FEATURE ENGINEERING FOR DATA ANALYTICS

by

Ke Xu

APPROVED BY SUPERVISORY COMMITTEE:

Haim Schweitzer, Chair

Farokh B. Bastani

Latifur Khan

Murat Kantarcioglu

FEATURE ENGINEERING FOR DATA ANALYTICS

by

KE XU, BS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2017

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Dr. Haim Schweitzer for his continuous support of my PhD study. It has been a great time working with him. He is the funniest advisor and one of the smartest people I know. He has been supportive and has given me the freedom to do whatever I like. When I run into problems he is always there to answer all my questions. He has provided insightful discussion about the research and he cares about the students very much. He is the reason that I could finish my degree. I always feel so lucky to have him as my advisor.

I would like to thank my thesis committee Dr. Farokh B. Bastani, Dr. Latifur Khan and Dr. Murat Kantarcioglu for their insightful comments and feedback.

I thank Hiromasa Arai, Crystal Maung, Tongyi Cao, Swair Shah and Baokun He for the joint work we have done and all the fun we have had during the past four years.

To my parents for their unconditional love over all these years. I would not have made it this far without them, and I thank them for their support and care.

Thanks to my friends for always being supportive and having faith in me. A special thanks to Feifei Zhu for always cheering me up and making me happy.

September 2017

FEATURE ENGINEERING FOR DATA ANALYTICS

Ke Xu, PhD
The University of Texas at Dallas, 2017

Supervising Professor: Haim Schweitzer

Data plays a fundamental role in modern science, engineering, and business applications. We investigate two important problems in data analytics. The first is feature selection, where we consider both the unsupervised and the supervised case. The second is data privacy, where we propose a new model and describe algorithms that improve data privacy in that model.

Feature selection is the process of removing redundant and irrelevant features from the data. There are two general classes of feature selection: the unsupervised case and the supervised case. In the unsupervised case features are selected to approximate the entire data matrix, while in the supervised case features are selected to predict a set of labels from the data matrix.

We describe several new algorithms for the unsupervised case that are closely related to the A^* heuristic search algorithm. These new algorithms can effectively select features from large datasets and are shown experimentally to be more accurate than the current state of the art. The evaluation criterion for feature selection is typically an error measured in the Frobenius norm. We generalize the criteria to a large family of unitarily invariant norms. These include, among others, the Spectral norm, the Nuclear norm, and Schatten p-norms.

We proposed several algorithms for supervised feature selection that improve the running time and the accuracy of the current state of the art. A common approach for reducing the running time is to perform the selection in two stages. In the first stage a fast filter is applied to select good candidates. The number of candidates is further reduced in the second stage by an accurate algorithm that may

run significantly slower. We describe a general framework that can use an arbitrary off-the-shelf unsupervised algorithm for the second stage. The algorithm is applied to the selection obtained in the first stage weighted appropriately.

Another common approach for accelerating the running time of feature selection is to use a greedy technique called "forward selection". We show how to use this technique to address the multi-label classification problem. Experimental results on real-world data demonstrate the effectiveness of the proposed approach.

We generalize the error criteria of forward selection to unitarily invariant functions. In particular we show how to minimize Schatten p-norms to solve the outlier-robust PCA problem. The algorithm is very efficient and experimental results show that it outperforms a well-known outlier-robust PCA method that uses convex optimization.

In standard machine learning and regression, feature values are used to predict some desired information from the data. One privacy concern is that the feature values may also expose information that one wishes to keep confidential. We propose a model that formulates this concern, and we show that such privacy can be achieved with almost no effect on the quality of predicting desired information. We describe two algorithms for the case in which the prediction model starts with a linear operator. The desired effect can be achieved by zeroing out feature components in the approximate null space of the linear operator.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 The problems being addressed

Data plays a fundamental role in modern science, engineering, and business applications. We investigate two important problems of data analytics: feature selection and data privacy. Our work on feature selection includes the development of several new algorithms that compare favorably with the current state of the art. Our contribution to data privacy includes the proposal of a new privacy model and algorithms that improve the privacy in that model.

1.1.1 Feature selection

Feature selection is a standard dimensionality reduction technique. See, e.g., (Guyon and Elisseeff, 2003; Fan and Lv, 2010; Chandrashekar and Sahin, 2014). Given data items described in terms of n features, the goal is to select $k<n$ features such that the reduced k dimensional vectors are useful for the task at hand.

The unsupervised feature selection technique gives a sparse approximation to the data matrix, and has found applications in many areas. These include the computation of stable and rank revealing QR factorizations (e.g., (Golub and Van-Loan, 2013; Gu and Eisenstat, 1996; Boutsidis et al., 2009)), feature selection in machine learning (e.g., (Drineas et al., 2010; Maung and Schweitzer, 2013; Wei and Billings, 2007), and data mining and knowledge representation (e.g., (Dasgupta et al., 2007; Kumar et al., 2012; Drineas et al., 2010)).

In supervised feature selection the features are selected for predicting labels associated with each data item. These can be linear regression (e.g., (Hastie et al., 2009; Miller, 2002; Dahmen and DeVore, 2008; Das and Kempe, 2008), multi-target regression (e.g., (Džeroski et al., 2000; Spyromitros-Xioufis et al., 2016)) and multi-label classification (e.g., (Boutell et al., 2004; Diplaris et al., 2005; Katakis et al., 2008)).

1.1.2 Data privacy

Data analytics dgebring enormous benefits to our society in areas such as social media, business and health care, etc. But it may also affect the privacy of individuals. There are many studies that address these problems with various techniques. For example, in differential privacy noise is added to blur the distinction between individual items in the data. See, e.g., (Dwork and Roth, 2014; Sarwate and Chaudhuri, 2013), Several studies investigate feature selection as a tool for obtaining privacy of database. See, e.g., (Pattuk et al., 2015; Jafer et al., 2015; Banerjee and Chakravarty, 2011). Recent studies (Enev et al., 2012; Hamm, 2015; Whitehill and Movellan, 2012) consider a setting that makes the distinction between desired and confidential information. In (Hamm, 2015), the ally (data aggregator) and the user (data contributor) participate in the data filtering. The goal is to prevent an adversary from predicting the private information of users while maintaining the utility of the data. The framework proposed in (Enev et al., 2012) transforms the data such that the covariance between the data and the desired information is increased, while the covariance between the data and confidential information is decreased.

1.2 Our contributions

1.2.1 Effective feature selection

We investigate several algorithms that enable effective selection of features from data for both the unsupervised and the supervised case. We describe new algorithms for the column subset selection problems that are motivated by the classic A* approach in heuristic search. Some of our algorithms are optimal but slow, and others are not guaranteed to find an optimal solution but run much faster than optimal algorithms (Arai et al., 2015). Experimental results show that these new algorithms are more accurate than the current state of the art while still being practical.

A common approach for reducing the run time of feature selection is to perform it in two stages. In the first stage candidate features are selected. These candidates are evaluated more thoroughly in

a second stage. See, e.g., (Guyon and Elisseeff, 2003; Dasgupta et al., 2007; Boutsidis et al., 2009). In supervised feature selection the second stage is typically supervised (and slow). We describe a general approach that enables combining two off-the-shelf algorithms to perform the selection where the second stage is unsupervised. The main idea is to apply the unsupervised algorithm to the selection obtained in the first stage weighted appropriately. Our main technical result is the computation of these weights. The method is shown experimentally to be almost as accurate as using supervised algorithms, while at the same time it may be significantly faster.

Another common fast approach for feature selection is named "forward selection". See, e.g., (Chen et al., 1989; Gharavi-Alkhansari and Huang, 1998; Rebollo-Neira and Lowe, 2002). We describe how to use this greedy approach to effectively select features for multi-label classification. The proposed algorithm is further generalized by adding monomial kernels which depend on a single variable. Experimental results on real-world data demonstrate the effectiveness of the proposed approaches.

1.2.2 Error criteria generalization

A common error evaluation criterion for feature selection is the Frobenius norm. See, e.g., (Frieze et al., 2004; Çivril and Magdon-Ismail, 2012; Paul et al., 2015). We generalize the error criterion to a large family of unitarily invariant norms. These include, among others, the Spectral norm, the Nuclear norm, and Schatten p-norms. We show how to obtain optimal subset selection under such error criteria. We generalize the classical forward selection algorithm so that it can be used with the various unitarily invariant functions. In particular, we show minimizing Schatten p-norms to solve the outlier-robust PCA problem. The algorithm is very efficient and experimental results show that it outperforms a well-known outlier-robust PCA method that uses convex optimization.

1.2.3 Privacy mechanism for predictors

The privacy concern we address is the potential inappropriate use of data to predict confidential information. Adapting machine learning terminology of a learning and a testing phase, our goal is

protecting the privacy of information during the testing phase. It is different from studies concerned with the privacy of training data, such as differential privacy.

The proposed algorithms can be applied to predictors that start with a linear operator. They are different from other recent studies that have different (or no) assumptions about the predictors. See, e.g., (Enev et al., 2012; Hamm, 2015; Whitehill and Movellan, 2012). The main observation is that projections on the null space of the predictors do not change the prediction value. Thus the desired effect can be achieved by zeroing out feature components in the approximate null space of the linear operator.

1.3 Dissertation organization

The dissertation is organized as follows. Chapters 2 and 3 introduce the main problems being addressed. Chapter 2 describes the problem of feature selection and Chapter 3 introduces privacy considerations in data analytics. In Chapters 4, 6 and 7, we describe algorithms for the effective feature selection. These include the approaches related to weighted A^* heuristic search, two-stage algorithm and generalized forward selection algorithm for multi-label classification. In Chapters 5 and 8, we show the generalization of evaluation criteria to unitarily invariant norms for both unsupervised feature selection and supervised feature selection. The approach for enhancing the data privacy is illustrated in Chapter 9.

CHAPTER 2

THE FEATURE SELECTION PARADIGM

Feature selection is a known problem in machine learning, data mining and knowledge representation. The goal is to identify and remove redundant and irrelevant features from data. Consider the simple example shown in Table 2.1. Features include height in centimeter, height in feet, age, and weight in pounds. Body mass index is the label to be predicted. Notice that the two features related to height can be converted to each other. Thus one of the height feature is redundant and can be removed. Another key point is that BMI depend only on the height and the weight of an individual. Therefore the age feature is irrelevant. In practice, identifying the redundant and irrelevant features is not as easy as this example. In many situations, the optimal solution is known to be NP-hard (Çivril, 2017; Shitov, 2017). Feature selection provides several important benefits as following:

- It simplifies the prediction model which makes it easier to interpret.
- It enables faster and more efficient prediction.
- In practice, it gives better generalization properties than using the entire feature set.

2.1 Problem formulation

Let Y be a data matrix of m rows and N columns. Its columns are $y_1, \ldots, y_N$, where a column y_i is an m dimensional vector. Similarly, let X be a "dictionary" matrix of m rows and n columns. Its columns, also called "atoms", are $x_1, \ldots, x_n$, where a column x_i is an m dimensional vector. We consider the problem of selecting a subset of k columns from X that can be used to approximate all the columns of Y . Specifically, suppose the columns $x_{s_1}, \ldots, x_{s_k}$ are selected from X, then each column y_i of Y can be approximated by the linear combination:

$$y_i \approx a_{i,1}x_{s_1} + \ldots + a_{i,k}x_{s_k} \tag{2.1}$$

Table 2.1: Example of feature selection.

