13:301–307, 2001.

[Shalev-Shwartz and Tewari, 2011] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l 1-regularized loss minimization. *The Journal of Machine Learning Research*, 12:1865–1892, 2011.

[Shalev-Shwartz *et al.*, 2010] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *The Journal of Machine Learning Research*, 11:2635–2670, 2010.

[Shen *et al.*, 2009] Chunhua Shen, Junae Kim, Lei Wang, and Anton Hengel. Positive semidefinite metric learning with boosting. In *Advances in Neural Information Processing Systems*, pages 1651–1659, 2009.

[Sivic and Zisserman, 2003] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Ninth IEEE International Conference on Computer Vision*, pages 1470–1477. IEEE, 2003.

[Suarez *et al.*, 2014] Ranyart R Suarez, Jose Maria Valencia-Ramirez, and Mario Graff. Genetic programming as a feature selection algorithm. In *IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pages 1–5. IEEE, 2014.

[Sun *et al.*, 2013] Wei Sun, Junhui Wang, and Yixin Fang. Consistent selection of tuning parameters via variable selection stability. *The Journal of Machine Learning Research*, 14(1):3419–3440, 2013.

[Tenenbaum *et al.*, 2000] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

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

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

[Torresani and Lee, 2006] Lorenzo Torresani and Kuang-chih Lee. Large margin component analysis. In *Advances in Neural Information Processing Systems*, pages 1385–1392, 2006.

[Toulis *et al.*, 2015] Panos Toulis, Dustin Tran, and Edoardo M Airoldi. Stability and optimality in stochastic gradient descent. *arXiv preprint arXiv:1505.02417*, 2015.

[Turki and Roshan, 2014] Turki Turki and Usman Roshan. Weighted maximum variance dimensionality reduction. In *Pattern Recognition*, pages 11–20. Springer, 2014.

[Tusher *et al.*, 2001] Virginia Goss Tusher, Robert Tibshirani, and Gilbert Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences*, 98(9):5116–5121, 2001.

[Vasilescu and Terzopoulos, 2003] M Alex O Vasilescu and Demetri Terzopoulos. Multilinear subspace analysis of image ensembles. In *IEEE Computer Society Con-*

ference on Computer Vision and Pattern Recognition, volume 2, pages II–93. IEEE, 2003.

[Wang *et al.*, 2011] Jun Wang, Huyen T Do, Adam Woznica, and Alexandros Kalousis. Metric learning with multiple kernels. In *Advances in Neural Information Processing Systems*, pages 1170–1178, 2011.

[Wang *et al.*, 2014] Jialei Wang, Peilin Zhao, Steven CH Hoi, and Rong Jin. Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):698–710, 2014.

[Wang *et al.*, 2015] Dayong Wang, Pengcheng Wu, Peilin Zhao, and Steven CH Hoi. A framework of sparse online learning and its applications. *arXiv preprint arXiv:1507.07146*, 2015.

[Wu *et al.*, 2014] Yue Wu, Steven CH Hoi, Tao Mei, and Nenghai Yu. Large-scale online feature selection for ultra-high dimensional sparse data. *arXiv preprint arXiv:1409.7794*, 2014.

[Xiao, 2009] Lin Xiao. Dual averaging method for regularized stochastic learning and online optimization. In *Advances in Neural Information Processing Systems*, pages 2116–2124, 2009.

[Xing *et al.*, 2002] Eric P Xing, Michael I Jordan, Stuart Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, pages 505–512, 2002.

[Yang and Jin, 2006] Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, pages 1–51, 2006.

[Ying *et al.*, 2009] Yiming Ying, Kaizhu Huang, and Colin Campbell. Sparse metric learning via smooth optimization. In *Advances in Neural Information Processing Systems*, pages 2214–2222, 2009.

[Yuan and Zhang, 2013] Xiao-Tong Yuan and Tong Zhang. Truncated power method for sparse eigenvalue problems. *The Journal of Machine Learning Research*, 14(1):899–925, 2013.

[Zhang *et al.*, 2006] Xuegong Zhang, Xin Lu, Qian Shi, Xiu-qin Xu, E Leung Honchiu, Lyndsay N Harris, James D Iglehart, Alexander Miron, Jun S Liu, and Wing H Wong. Recursive svm feature selection and sample classification for mass-spectrometry and microarray data. *BMC bioinformatics*, 7(1):1, 2006.

[Zhang, 2004] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004.

[Zhu *et al.*, 2009] Ji Zhu, Hui Zou, Saharon Rosset, and Trevor Hastie. Multi-class adaboost. In *Statistics and its Interface*. Citeseer, 2009.

[Zinkevich *et al.*, 2010] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 2595–2603, 2010.