Name	Height (cm)	Height (ft)	Age	Weight (lb)	Body Mass Index (BMI)
Lucy	160	5.24	17	98	17.4
Alexa	173	5.67	26	121	18.4
Jack	185	6.07	40	174	23

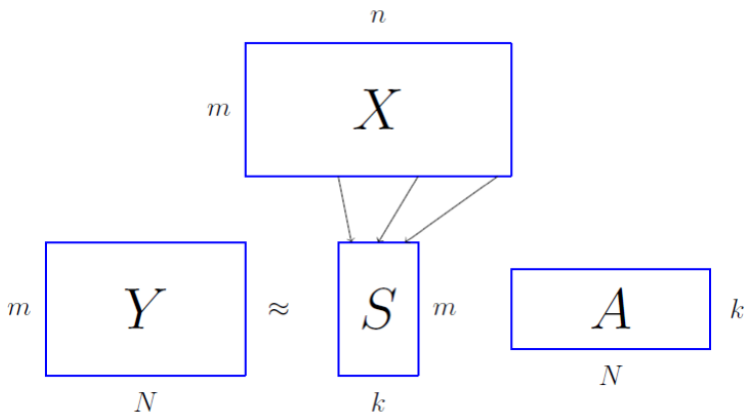


Figure 2.1: Feature selection illustration. The data matrix Y is $m \times N$, the dictionary matrix X is $m \times n$, and the selection matrix S is $m \times k$.

Here $a_{i,1}, \ldots, a_{i,k}$ are coefficients associated with y_i and chosen to minimize the approximation error. In matrix form this can be written as:

$$Y \approx SA \tag{2.2}$$

Here $S = (x_{s_1}, \ldots, x_{s_k})$ is the $m \times k$ selection matrix created from the k columns of X, and the coefficients matrix A is $k \times N$. The quality of the selection S is determined by the following error:

$$E = \min_{A} \|Y - SA\|_F^2 \tag{2.3}$$

where the matrix norm is the Frobenius norm. We are interested in the case where n (the dictionary size) is big and k (the selection size) is small, as illustrated in Figure 2.1.

2.2 Unsupervised and supervised feature selection

There are two general classes of feature selection: the unsupervised and the supervised case.

The unsupervised case has X=Y, taking the dictionary to be the same as the data matrix. Here the challenge is to approximate the data matrix Y by a small subset of its columns. This is called "unsupervised feature selection" in machine learning (e.g. (Hastie et al., 2009; Maung and Schweitzer, 2013)) and "column subset selection" in computational linear algebra (e.g. (Golub and Van-Loan, 2013)). The most important application in numerical linear algebra is the computation of a stable QR factorization, which can be achieved by computing the orthogonal matrix Q only from the selected column subset (Golub and Van-Loan, 2013). Using this framework for unsupervised feature selection was shown to be useful in areas such as text classification (Dasgupta et al., 2007) and the analysis of microarray data (Mahoney and Drineas, 2009).

In the supervised case X is not the same as Y. Referring to the matrix X as a dictionary is common in the signal processing literature where in the heavily analyzed case of N=1 the matrix Y is called "the signal". Applications include wavelet decomposition (e.g. (Mallat, 1999; Pati et al., 1993)), image processing (e.g. (Bergeaud and Mallat, 1995)), and video coding (e.g. (Neff and Zakhor, 1997)). The case N=1 is also common in machine learning and related studies, where it is known as feature selection in linear regression (e.g. (Hastie et al., 2009; Miller, 2002; Dahmen and DeVore, 2008; Das and Kempe, 2008)). Detailed analysis appears, for example, in (Mao, 2004). Applications to finance are described in (Fan and Lv, 2010). The case $N > 1$ is known as multi-label learning where each sample is associated with a set of labels. It has attracted a lot of interest in recent years. The applications include image annotation (Boutell et al., 2004), protein function classification (Diplaris et al., 2005) and text categorization (Katakis et al., 2008).

CHAPTER 3

THE PRIVACY PARADIGM

There are many who consider the "sensor revolution" as *the next big thing* (Hardy, 2016). Cyber-physical systems surround us with sensors that collect data. The data can then be analyzed, and provide useful information that can help improve many aspects of our lives. The drawback is that the same data may reveal sensitive information that one may wish to keep private.

Informally, privacy considerations in cyber-physical systems can be taken as the need to protect information about humans from unauthorized access by humans or machines. We argue here that there are some fundamental differences between this and the privacy of datasets, where the goal is to keep private the data of individual users in the dataset, while allowing the mining of useful information from the dataset (Dwork and Roth, 2014). These fundamental differences suggest new approaches to enhance sensor data privacy that are very different from approaches that were developed for datasets privacy.

Consider the flow of information collected with sensors, as illustrated in Figure 3.1. Sensors obtain information about humans, and then transmit that information to a data collection center. An analysis is then performed on data obtained from the Data Collection Center. There are different privacy considerations depending on where the data may be intercepted along the flow diagram, how much data is kept at the Data Collection Center, and for how long the data is maintained. Three cases are discussed below.

Case 1: Data collected from many individuals.

In this case the data is intercepted between the Data Collection Center and the Analysis module. Privacy violation is the exposure of sensor data from any particular individual. This is essentially the same as the privacy problem in datasets, and can be approached using similar tools. In particular, the data can be "blurred" to the point where it is impossible to distinguish particular individuals, but the data can still be used to mine relevant statistical information. As an example,

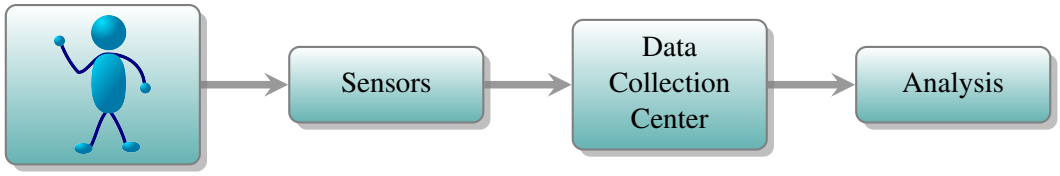


Figure 3.1: The flow of information collected with sensors

smart electricity meters measure electricity consumption at small time intervals. In this case the Data Collection Center is the electricity provider. The privacy requirement is to prevent an adversary from learning about the habits of the occupants of any particular house. For this and other examples see, e.g., (McDaniel and McLaughlin, 2009; Sarwate and Chaudhuri, 2013; Cortés et al., 2016; Kerschbaum and Strüker, 2012).

Case 2: Data collected from a single individual, combined with external datasets.

In this case the data can be intercepted anywhere between the sensors module and the Analysis module. Privacy violation occurs when it is possible to identify the particular individual in another, external dataset, which contains information about many individuals. Standard approaches for this case are the encryption of the sensor data, or making sure that the external datasets are anonymized. See for example (Koufogiannis and Pappas, 2016), where GPS data is combined with external data to identify the individual driving the car.

Case 3: Data collected from a single individual, no external datasets.

As in Case 2 the data can be intercepted anywhere between the sensors module and the Analysis module. However, the violation of privacy in this case is different. It occurs when an adversary can use external information to build a model that enables inferring confidential information about the individual from the sensor data. The main difference between Case 3 and cases 1,2 is that the external information does not come from the same individual who is being examined. Still, the external information can be used as "training data" in a machine learning setting, and enable accurate prediction of confidential information.

As an example consider the case of electroencephalogram (EEG) measurements. It is known that epileptic seizures can be predicted from such measurements, a discovery which has the potential of significantly improving the lives of 1% of the world population (Altunay et al., 2010; Carney et al., 2011). However, it was recently found that the same EEG measurements can also be used to detect schizophrenia. Even worse, they may be used to predict who is likely to develop schizophrenia (Johannesen et al., 2016). Clearly, epileptic individuals may be concerned about allowing others to learn that they "may" develop schizophrenia, to the extent where they may decide not to use EEG to predict their epileptic seizures.

Cases 2 and 3 may appear similar, as both use sensor measurements from a single individual, and look for intersections of these measurements with an external dataset. However, the privacy consideration is very different. In Case 2 it has to do with identifying an individual in a dataset, while in Case 3 the particular dataset that was used to train the classifier is irrelevant.

There are many studies that address privacy in situations similar to cases 1 and 2. It appears that the tools used to achieve privacy in these cases are similar to those used in the investigation of dataset privacy. In particular, this includes differential privacy and cryptographic techniques. There are very few studies in dataset privacy that can be applied to Case 3. We investigate possible approaches to enhance the privacy in Case 3.

The reason why Case 3 is so different from cases 1,2 is that it involves no explicit dataset. The fundamental idea behind dataset privacy is that statistical information can be extracted without exposing personal information. But there can be no "statistics" of a single measure, which means that addressing privacy in Case 3 requires an entirely different approach. We discuss a mechanism aimed at providing such privacy in Chapter 9.

CHAPTER 4

COLUMN SUBSET SELECTION WITH PROVABLE BOUNDS ON SUBOPTIMALITY BY HEURISTIC SEARCH

Identifying a small number of features that can represent the data is a known problem that comes up in areas such as machine learning, knowledge representation, data mining, and numerical linear algebra. Computing an optimal solution is believed to be NP-hard, and there is extensive work on approximation algorithms. Classic approaches exploit the algebraic structure of the underlying matrix, while more recent approaches use randomization. An entirely different approach that uses the A* heuristic search algorithm to find an optimal solution was recently proposed. Not surprisingly it is limited to effectively selecting only a small number of features. We propose a similar approach related to the Weighted A^* algorithm. This gives algorithms that are not guaranteed to find an optimal solution but run much faster than the A^* approach, enabling effective selection of many features from large datasets. We demonstrate experimentally that these new algorithms are more accurate than the current state-of-the-art while still being practical. Furthermore, they come with an adjustable guarantee on how different their error may be from the smallest possible (optimal) error. Their accuracy can always be increased at the expense of a longer running time.

Portions of this chapter are the results of collaboration with Hiromasa Arai and Crystal Maung. The results are published in (Arai et al., 2016).

4.1 Introduction

Feature selection is a standard dimensionality reduction technique. Given data items described in terms of n features, the goal is to select $k<n$ features such that the reduced k dimensional vectors are useful for the task at hand. In supervised feature selection the features are selected for predicting values associated with each data item. These can be classification labels (e.g., (Guyon and Elisseeff, 2003)), or multi-valued regression (e.g., (Maung and Schweitzer, 2015)). In unsupervised feature selection the standard criterion is to select k features that can be used to approximate

all n features. There are two common approaches for computing the unsupervised selection. The first is to partition the feature space into k clusters and then select a representative from each cluster (e.g., (Dy et al., 2003; Li et al., 2012)). This assumes a feature to be redundant if another feature similar to it has already been selected. The second common approach assumes that a feature is redundant if it can be approximated by a linear combination of other features that have already been selected (e.g., (Jimenez-Rodriguez et al., 2007; Wei and Billings, 2007; Drineas et al., 2010; Dasgupta et al., 2007; Wang et al., 2015)). Thus, if the goal is to select features for a nearest neighbor algorithm the first approach seems most appropriate. But if the selected features are to be used in a Gaussian classifier, the second approach seems more appropriate (Duda et al., 2001). In this dissertation we consider only the second approach, which is formally described as follows.

Let X be an $m \times n$ data matrix of m items (rows), each described in terms of n features (columns). We wish to approximate X by a linear combination of k of its columns: $X \approx S_k A$. The matrix $S_k = (x_{s_1}, \ldots, x_{s_k})$ is formed by the selected columns, and A is the coefficients matrix. In Frobenius norm the approximation error is:

$$\text{Err}(S_k) = \min_A \|X - S_k A\|_F^2 \tag{4.1}$$

Finding S_k that minimizes (4.1) is also known as the "*Column Subset Selection Problem*" (CSSP) in numerical linear algebra. Previous studies of the CSSP with the Frobenius norm error include (Wei and Billings, 2007; Golub and Van-Loan, 2013; Boutsidis et al., 2009; Çivril and Magdon-Ismail, 2012; Çivril, 2014). Obtaining approximations in other norms, such as the spectral norm or the l_1 norm is considered to be harder (e.g., (Tropp, 2004)).

Unsupervised feature selection gives a sparse approximation to the data matrix, and has found applications in many areas. These include the computation of stable and rank revealing QR factorizations (e.g. (Golub and Van-Loan, 2013; Gu and Eisenstat, 1996; Boutsidis et al., 2009)), feature selection in machine learning (e.g. (Drineas et al., 2010; Maung and Schweitzer, 2013; Wei and Billings, 2007), remote sensing (e.g., (Jimenez-Rodriguez et al., 2007; Wang et al., 2015)),

and data mining and knowledge representation (e.g. (Dasgupta et al., 2007; Kumar et al., 2012; Drineas et al., 2010)).

Computing the selection S_k that minimizes (4.1) is believed to be NP-hard (Çivril and Magdon-Ismail, 2009; Çivril, 2014). Most current algorithms compute an approximate solution, and the current state-of-the-art is reviewed in Section 4.2. The only exception we are aware of is a recent algorithm (Arai et al., 2015) that uses the A* algorithm to compute a guaranteed optimal solution. The algorithm gives huge runtime improvements over exhaustive search. However, as implied by the NP-hardness of the problem, it can compute effectively only a small number of features from small to moderate size datasets. Therefore, this algorithm is not a competitor to the current state-of-the-art approximation algorithms.

In this dissertation we show that heuristics similar to those proposed in (Arai et al., 2015) can be used to design much faster algorithms, that are more accurate than the current state-of-the-art. These new algorithms are not guaranteed to produce the optimal solution, but they can be applied to very large datasets and effectively select hundreds of features. Experimental evaluation shows that these algorithms are almost always more accurate than all other practical algorithms we are aware of. Furthermore, they come with an adjustable guarantee on how different their error may be from the smallest possible (optimal) error.

The chapter is organized as follows. The current state-of-the-art is reviewed in Section 4.2. The algorithm of (Arai et al., 2015) which forms the basis to our results is described in Section 4.2.3. Our main result, which extends this to the Weighted A* algorithm is described in Section 4.3. Experimental results are shown in Section 4.4.

4.2 The current state-of-the-art

Let S_k^* denote a (best) selection of k columns that minimizes the error in (4.1), and let Err(S_k^*) be the corresponding (minimal) error. Recent studies suggest that computing S_k^* is most likely NP-hard (Çivril and Magdon-Ismail, 2012; Çivril, 2014). The eigenvalues of the matrix XX^T can

be used to obtain bounds on $\text{Err}(S_k^*)$. Let Err_k^* denote the error of the best approximation of X in terms of a rank-k matrix. Then the following inequalities hold:

$$\text{Err}_k^* \leq \text{Err}(S_k^*) \leq (k+1)\text{Err}_k^* \tag{4.2}$$

Calculating Err_k^* is easy, since the columns of the best rank-k matrix are the k dominant eigenvectors of XX^T, scaled by the corresponding eigenvalues. From this it follows that:

$$\text{Err}_k^* = \sum_{t=k+1}^{m} \lambda_t$$

where λ_t is the t-largest eigenvalue of XX^T. This fact, and the left-hand side inequality in (4.2) follow from the Courant Fischer theorem (Golub and Van-Loan, 2013). The right-hand side inequality is a recent result (Deshpande et al., 2006; Deshpande and Rademacher, 2010). The same references also show that the upper bound in (4.2) is tight in the following sense: If Err_k^* is given, then the constant $k+1$ in (4.2) is the best possible.

4.2.1 Algorithms from numerical linear algebra

Subset selection algorithms were developed in numerical linear algebra for computing a stable and rank-revealing QR factorization. The algorithm that we call **QRP** performs column subset selection by computing the QR factorization of X with column pivoting. It is numerically stable and efficient, but lacks in accuracy when compared to some of the more sophisticated algorithms. See (Golub and Van-Loan, 2013; Gu and Eisenstat, 1996) for details. A more accurate variant of this algorithms applies the QRP to the "leverage" vectors. These are truncated right singular vectors of the SVD decomposition of X. It was invented by Golub, Klema and Stewart and we refer to it as the **GKS**. See (Golub and Van-Loan, 2013) for details.

Among the many other algorithms that were developed in numerical linear algebra the algorithm of Gu and Eisenstat (Gu and Eisenstat, 1996) that we call **GE** is considered the most accurate. It typically starts with a selection computed by the **QRP**, and then updates the selection, swapping columns to increase the "volume" (product of singular values squared) of the selection matrix.

4.2.2 Randomized algorithms

Randomized algorithms come with a small probability of failure. Some are very fast, and with high probability select a subset of size $O(k \log k)$ that can be shown to contain an accurate selection of size k. They can be converted into algorithms that select exactly k vectors by the approach of (Boutsidis et al., 2009). The idea is to apply a randomized algorithm in the first stage to select more than k vectors, and then apply another deterministic algorithm on the (properly weighted) selection to obtain a reduced selection of size k.

The randomized algorithm described in (Frieze et al., 2004) selects columns with probabilities proportional to their squared norms. Using it for the randomized phase and the **GE** for the deterministic phase gives the algorithm that we call **FKV-GE**.

The randomized algorithm described in (Boutsidis et al., 2009) selects columns with probability proportional to the squared norm of the corresponding truncated singular vectors of the SVD decomposition (the leverage space). Using it for the randomized phase and the **GE** for the deterministic phase gives the algorithm that we call **LEV-GE**.

An algorithm that samples k-tuples of columns with probability proportional to their "volume" (product of singular values squared) was developed in (Deshpande et al., 2006; Deshpande and Rademacher, 2010; Guruswami and Sinop, 2012). Its accuracy can be shown to meet the upper bound specified in (4.2).

4.2.3 Heuristic search for the optimal solution

The algorithms reviewed in Sections 4.2.1,4.2.2 are fast, but typically do not produce an optimal solution. A recent study (Arai et al., 2015) has shown how to model the CSSP as a graph search, and then use the classic A* algorithm (Hart et al., 1968; Pearl, 1984) to find the optimal solution. Their algorithm, that we call **CSSP-A***, is guaranteed to find the optimal solution, runs much faster than exhaustive search, but much slower than the other algorithms described in Section 4.2. In particular, it can only be used when k is very small and only on small to moderate size datasets.

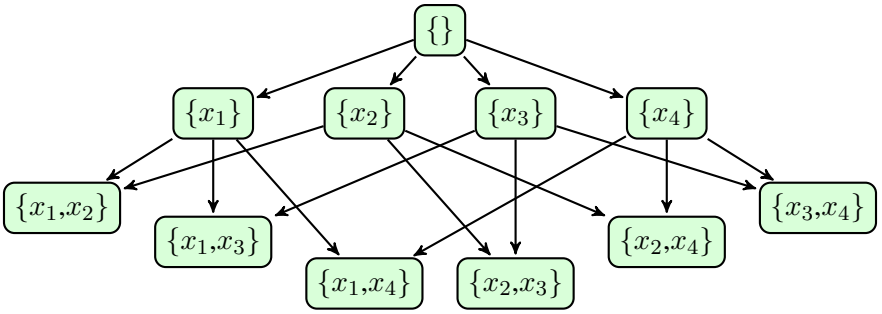


Figure 4.1: Example of subset graph for $n = 4, k = 2$.

0. Put the root node into F.

1. While F is nonempty and no solution found:
- **1.1** Pick n_i with the smallest $f'(n_i)$ from F.
- **1.2** If n_i has k columns return it as the solution.
- **1.3** Otherwise:
  - **1.3.1** Add n_i to C.
  - **1.3.2** Examine all children n_j of n_i.
  - **1.3.2.1** If n_j is in C or F do nothing.
  - **1.3.2.2** Otherwise put n_j in F.

Figure 4.2: Example of the subsets graph and the generic heuristic search algorithm. The algorithm maintains the fringe list F and the closed nodes list C. Several choices of $f'(n_i)$ are discussed in the text.

The CSSP-A* operates on a graph, where nodes correspond to column subsets. There is an edge from subset s_1 to subset s_2 if adding one column to s_1 creates s_2. We call the node s_r the root of the subgraph consisting of all nodes s satisfying $s \supset s_r$. We call the empty set the root of the graph. An example is shown in Figure 4.1.

The algorithm shown in Figure 4.2 is the classic A* algorithm applied to the subsets graph. The description leaves the value of $f'(n_i)$ unspecified.

4.2.4 A* and the Weighted A* algorithms

The standard formulation of the A* algorithm (e.g., (Pearl, 1984; Russell and Norvig, 2010)) involves the three functions $f(n_i), g(n_i), h(n_i)$, that are defined for each node n_i. The value of $g(n_i)$ is the length of the path from the root to n_i, the value of $h(n_i)$ is an estimate of the distance between n_i and the goal, and $f(n_i)$ is calculated as $f(n_i) = g(n_i) + h(n_i)$. Under some restrictions on $h(n_i)$ (it should be "consistent"), the value of $f(n_i)$ is non decreasing along any path. This property can be used to prove the optimality of the A* algorithm.

A standard approach for accelerating the running time of the A* algorithm is the "Weighted A* heuristic". Instead of taking $f'(n_i) = g(n_i) + h(n_i)$, the Weighted A* algorithm increases the weight of the $h(n_i)$ component by using $f'(n_i) = g(n_i) + (1 + \epsilon)h(n_i)$. It can be shown ((Pearl, 1984)) that with this choice of $f'(n_i)$ the A* algorithm gives a solution with error value g_{**} satisfying: $g_{**} \leq (1 + \epsilon)g_*$, where g_* is the error of the optimal solution. Experimental results and theoretical analysis (e.g., (Pearl, 1984; Likhachev et al., 2003)) show that the Weighted A* algorithm runs much faster than the A*. Observe that the bound on g_{**} is multiplicative.

4.3 Weighted A* algorithms for the CSSP

In this section we prove a theorem and use it to derive weighted A* algorithms for the CSSP. The theorem is stated in terms of the functions $f(n_i), g(n_i), h(n_i)$, that are defined for each node n_i. These functions are defined in Section 4.3.1 in terms of eigenvalues of a certain matrix. In the beginning of this section we give a non-computational definition that is sufficient for stating and proving the theorem.

Let n_i be a node in the subsets graph. Let j denote the corresponding number of columns for subset S_i. Define:

$g(n_i) \triangleq$ Error of estimating X using the j columns in S_i.

$f(n_i) \triangleq$ Error of estimating X using the j columns in S_i and additional "best possible" k–j vectors.

$h(n_i) \triangleq g(n_i) - f(n_i)$.

Observe that by definition $g(n_i)$ is monotonically decreasing along any path, and $f(n_i)$ is monotonically increasing along any path. From this it follows that $h(n_i)$ is monotonically decreasing along any path. In addition, at each leaf node q (a goal node of the search) j=k, so that $f(n_q) = g(n_q)$, and $h(n_q) = 0$.

Let n_* be an optimal solution node so that $g(n_*)$ is the smallest possible error in Equation (4.1). Let n_{**} be a solution obtained by the algorithm in Figure 4.1 using a particular choice for $f'(n_i)$ with the (not necessarily optimal) error $g(n_{**})$. It was proved in (Arai et al., 2015) that if $f'(n_i) = f(n_i) = g(n_i) - h(n_i)$ then $g(n_{**}) = g(n_*)$ (i.e, the solution found by the algorithm is optimal). As in Section 4.2.4, this suggests that adding weight to $h(n_i)$ would give a weighted A* algorithm with an improved running time. Suppose instead of $f'(n_i)$=$g(n_i)$–$h(n_i)$ we take:

$$f'(n_i) = g(n_i) - (1 + \epsilon)h(n_i) \tag{4.3}$$

Observe that in this case $f'(n_i)$ is monotonically increasing, and it is possible to give proof similar to the one in (Pearl, 1984) which shows that $g(n_{**}) \leq (1+\epsilon)g(n_*)$. We skip the details of this case because we have found experimentally that this algorithm is impractical. Its running time is even worse than that of the CSSP-A*. Other attempts of defining a monotonically increasing $f'(n_i)$ in terms of $g(n_i), h(n_i)$ also failed to improve the running time. As we describe next, the solution we have found does not have $f'(n_i)$ monotonically increasing or decreasing, but it still gives exact bounds on suboptimality.

Theorem 1: Let n_* be an optimal solution node of the CSSP, and let $g(n_*)$ be the corresponding (smallest) error value. Suppose the function $v(n_i)\geq 0$ can be computed at each node n_i. Define:

$$f'(n_i) = g(n_i) - h(n_i) + \epsilon v(n_i), \quad \epsilon \geq 0$$

Then if $f'(n_i)$ is used by the algorithm in Figure 4.1, it will terminate with a solution node n_{**} satisfying:

$$g(n_{**}) \leq g(n_*) + \epsilon v_{\max}, \quad v_{\max} = \max_{n_i} v(n_i) \tag{4.4}$$

Proof: Suppose the theorem is false. Then there is a case where the algorithm terminates at the solution node n_{**} with the associated error $g(n_{**})$ which satisfies: $g(n_{**}) > g(n_*) + \epsilon v_{\max}$. In Lemma 1 below we prove that under this condition if n_z is any node on the path from the root to n_* then $f'(n_z) < f'(n_{**})$. We proceed to show that this leads to a contradiction.

To see the contradiction observe that the fringe list always contains a node n_z on the path from the root to n_*. (The root is such a node, and if such a node is selected from the fringe, one of its children will be on the path from the root to n_*.) But then since $f'(n_z) < f'(n_{**})$ any such node n_z in the fringe will be selected before n_{**}, so that eventually n_* is selected before n_{**} contradicting the assumption. ■

Lemma 1: With the notation of Theorem 1 under the assumption $g(n_{**}) > g(n_*) + \epsilon v_{\max}$, if n_z is on the path to n_* then $f'(n_z) < f'(n_{**})$.

Proof:

$$\begin{aligned} f'(n_{**}) &= g(n_{**}) – h(n_{**}) + \epsilon v(n_{**}) = g(n_{**}) + \epsilon v(n_{**}) \\ &> g(n_*) + \epsilon v_{\max} + \epsilon v(n_{**}) \\ &\geq f(n_*) + \epsilon v_{\max} \geq f(n_z) + \epsilon v(n_z) \\ &= f'(n_z) \end{aligned}$$

In the above inequality chain we used the following:

1. $h(n_{**})$=0 and $f(n_*) = g(n_*)$, since both n_{**} and n_* are leaf (goal) nodes.

2. $f(n_*) \geq f(n_z)$, since node n_z is on the path to node n_* and $f(n_i)$ is monotonically increasing along any path.

3. $v_{\max} \geq v(n_z)$. ■

Theorem 1 is a useful tool for proving bounds on the suboptimality of the solution. But by itself it doesn't provide guidance as to what constitutes a useful function $v(n_i)$. The following corollary suggests that $v(n_i)$ should be chosen monotonically decreasing along any path. Under this condition the bound becomes tighter during the run of the algorithm.

Table 4.1: Weighted A* algorithms.

Name	f'_i	Suboptimality Bound
CSSP-WA*-g	$g_i - h_i + \epsilon g_i$	$g_{**} \leq g_* + \epsilon g_0$
CSSP-WA*-h	$g_i - h_i + \epsilon h_i$	$g_{**} \leq g_* + \epsilon h_0$
CSSP-WA*-b	$g_i - h_i + \epsilon b_i$	$g_{**} \leq g_* + \epsilon b_0$ $g_{**} \leq (1+\epsilon(k+1))g_*$

Corollary 1: Let n_{**}, n_*, and $v(n_i)$ be as in Theorem 1. Suppose $v(n_i)$ is monotonically decreasing (non-increasing) along any path. Let n_r be a visited node in the path to the solution. Then:

$$g(n_{**}) \leq g(n_*) + \epsilon v(n_r) \tag{4.5}$$

Proof: Apply Theorem 1 to the subgraph rooted at n_r. ■

The guarantee in the corollary may be much stronger then the guarantee provided by the theorem since $v(n_r)$ may be much smaller than the value of $v(n_i)$ at the root.

The suboptimality bounds on $g(n_{**})$ that can be proved from the theorem and the corollary are additive. This is weaker than the multiplicative bound of the classic WA* algorithm (e.g., (Pearl, 1984). Multiplicative bounds are typically preferred because they are more accurate for small $g(n_*)$ values. In particular, when $g(n_*) = 0$ algorithms with multiplicative bounds on their suboptimality return the optimal solution regardless of the value of ϵ.

Since both $g(n_i)$ and $h(n_i)$, as defined in Section 4.3, are monotonically decreasing, we have experimented with them as possible choices for $v(n_i)$. (As discussed in Section 4.3.3 these choices give essentially the same algorithm.) In addition, we propose a third option, the function $b(n_i)$, which gives a multiplicative bound on the algorithm suboptimality.

The algorithms are summarized in Table 4.1. We write f'_i, g_i, h_i, for $f'(n_i), g(n_i), h(n_i)$ respectively, g_0, h_0, b_0 for the values of $g(), h(), b()$ at the root, and g_*, g_{**} for $g(n_*), g(n_{**})$ respectively.

4.3.1 Efficient calculation of heuristic values

In calculating $f(n_i), g(n_i), h(n_i)$ we follow (Arai et al., 2015). Let S_i be the columns at node n_i, and set $j = |S_i|$. Compute Q_i, an orthogonal matrix which forms a basis to S_i. Compute: $X_i = X - Q_i Q_i^T X$, and $B_i = X_i X_i^T$. Let $\lambda_1 \geq \ldots \geq \lambda_m$ be the eigenvalues of B_i. Then:

$$g(n_i) = \sum_{t=1}^{m} \lambda_t = \mathsf{Trace}(B_i) = \|X_i\|_F^2,$$

$$h(n_i) = \sum_{t=1}^{j} \lambda_t, \quad f(n_i) = g(n_i) - h(n_i) = \sum_{t=j+1}^{m} \lambda_t$$

We observe that the heuristics are needed only after line 1.3.2.1 of the algorithm in Figure 4.2. At that point the parent of the node is known. We show how to calculate the children eigenvalues efficiently by performing a more expensive eigenvalue decomposition for the parent, at line 1.3.1 of the algorithm. Furthermore, these more expensive calculations can be reduced by computing an expensive eigenvalue decomposition once, at line 0. To summarize, we discuss three different types of eigenvalue decompositions. The first is calculated once, at the root, the second is calculated for each selected node, and the third is calculated for each child.

Eigenvalue decomposition at the root

Define:

$$B = YY^T$$

The eigenvalue decomposition at the root is performed on B:

$$B = VDV^T$$

Here V has orthonormal columns, and D is diagonal. We note that the values of V and D can also be calculated from the SVD of Y. There are efficient algorithms for computing this factorization. The one we have used is described in (Halko et al., 2011). Let r be the numeric rank of Y then $r \leq \min\{m, n\}$. It is enough to keep V as an $m \times d$ matrix, and retain only d eigenvalues.

Eigenvalue decomposition for parent nodes

This eigenvalue decomposition is performed at line 1.3.1 of algorithm. Instead of working with the matrix B_1, as defined in Theorem 3, we show that there an equivalent matrix that is easier to calculate.

Lemma 1: Define the $r \times r$ matrix H by:

$$H = D - ZZ^T = D - \sum_{j=1}^{r} z_j z_j^T \tag{4.6}$$

where $Z = D^{1/2}V^T Q_1$ and $z_j = D^{1/2}V^T q_j$. Then the matrices B_1 and H have the same eigenvalues (and therefore also the same trace).

Proof: Using the eigenvalue decomposition of B we have:

$$B_1 = (I - Q_1 Q_1^T)B(I - Q_1 Q_1^T) = GDG^T$$

where $G = (I - Q_1 Q_1^T)V$. As we show later H can be written as: $H = D^{1/2}G^T G D^{1/2}$. From this it follows that both H and B_1 can be viewed as the product of the same two matrices (in different order) which implies that they have the same eigenvalues. To complete the proof it remains to show that the two expressions for H are identical. Direct calculation gives:

$$G^T G = V^T(I - Q_1 Q_1^T)V = I - V^T Q_1 Q_1^T V$$

Therefore:

$$\begin{aligned} D^{1/2}G^T G D^{1/2} &= D - D^{1/2}V^T Q_1 Q_1^T V D^{1/2} \\ &= D - ZZ^T \end{aligned}$$

This completes the proof of the lemma. The special structure of the matrix H, which can be expressed as r rank-1 updates to a diagonal matrix, allows for specialized routines for computing its eigenvectors and eigenvalues. See, e.g (Bini and Robol, 2014; Melman, 1998; Golub, 1973).

Computing eigenvalues for children nodes

The calculation of the heuristic at line 1.3.2.2 of the algorithm is the most time consuming part of the algorithm. We show how to compute these values efficiently from the eigenvalue decomposition of the parent.

Theorem 4 Let n_p be a parent node encountered at line 1.3.1 of the algorithm with selection of size k_1. Let Q_p be an orthonormal matrix that spans these k_1 vectors. Let H_p be the matrix computed according to (4.6). Let Trace$_p$ be the trace of H_p. Suppose a child node n_y is created by adding y to the selection of n_p. Then the following procedure computes the heuristic values at n_y without explicitly computing the associated matrix H_y of Equation (4.6).

1. $q_y = Q_p Q_p^T y$.

2. $z_y = D^{1/2} V^T q_y$.

3. $g_y = \text{Trace}(H_y) = \text{Trace}_p - |z_y|^2$.

4. The top $k - k_1 - 1$ eigenvalues of H_y are computed using a specialized method.

5 Compute h_y as the sum of the eigenvalues computed at 4.

6 $f_y = g_y - h_y$.

Proof: Then from (4.6) it follows that:

$$H_y = H_p - z_y z_y^T, \quad z_y = D^{1/2} V^T q_y \tag{4.7}$$

The formula in 3. follows from the linearity of the trace. The computation of the top eigenvalues in 4. can exploit the special structure shown in (4.7). Our particular implementation uses Gragg's method (Melman, 1998). This is an iterative procedure with cubic convergence speed.

4.3.2 A multiplicative bound on suboptimality

We begin by sharpening the right hand size of Equation (4.2): $\text{Err}(S_k^*) \leq (k+1)\text{Err}_k^*$. With the notation used in this section this can be written as: $g(n_*) \leq (k+1)f(n_0)$, where n_0 is the root. Consider a node n_i corresponding to a subset S_i of j columns, where k–j columns still need to be selected. Let S_i^* be the best solution satisfying $S_i^* \supset S_i$. We have:

$$g(n_*) \leq \text{Err}(S_i^*) \leq (k\text{–}j\text{+}1)f(n_i) = (k\text{–}j\text{+}1)(\sum_{t=k-j+1}^{m} \lambda_t)$$

where the eigenvalue calculations are as defined in Section 4.3.1. Now consider adding a zero vector to S_i. This does not change the eigenvalues, but increases the value of j. Therefore, $(k-j)(\sum_{t=k-j}^{m} \lambda_t)$ is also an upper bound. Since this process can be repeated by adding l zero vectors, for $l = 0, \ldots, k-j$, we get the following sharper upper bound:

$$g(n_*) \leq \text{Err}(S_i^*) \leq \min_{l=0,\ldots,k-j}(k\text{–}j\text{+}1\text{–}l)(\sum_{t=k-j+1-l}^{m} \lambda_t)$$

We define the right hand side as $b(n_i)$:

$$b(n_i) = \min_{l=0,\ldots,k-j}(k\text{–}j\text{+}1\text{–}l)(\sum_{t=k-j+1-l}^{m} \lambda_t)$$

As an upper bound $b(n_i)$ is expected to decrease (but not necessarily monotonically). For root node n_0 we have:

$$\begin{aligned} b(n_0) &= \min_{l=0,\ldots,k}(k\text{+}1\text{–}l)(\sum_{t=k+1-l}^{m} \lambda_t) \\ &\leq (k+1)(\sum_{t=k+1}^{m} \lambda_t) \\ &= (k+1)g(n_*) \end{aligned}$$

This shows that the additive upper bound $g_{**} \leq g_* + \epsilon b_0$ implies the multiplicative upper bound $g_{**} \leq (1+\epsilon(k+1))g_*$.

4.3.3 The relationship between the three algorithms

All three algorithms reduce to the CSSP-A* for $\epsilon = 0$. As shown above, the CSSP-WA*-b has a multiplicative bound on its suboptimality while the other two have only additive bounds. This implies that when g_* is very small, the CSSP-WA*-b is more accurate than the other two. Our experiments support this observation.

We proceed to show that the CSSP-WA*-g and the CSSP-WA*-h are essentially the same algorithm corresponding to different choices of ϵ. The f' values of these algorithms can be expressed as:

$$f'_g(n_i) = (1 + \epsilon_g)g(n_i) - h(n_i)$$
$$f'_h(n_i) = g(n_i) - (1 - \epsilon_h)h(n_i)$$

Since an arbitrary scaling of the f' values does not affect the behavior of the algorithm we have the following equivalent form of $f'_g(n_i)$:

$$f''_g(n_i) = \frac{f'_g(n_i)}{1 + \epsilon_g} = g(n_i) - \frac{h(n_i)}{1 + \epsilon_g}$$

Comparing this with the expression for $f'_h(n_i)$ shows that the two algorithms are equivalent under the condition:

$$\epsilon_h = \frac{\epsilon_g}{1 + \epsilon_g}$$

This shows the two algorithms to be equivalent whenever $\epsilon_h < 1$.

4.4 Experimental results

We implemented and tested the three CSSP-WA* algorithms that were described in Section 4.3. The experiments described here compare our algorithms to the current state-of-the-art on several large publicly available datasets.

The state-of-the-art algorithms considered here (see Section 4.2) are the GKS (see (Golub and Van-Loan, 2013)) and the GE ((Gu and Eisenstat, 1996)) from numerical linear algebra, and the

Table 4.2: Dataset description.

Dataset	Size	Availability
Madelon	$2,000 \times 500$	UCI
Isolet5	$1,559 \times 618$	UCI
CNAE-9	$1,080 \times 857$	UCI
TechTC01	$163 \times 29,261$	Technion
Day1	$20,000 \times 3,231,957$	UCI

randomized LEV-GE two-stage-method of Boutsidis et.al (Boutsidis et al., 2009). The datasets information is shown in Table 4.2.

4.4.1 Evaluating accuracy and running time

The results of accuracy evaluation are shown in the Table 4.3. Observe that the WA* algorithms always come as most accurate. On the other hand, the WA* algorithms are significantly slower than the other algorithms as shown in Table 4.4 . The running time is measured in minutes. Therefore WA* algorithms are still being practical.

4.4.2 Dependency on ϵ

The results of experiments for evaluating the performance of the WA* algorithms as a function of ϵ are shown in Figure 4.3. The experiments were performed on the TechTC01 dataset with $k = 50$. The upper left panel shows runtime as a function of ϵ. Although the results for the WA* are clearly worse than the results of the other algorithms, they are still measured in minutes. As a comparison, running the CSSP-A* on the same dataset is impractical, as the results may take hundreds of years. The bottom left panel shows the results only for the WA* algorithms. It clearly shows that the runtime improves with the increase of ϵ value but only up to a certain point. Increasing ϵ above 0.4 gives little or no improvement in running time.

The upper right panel shows accuracy as a function of ϵ. Observe that the WA* algorithms are significantly more accurate than the other algorithms for all values of ϵ in the $0 - 1$ range that we

Table 4.3: Error comparison. The most accurate result for each experiment is underlined. Some results are missing because they are too slow. The WA* algorithms use $\epsilon = 0.5$ on Madelon, CNAE-9 and TechTC01, and $\epsilon = 0.9$ on the other datasets.

k	GKS	GE	LEV_GE	WA*-h	WA*-g	WA*-b
			Dataset Madelon			
20	6.97E+08	7.00E+08	6.52E+08	**6.05E+08**	**6.05E+08**	**6.05E+08**
60	4.82E+08	4.86E+08	5.27E+08	**4.59E+08**	**4.59E+08**	**4.59E+08**
100	3.48E+08	3.47E+08	4.21E+08	**3.38E+08**	**3.38E+08**	**3.38E+08**
			Dataset Isolet5			
20	7.72E+04	9.30E+04	8.01E+04	**6.58E+04**	**6.58E+04**	6.60E+04
60	4.01E+04	4.57E+04	4.10E+04	3.72E+04	3.72E+04	**3.70E+04**
100	2.62E+04	2.88E+04	2.74E+04	2.48E+04	2.48E+04	**2.47E+04**
			Dataset CNAE-9			
20	4.28E+03	4.31E+03	4.97E+03	**4.27E+03**	4.28E+03	4.28E+03
60	2.58E+03	2.56E+03	3.38E+03	**2.55E+03**	**2.55E+03**	**2.55E+03**
100	1.77E+03	1.79E+03	2.43E+03	**1.76E+03**	**1.76E+03**	**1.76E+03**
			Dataset TechTC01			
20	7.14E+05	7.68E+05	8.68E+05	6.93E+05	**6.92E+05**	**6.92E+05**
60	1.52E+05	1.59E+05	1.65E+05	1.29E+05	**1.28E+05**	**1.28E+05**
100	2.78E+04	3.18E+04	2.80E+04	2.12E+04	**2.07E+04**	**2.07E+04**
			Dataset Day1			
20	4.82E+05	-	5.01E+05	4.60E+05	**4.59E+05**	**4.59E+05**
60	3.72E+05	-	3.97E+05	3.50E+05	**3.49E+05**	**3.49E+05**
100	3.20E+05	-	3.48E+05	3.02E+05	**3.01E+05**	-

experimented with. The same comparison showing only the WA* algorithms is shown in the lower right panel. In terms of accuracy the results are less clear, although in general the results seem to improve for smaller ϵ values.

4.5 Discussion

This dissertation describes new algorithms for the column subset selection problems that are based on the classic Weighted A* approach in heuristic search. Our algorithms run slower than the current state-of-the-art, but are still practical even for large datasets. Their advantage is having

Table 4.4: Time comparison, measured in minutes. Some results are missing because they are too slow. The WA* algorithms use $\epsilon = 0.5$ on Madelon, CNAE-9 and TechTC01, and $\epsilon = 0.9$ on the other datasets.

k	GKS	GE	LEV_GE	WA*-h	WA*-g	WA*-b
Dataset Madelon						
20	0.01	0.003	0.30	0.58	0.59	0.58
60	0.01	0.003	0.87	2.11	2.39	2.36
100	0.02	0.006	1.52	4.47	4.92	4.86
Dataset Isolet5						
20	0.01	0.002	0.45	0.97	0.92	0.80
60	0.01	0.008	1.78	3.63	3.50	6.77
100	0.02	0.016	3.94	6.90	6.75	6.78
Dataset CNAE-9						
20	0.01	0.002	0.29	1.82	1.84	1.83
60	0.01	0.002	0.89	6.15	6.27	6.29
100	0.02	0.003	1.55	10.80	10.96	10.97
Dataset TechTC01						
20	0.02	0.006	0.67	0.75	0.79	0.80
60	0.04	0.011	1.93	5.65	5.68	5.68
100	0.06	0.350	3.15	14.50	14.54	14.47
Dataset Day1						
20	0.50	-	10.67	31.25	31.24	31.40
60	1.50	-	35.18	140.68	140.57	140.59
100	2.84	-	63.01	272.58	297.58	-

better accuracy than the current state-of-the-art. Their accuracy and running time can be fine tuned with the ϵ parameter. For $\epsilon = 0$ they all find the optimal solution, but this will most likely take exponential time. Our claim of practicality is for larger ϵ values where our experiments indicate superior accuracy at an "acceptable" running time. We are not aware of any other algorithm in the literature that is practical, and can have similar accuracy levels.

Since the algorithms presented here are still much slower than some of the algebraic/randomized techniques, it is questionable whether they should be an appropriate tool for big data. (See, for ex-

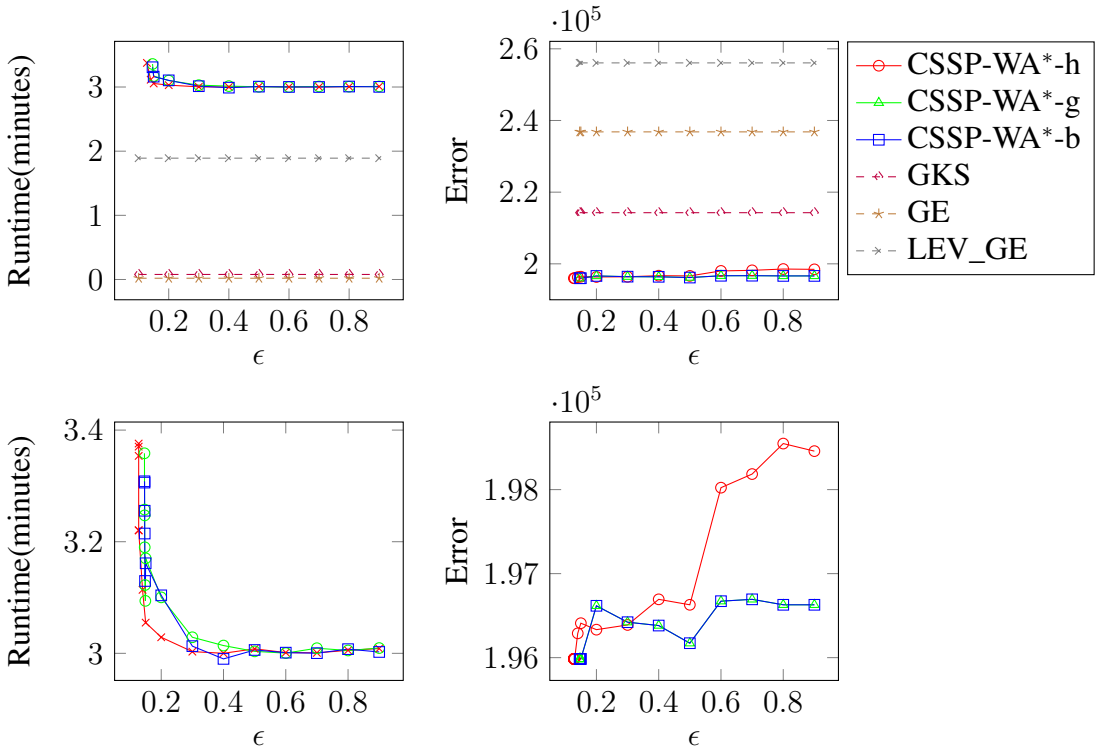


Figure 4.3: Performance comparison as a function of ϵ on TechTC01, for k=50.

ample, the runtime results on the Day1 dataset.) But for most other applications they give practical solutions with superior accuracy to currently available methods.

CHAPTER 5

OPTIMAL COLUMN SUBSET SELECTION AND RELATED PROBLEMS UNDER UNITARILY INVARIANT CRITERIA BY HEURISTIC SEARCH

Chapter 4 describes a solution to the classical column subset selection problem with provable bounds on suboptimality. It computes a suboptimal solution by the weighted A^* heuristic search and the error is measured in Frobenius norm. In this chapter, an optimal solution for a large family of unitarily invariant criteria is discussed. These criteria include, among others, the Spectral norm, the Nuclear norm, and the Schatten p-norms. Related optimal algorithms for expressing a matrix as a linear combination of a column-sparse matrix and a low-rank matrix are also described. The approach is combinatorial, and generalizes previous studies that use the A^* technique with the Frobenius norm.

Since the optimization problem is most likely NP-hard, it is not surprising that our optimal algorithms are only practical for a relatively small selection size. By relaxing the optimality requirement the combinatorial approach produces a solution that is guaranteed to be near the optimal. This sub-optimal algorithm runs much faster, and can be applied to select significantly larger column subsets. Our main contributions are as follows: 1. Optimal subset selection algorithms under unitarily invariant criteria that are much faster than exhaustive search. 2. Optimal algorithms for approximating a matrix by a linear combination of a column-sparse matrix and a low-rank matrix. 3. Sub-optimal algorithms for 1 and 2 that have accuracy guarantees and are shown experimentally to be more accurate than the current state of the art.

Portions of this chapter are the results of collaboration with Swair Shah.

5.1 Introduction

The selection of a small subset of matrix columns that can be used to approximate the entire matrix is known as the Column Subset Selection Problem (**CSSP**). Formally, let X be an $m \times n$

matrix, and let S be an $m \times k$ matrix consisting of k columns selected from X. Then X can be approximated by a linear combination of the columns of S as follows: $X \approx SA$, where A is the $k \times n$ coefficients matrix. A related problem is the unconstrained low rank approximation, where X is approximated by a linear combination of k columns that form the matrix V as follows: $X \approx VA$, where A is the $k \times n$ coefficients matrix. This is the classical **PCA** (Principal Component Analysis). Combining these two low-rank representations gives what we call the "double low-rank representation", or **DLRR**. Here $X \approx SA_1 + VA_2$, where S consists of r_1 columns from X, and V is an unconstrained $m \times r_2$ matrix. Let Θ be an error criterion, then the approximation errors of these representations are:

$$\begin{aligned}
&\text{PCA: } e(X,k) = \min_{V,A} \Theta(X - VA), \text{ with } |V| = k \\
&\text{CSSP: } e(X,k) = \min_{S,A} \Theta(X - SA), \text{ with } S \subset X, |S| = k \\
&\text{DLRR: } e(X,r_1,r_2) = \min_{S,A_1,V,A_2} \Theta(X - SA_1 - VA_2), \text{ with } S \subset X, |S| = r_1, |V| = r_2
\end{aligned} \tag{5.1}$$

In the above equations we use the notation $S \subset X$ to indicate that the columns of S are a subset of the columns of X, and the notation $|S|$ to specify the number of S columns (similarly for V).

The solution to the PCA is well known (e.g., (Jolliffe, 1986)). When Θ is any unitarily invariant norm the best matrix V consists of the k eigenvectors of XX^T corresponding to its largest eigenvalues. In Section 5.2 we point out that this characterization is even more general, and holds for all monotonically increasing unitarily invariant error criteria (these criteria are defined in Section 5.2). Our main result is algorithms for the CSSP and the DLRR that are optimal under the same general monotonically increasing unitarily invariant error criteria.

Our main interest is in the CSSP. Even though the DLRR appears similar to other classic matrix representations which express a matrix as the sum of a sparse matrix and a low-rank matrix (e.g., (Candès et al., 2011)), the representation is not the same. Our interest in the DLRR is as an intermediate step to obtain the CSSP algorithms. We do not explore explicit applications of this representation.

The CSSP has attracted a lot of attention, with the first algorithm (pivoted QR) being developed more than 50 years ago (Businger and Golub, 1965). Over the years algorithms were designed to minimize various error criteria. Numerical linear algebra studies attempt to minimize the Spectral norm (these norms are formally defined in Section 5.2). See, e.g., Chapter 5 in (Golub and Van-Loan, 2013). The most accurate deterministic CSSP algorithm for optimizing the Spectral norm is described in (Gu and Eisenstat, 1996). A randomized algorithm for the Spectral norm is described in (Boutsidis et al., 2009). Randomized algorithms for the Frobenius norm include, among others, (Frieze et al., 2004; Boutsidis et al., 2009; Deshpande and Rademacher, 2010; Guruswami and Sinop, 2012; Paul et al., 2015). Deterministic algorithms for the Frobenius norm include, among others, (Çivril and Magdon-Ismail, 2012; Guruswami and Sinop, 2012; Arai et al., 2015, 2016). Algorithms and analysis related to the Nuclear norm include, among others, (Candès et al., 2011; Cabral et al., 2013). Algorithms and analysis related to the Schatten p-norms include, among others (Nie et al., 2012; Zhu et al., 2015).

Finding the optimal subset is known to be NP-hard for the Volume criterion (Çivril and Magdon-Ismail, 2009). There is also strong evidence that it is NP-hard for the Frobenius norm (Çivril, 2014). The general belief is that CSSP is NP-hard for all "interesting" error criteria, such as all those discussed in Section 5.2. As is the case for many other NP-hard problems, it is still possible to develop algorithms that can do significantly better than exhaustive search. We study such algorithms in this dissertation.

An algorithm that computes the optimal solution to the CSSP in Frobenius norm was described in (Arai et al., 2015). The optimization problem was converted into a graph search problem and solved with the A^* algorithm. Unfortunately, the algorithm in (Arai et al., 2015) works only for the Frobenius norm. The heuristic function "h" that is used by A^* (see, e.g., (Russell and Norvig, 2010)) has no natural interpretation in other error criteria. However, as we show, a similar approach discussed below can still work.

Our approach is to consider the CSSP as a special case of the DLRR. We show in Section 5.3 that the DLRR can be solved by A^* with *no assumptions* on the error criterion. An efficient

implementation requires optimally evaluating a nontrivial function, defined in equation 5.2. We show (in Section 5.4) how to calculate this function efficiently whenever the error criterion is a *monotonically increasing unitarily invariant function*.

5.2 Unitarily invariant monotonic functions

The function Θ in Equation (5.1) associates a matrix with a nonnegative real value. An obvious choice for Θ is a matrix norm. However, some of the criteria that are being used in applications are not norms. An example is the Schatten p-norms with $0 < p < 1$. See, e.g., (Nie et al., 2012; Zhu et al., 2015; Tao, 2012). We proceed to show that the most commonly used error criteria are monotonically increasing unitarily invariant functions.

Unitarily invariant matrix norms do not change their values if the matrix is rotated. It can be shown (e.g., (Marshall et al., 2011)) that a unitarily invariant norm can always be expressed as a monotonically nondecreasing function of its singular values. Thus, we have the following relation:

$$\Theta(A) = \theta(\sigma_1, \ldots, \sigma_m), \text{ where } A \in \mathbb{R}^{m \times n}, \text{ and } \sigma_1 \geq \sigma_2 \ldots \geq \sigma_m \text{ are its singular values}$$

We observe that the opposite is not true. There are monotonically increasing functions θ that do not correspond to norms. Here is a list of the most common error criteria and their description in terms of the function θ:

$$\begin{array}{llll} \text{Spectral:} & \theta = \sigma_1, & \text{Frobenius:} & \theta = (\sum_{j=1}^{m} \sigma_j^2)^{1/2}, \\ \text{Nuclear:} & \theta = \sum_{j=1}^{m} \sigma_j, & \text{Schatten:} & \theta = (\sum_{j=1}^{m} \sigma_j^p)^{1/p} \text{ for } p > 0 \end{array}$$

Clearly, these are all monotonically increasing functions. However, the Schatten p-norms are not matrix norms for $0 < p < 1$. See, e.g., (Tao, 2012). Also observe that both the Frobenius and the Nuclear norms are special cases of the Schatten p-norm.

Definition: A matrix function Θ is called **unitarily invariant monotonic** if:

1. $\Theta(E) = \theta(\sigma_1, \ldots, \sigma_m)$, where σ_i are the singular values of E.

2. θ is nondecreasing in all of its arguments.

We are interested in unitarily invariant monotonic functions because of the following two properties that are given in Theorem 1.

Theorem 1: Suppose $\Theta(E)$ is unitarily invariant monotonic. Let $X \in \mathbb{R}^{m\times n}$,$V \in \mathbb{R}^{m\times k}$, $A \in \mathbb{R}^{k\times n}$ be three matrices. Then:

1. If X and V are known then $A = V^{+}X$ minimizes $\Theta(X - VA)$, where V^{+} is the pseudo-inverse of V.

2. If X is known then the matrix V consisting of the k eigenvectors corresponding to the largest eigenvalues of X minimizes $\Theta(X - VA)$.

Proof: Part 1 appears in the proof of Theorem B.7 in (Marshall et al., 2011). When Θ is a unitarily invariant norm Part 2 is the celebrated Eckart and Young theorem (Marshall et al., 2011). The extension to monotonic functions that are not norms is less well known. A proof can be found in Part b of Theorem 3.2 in (Mathar and Meyer, 1993).

5.3 The A* algorithm

Throughout this section we use the following notation. For two matrices A, B we write $A \subset B$ to indicate that the columns of A are a subset of the columns of B. We write $|A|$ for the number of columns in A, and $A|B$ for the matrix consisting of the columns of A followed by the columns of B.

When the matrix X and the error criterion Θ are known from the context we use the following shortcut. For a matrix Y the function $e(Y)$ is defined as follows:

$$e(Y) = \min_{A} \Theta(X - YA)$$

Observe the following:

$$\min_{A_1,A_2} \Theta(X - S_iA_1 - V_iA_2) = \min_{A} \Theta(X - (S_i|V_i)A) = e(S_i|V_i)$$

5.3.1 Optimality and sub-optimality with accuracy guarantees

As discussed in Section 5.1 we describe algorithms for computing the optimal solutions to the CSSP and the DLRR. An optimal solution gives the smallest approximation error, as defined below. Let Θ be an error criterion (not necessarily unitarily invariant monotonic). The matrices S^*, V^*, A_1^*, A_2^* satisfying $S^* \subset X$, $|S^*| = r_1$, $|V^*| = r_2$ are the optimal solution for the DLRR if:

$$\Theta(X - S^*A_1^* - V^*A_2^*) \leq \Theta(X - SA_1 - VA_2) \quad \text{for any } S \subset X, |S| = r_1, |V| = r_2, A_1, A_2.$$

Sub-optimality with accuracy guarantees is defined as follows. Let S^*, V^*, A_1^*, A_2^* be the optimal solution as defined above. The matrices $S^{**}, V^{**}, A_1^{**}, A_2^{**}$ satisfying $S^{**} \subset X$, $|S^{**}| = r_1$, $|V^{**}| = r_2$ are the sub-optimal solution with the parameters ϵ and μ if:

$$\Theta(X - S^{**}A_1^{**} - V^{**}A_2^{**}) \leq \Theta(X - S^*A_1^* - V^*A_2^*) + \epsilon\mu$$

Observe that a sub-optimal solution with $\epsilon = 0$ is an optimal solution.

5.3.2 The algorithm

It was shown in (Arai et al., 2015, 2016) that the CSSP can be converted into a graph search problem. Here we show that a similar approach can be used with the DLRR. We create the same column subsets graph as in (Arai et al., 2015, 2016), where nodes correspond to column subsets, and there is an edge from subset S_i to subset S_j if adding one column to S_i creates S_j. The graph generated for the matrix $X = (x_1, x_2, x_3)$ is shown on the right side of Figure 5.1.

The A^* algorithm is shown on the left side of Figure 5.1. It keeps a fringe list L of nodes that need to be examined, and a list C of closed nodes, containing nodes that need not be visited again. It also uses a pruning value P, known to be an upper bound on the error of the desired solution.

We proceed to define the function f'. We need the following terminology that, whenever possible, is taken to be identical or similar to (Arai et al., 2016). The DLRR problem is specified in terms of the pair of integers r_1, r_2. The column subset at node n_i is denoted by S_i, and it is of

Input: X,r_1,r_2.
Each node n_i has a subset S_i of size k_i.
Initialization: Put an empty subset into L.

```
while L is nonempty do
    Pick n_i with the smallest f_i' from L.
    if k_i = r_1 then
        stop and return n_i as the solution node.
    else
        Add n_i to C.
        for all children n_j of n_i do
            if n_j is not in C or L then
                if f_j' > P then
                    put n_j in C.
                else
                    put n_j in L.
                end
            end
        end
    end
end
Here L is empty. Solution was not found.
```

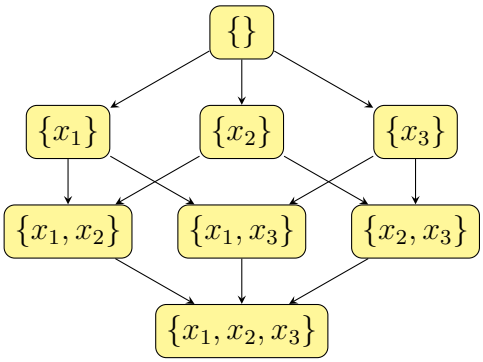


Figure 5.1: The A^* algorithm and an example of the column subsets graph of 3 columns.

size k_i. The function f_i calculates the matrix V_i of size $r_1 + r_2 - k_i$, and the matrices A_1, A_2 of appropriate size. It returns the following value:

$$f_i(r_1, r_2) = \min_{A_1, A_2, |V_i| = r_1 + r_2 - k_i} \Theta(X - S_i A_1 - V_i A_2) = \min_{|V_i| = r_1 + r_2 - k_i} e(S_i | V_i) \tag{5.2}$$

The key idea behind the algorithm is that the desired solution is obtained at the first time the visited node n_i has exactly r_1 columns. We prove this result for the sub-optimality case that has exact optimality as a special case with $\epsilon = 0$.

Theorem 2: Let n_* be an optimal solution node for the DLRR, with the corresponding values S^*, V^*, A_1^*, A_2^*. Suppose the nonnegative function μ_i can be computed at each node n_i. Define:

$$f_i'(r_1, r_2) = f_i(r_1, r_2) + \epsilon \mu_i, \quad \epsilon \geq 0$$

Then the algorithm in Figure 5.1, will terminate with a sub-optimal solution node n_{**} satisfying:

$$\Theta(X - S^{**}A_1^{**} - V^{**}A_2^{**}) \leq \Theta(X - S^{*}A_1^{*} - V^{*}A_2^{*}) + \epsilon\mu_{\max} \tag{5.3}$$

where $\mu_{\max} = \max_i \mu_i$. We begin with two easy claims.

Claim 1: f_i is monotonically increasing along any path.

Proof of Claim 1: Suppose n_j is a child of n_i, so that $S_j = (S_i|x)$, where x is the added column. We need to show the following:

$$f_j(r_1, r_2) \quad = \quad \min_{|V_j|=r_1+r_2-k_i-1} e(S_i|x|V_j) \quad \geq \quad \min_{|V_i|=r_1+r_2-k_i} e(S_i|V_i) \quad = \quad f_i(r_1, r_2)$$

and this follows because the minimum on the right hand side is computed over more instances than the minimum on the left hand side. ■

Claim 2: Suppose the theorem is false. Then for any node n_z on the path from the root to n_* the following condition holds: $f'_z(r_1, r_2) < f'_{**}(r_1, r_2)$.

Proof of Claim 2: Define:

$$g_i(r_1, r_2) = \min_{|V|=r_2} e(S_i|V) \tag{5.4}$$

Clearly, at a solution node i where $k_i = r_1$ we have: $g_i(r_1, r_2) = f_i(r_1, r_2)$. Furthermore, the assumption that Theorem 2 is false can be written as $g_{**}(r_1, r_2) > g_*(r_1, r_2) + \epsilon\mu_{\max}$. The claim can now be proved by the following chain of equalities / inequalities.

$$\begin{aligned}
f'_{**}(r_1, r_2) &= f_{**}(r_1, r_2) + \epsilon\mu_{**} = g_{**}(r_1, r_2) + \epsilon\mu_{**} \\
&> g_*(r_1, r_2) + \epsilon\mu_{\max} + \epsilon\mu_{**} && \text{(from the assumption)} \\
&= f_*(r_1, r_2) + \epsilon\mu_{\max} + \epsilon\mu_{**} \geq f_z(r_1, r_2) + \epsilon\mu_{\max} + \epsilon\mu_{**} && \text{(from Claim 1)} \\
&\geq f_z(r_1, r_2) + \epsilon\mu_z = f'_z(r_1, r_2) \quad \blacksquare
\end{aligned}$$

Proof of Theorem 2: If the theorem is false then from Claim 2 it follows that all nodes on the path from the root to n_* have smaller f' values than $f'_{**}(r_1, r_2)$. Since at any given time at least one of

them is in the fringe list, they should all be selected before n_{**} is selected. But this means that n_* is selected as the solution and not n_{**}.

We point out that the proof does not use any properties of the error criterion Θ. It is also easy to see that this theorem generalizes the results of (Arai et al., 2015, 2016). Specifically, the proof in (Arai et al., 2015) is for the Frobenius case with $r_2 = 0$ and $\epsilon = 0$. The proof in (Arai et al., 2016) is for the Frobenius case with $r_2 = 0$. ■

5.3.3 Pruning

Line 9 of the algorithm implements pruning. The idea is to use an upper bound on the value of the desired solution, and prune all nodes with a larger value. In (Arai et al., 2015) the upper bound was obtained using formulas that hold only for the Frobenius norm. To implement pruning in arbitrary norms we use the following process:

1. Use one of the known (and non-optimal) column subset selection algorithms to obtain a candidate solution.

2. Use the error of the candidate computed in 1. as the value of P in the A^* algorithm.

In our implementation we used the pivoted QR algorithm in Step 1. See Chapter 5 in (Golub and Van-Loan, 2013).

5.4 Efficient implementation with unitarily invariant monotonic functions

Even though the algorithm show in in Figure 5.1 makes no assumptions about the error criteria, it does not provide a practical method for calculating the function f_i as defined in Equation 5.2. In this section we show how to calculate f_i for unitarily invariant monotonic functions. The idea is that the same steps used in (Arai et al., 2015) for the Frobenius norm can still be used in the more general case considered here. Recall that

$$f_i(r_1, r_2) = \min_{A_1, A_2, |V_i| = r_1 + r_2 - k_i} \Theta(X - S_i A_1 - V_i A_2)$$

Table 5.1: Four algorithms solving CSSP and DLRR.

Name	Problem	Parameters
Optimal A^*	CSSP	$r_2 = 0, \epsilon = 0$
Sub-optimal A^*	CSSP	$r_2 = 0, \epsilon > 0$
Optimal A^*	DLRR	$r_2 > 0, \epsilon = 0$
Sub-optimal A^*	DLRR	$r_2 > 0, \epsilon > 0$

Table 5.2: Dataset description.

Dataset	m (examples)	n (features)
libras	360	90
sepctf	267	45
vehicle	846	18

The optimization can be computed in the following sequential way:

1. Compute $A_1 = \arg\min_{A_1} \Theta(X - S_i A_1)$.

2. Compute $X_1 = X - S_i A_1$.

3. Compute $V_i = \arg\min_{|V_i|=r_1+r_2-k_i} \Theta(X_1 - V_i A_2)$

4. Compute $A_2 = \arg\min_{A_2} \Theta(X_1 - V_i A_2)$.

5. Compute $f = \Theta(X_1 - V_i A_2)$.

From Part 1 of Theorem 1 it follows that $A_1 = S_i^+ X$, and $A_2 = V_i^+ X_1$. From Part 2 of Theorem 1 it follows that V_i can be taken as the $r_1 + r_2 - k_i$ top eigenvectors of X_1.

5.5 Experimental Results

We implemented four algorithms to solve the CSSP and the DLRR by choosing different paramenters for the algorithm in Figure 5.1. These parameters are listed in Table 5.1.

Following (Arai et al., 2016) we took μ_i to be the function g_i as defined in Equation (5.4). The experiments were performed on several datasets from the UCI ML Repository (Lichman, 2013). The dataset information is shown in Table 5.2.

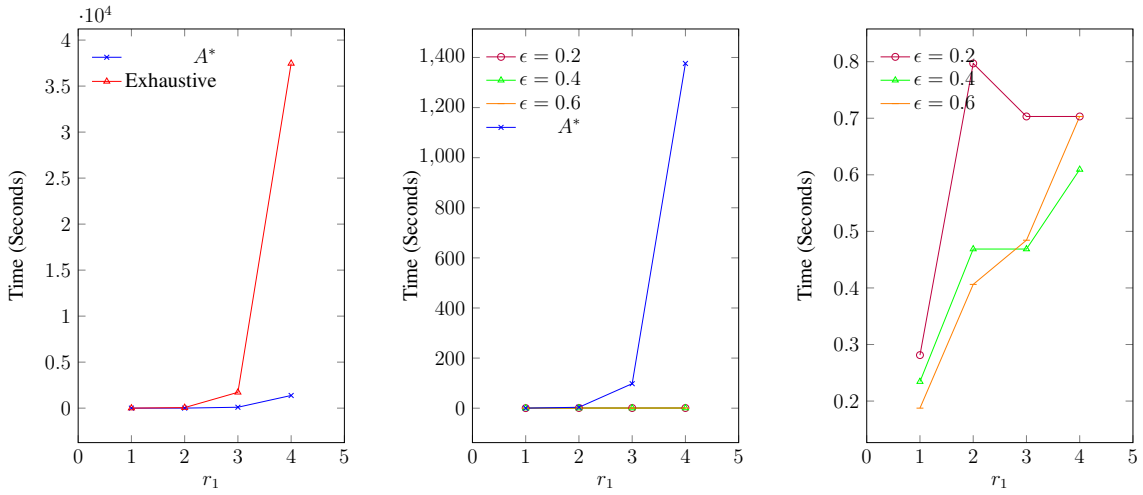


Figure 5.2: Run-time results on the dataset *libras* with optimal A^* to minimize the Schatten p-Norm with $p = 0.25$. Left panel shows improvements over exhaustive search. Further improvements in speed with sub-optimal A^* are shown in the middle and right panels. Notice the scale of the time axis in all three panels.

5.5.1 Run-time comparisons

Figure 5.2 shows the run-time results for the dataset *libras*. In the left panel we observe that Optimal A^* is about four orders of magnitude faster than exhaustive search (while still finding the optimal subset). The middle panel shows that sub-optimal A^* is significantly faster than optimal A^*. Comparison between running time for different values of the parameter ϵ are shown in the right panel. It is not that clear how ϵ affect the run-time with different choices of k.

5.5.2 Accuracy comparisons

Accuracy comparisons are shown between the optimal A^*, the sub-optimal A^*, and a recently published algorithm ARSS (Zhu et al., 2015) that can handle Schatten p-norms. The error criteria used are Schatten p-norm with $p = 0.25$, nuclear norm (which is the Schatten p-norm with $p = 1$), Frobenius norm ($p = 2$) and Spectral norm. Table 5.3 shows this comparison. (The algorithm in (Zhu et al., 2015) solves a convex relaxation of the subset selection problem.) Table 5.4 shows

Table 5.3: Accuracy results for optimal A^*, sub-optimal A^* and ARSS. The best results are highlighted.

Error Criterion	r_1	A^*	Sub-optimal $\epsilon = 0.2$	Sub-optimal $\epsilon = 0.4$	Sub-optimal $\epsilon = 0.8$	ARSS
Dataset vehicle						
Schatten (0.5)	5	**1.252E+02**	**1.252E+02**	**1.252E+02**	**1.252E+02**	1.731E+02
Nuclear	5	**1.399E+03**	1.403E+03	1.569E+03	1.569E+03	3.466E+03
Frobenius	5	**2.229E+05**	2.983E+05	2.983E+05	2.983E+05	4.296E+06
Spectral	5	**247.58**	326.12	326.12	326.12	-
Schatten (0.5)	10	**5.799E+01**	**5.799E+01**	**5.799E+01**	**5.799E+01**	1.086E+02
Nuclear	10	**4.668E+02**	5.202E+02	5.202E+02	5.202E+02	1.682E+03
Frobenius	10	**3.603E+04**	4.731E+04	4.731E+04	5.064E+04	4.138E+05
Spectral	10	**112.19**	138.8	144.99	144.99	-
Dataset spectf						
Schatten (0.5)	4	**3.857E+02**	**3.857E+02**	3.862E+02	**3.857E+02**	4.084E+02
Nuclear	4	**4.029E+03**	**4.029E+03**	**4.029E+03**	4.033E+03	4.735E+03
Frobenius	4	**5.759E+05**	**5.759E+05**	**5.759E+05**	**5.759E+05**	9.450E+05
Spectral	4	**278.300**	300.943	312.813	312.813	-
Schatten (0.5)	5	**3.719E+02**	3.720E+02	3.724E+02	3.722E+02	3.976E+02
Nuclear	5	**3.814E+03**	**3.814E+03**	3.817E+03	3.817E+03	4.598E+03
Frobenius	5	**5.118E+05**	5.131E+05	5.147E+05	5.169E+05	6.551E+05
Spectral	5	**252.658**	257.912	290.597	290.597	-
Dataset libras						
Schatten (0.5)	3	**5.391E+01**	5.392E+01	5.392E+01	5.393E+01	5.452E+01
Nuclear	3	**8.116E+01**	8.122E+01	8.122E+01	8.105E+01	9.179E+01
Frobenius	3	**5.601E+02**	**5.601E+02**	6.299E+02	6.323E+02	1.523E+03
Spectral	3	**13.516**	13.621	13.716	13.866	-
Schatten (0.5)	4	**5.074E+01**	**5.074E+01**	5.075E+01	5.076E+01	5.452E+01
Nuclear	4	**6.844E+01**	6.853E+01	6.853E+01	6.848E+01	9.179E+01
Frobenius	4	**3.576E+02**	3.627E+02	3.963E+02	3.963E+02	1.523E+03
Spectral	4	**8.558**	9.954	9.954	9.954	-

comparison to the Gu and Eisenstat algorithm (Gu and Eisenstat, 1996). Clearly that the optimal A^* algorithm performs better than Gu and Eisenstat algorithm in terms of accuracy.

5.5.3 Experiments with double low-rank representations (DLRR)

Table 5.5 shows the result of using the optimal A^* algorithm for computing double low-rank representations. We compare the error obtained by the optimal algorithm to the error obtained by first

Table 5.4: Error comparison with Gu Eisenstat algorithm for spectral norm.

r_1	A^*	Gu Eisenstat
Dataset vehicle		
5	351.43	371.17
6	247.58	248.58
7	206.59	228.03
Dataset spectf		
3	317.52	382.58
4	278.30	510.18
5	252.65	348.23
Dataset libras		
3	13.51	18.66
4	8.55	11.86

computing the optimal CSSP and then approximating the remainder error with PCA. The error is measured in the appropriate norm of the residual matrix. The results clearly show that the optimal algorithm is better.

5.6 Discussion

Column subset selection is an important problem that has attracted a lot of attention. While there are many algorithms that address speed and accuracy, few come with a guarantee on optimality. We wish to point out that some previous studies have a different interpretation of optimality than the one discussed here. Consider the following two error measures:

$$e_1(X,k) = \min_{V,A} \Theta(X - VA), \text{ with } |V| = k$$

$$e_2(X,k) = \min_{S,A} \Theta(X - SA), \text{ with } S \subset X, |S| = k$$

Here e_1 the the PCA error, and e_2 is the CSSP error. Clearly $e_1(X,k) \leq e_2(X,k)$, and it may be interesting to determine how bad is e_2 when compared to e_1 in the worst case. Many studies addressed this problem by searching for a function $p(k,m,n)$ independent of X such that:

$$e_1(X,k) \leq e_2(X,k) \leq P(k,m,n)e_1(X,k) \quad \text{for all } X.$$

Table 5.5: Error comparision between DLRR and CSSP followed by PCA.

Error Criterion	r_1	r_2	Optimal DLRR	Optimal CSSP followed by PCA
Dataset vehicle				
Schatten ($p = 0.5$)	4	6	**5.490E+01**	5.649E+01
Nuclear	4	6	**4.187E+02**	4.438E+02
Frobenius	4	6	**2.942E+04**	3.170E+04
Spectral	4	6	**1.004E+02**	1.028E+02
Schatten ($p = 0.5$)	6	4	**5.538E+01**	5.672E+01
Nuclear	6	4	**4.266E+02**	4.531E+02
Frobenius	6	4	**3.057E+04**	3.415E+04
Spectral	6	4	**1.019E+02**	1.095E+02
Schatten ($p = 0.5$)	6	6	**3.547E+01**	3.646E+01
Nuclear	6	6	**2.292E+02**	2.476E+02
Frobenius	6	6	**1.131E+04**	1.294E+04
Spectral	6	6	**7.085E+01**	7.317E+01
Dataset spectf				
Schatten ($p = 0.5$)	2	3	**3.561E+02**	3.604E+02
Nuclear	2	3	**3.443E+03**	3.547E+03
Frobenius	2	3	**3.920E+05**	4.262E+05
Spectral	2	3	**2.035E+02**	2.348E+02
Schatten ($p = 0.5$)	3	2	**3.599E+02**	3.631E+02
Nuclear	3	2	**3.523E+03**	3.670E+03
Frobenius	3	2	**4.138E+05**	4.680E+05
Spectral	3	2	**2.070E+02**	2.717E+02
Schatten ($p = 0.5$)	3	3	**3.440E+02**	3.466E+02
Nuclear	3	3	**3.280E+03**	3.399E+03
Frobenius	3	3	**3.593E+05**	3.948E+05
Spectral	3	3	**1.918E+02**	2.338E+02

See, e.g., (Boutsidis et al., 2009; Deshpande and Rademacher, 2010; Guruswami and Sinop, 2012). Among the results that were obtained in this line of study it was shown that for the Frobenius norm the smallest possible is $P(k, m, n) = \sqrt{k+1}$. This suggests that an algorithm that can guarantee error $= \sqrt{k+1}\, e_1(X, k)$ is optimal. This is very different from our result since for most matrices X the value of $e_2(X, k)$ can be much smaller than $\sqrt{k+1}\, e_1(X, k)$.

The algorithms developed in this chapter compute the optimal column subset selection much faster than exhautive search. There are currently no alternatives to these algorithms when one is interested in obtaining the best possible selection in anything but the Frobenius norm. Current state-of-the-art algorithms cannot provide a guarantee on the optimality of a solution even in situations where such a solution is found.

CHAPTER 6

TWO-STAGE FEATURE SELECTION WITH UNSUPERVISED SECOND STAGE

A common technique for reducing the run time of feature selection is to perform it in two stages. In the first stage a fast and simple filter is applied to select good candidates. These candidates are further reduced in the second stage by an accurate algorithm that may run significantly slower.

It is common to distinguish between supervised and unsupervised feature selection. In the supervised case features are selected for predicting a set of labels. Such label information is not used in the unsupervised case.

We describe a general framework that can use an arbitrary off-the-shelf unsupervised algorithm for the second stage. The algorithm is applied to the selection obtained in the first stage weighted appropriately. Our main technical result is the computation of these weights. We show that the weights can be found as the solution to a small quadratic optimization problem. The solution is deterministic, and improves on previously published studies that use probabilistic ideas to compute the weights. To the best of our knowledge our approach is the first technique for converting a supervised feature selection problem into an unsupervised problem.

Complexity analysis shows that the proposed technique is very fast, can be implemented in a single pass over the data, and can take advantage of the data sparsity. Experimental results show its accuracy to be comparable to that of much slower techniques.

Portions of this chapter are the results of collaboration with Hiromasa Arai and Crystal Maung. The results are published in International Conference on Tools with Artificial Intelligence (ICTAI), 2017.

6.1 Introduction

The selection of a small number of useful features for applications such as machine learning and data mining is computationally challenging even for moderate size data (Guyon and Elisseeff,

Table 6.1: Cases of two-stage algorithms.

Case	Original Problem	Stage 1	Stage 2	Comment
1	s	s	s	Typical case
2	s	s	u	**This dissertation**
3	s	u	s	Others
4	s	u	u	**This dissertation**
5	u	s	s	X
6	u	s	u	X
7	u	u	s	X
8	u	u	u	**This dissertation** and others

2003). The problem becomes much harder with the increase in data size. A common approach for reducing the run time is to perform the selection in two stages. An initial selection is computed in the first stage by a fast and simple feature selection technique, followed by a second stage that uses a more accurate and time consuming method.

To properly describe the relationship between the original feature selection problem and algorithms used in the first and second stage we need the distinction between supervised and unsupervised feature selection. The input to the supervised case is labeled features, and for the unsupervised case it is unlabeled features. In the supervised case the goal is to select features that can be used to approximate the labels. In the unsupervised case there are no labels, and features are typically selected for estimating the data itself.

Since each algorithm can be either supervised or unsupervised there are 8 cases relating the original selection problem to the algorithms in the two stages. They are shown in Table 6.1. The entry "s" indicates supervised, and "u" indicates unsupervised. We are not aware of studies corresponding to cases 5,6,7.

Case 1 is the most common. The problem is supervised, and it is typically addressed by filters in the first stage followed by embedded methods in the second stage. See, e.g., (Guyon and Elisseeff, 2003; Guyon et al., 2004; Tang et al., 2007; Javed et al., 2012; Liao et al., 2014; Fan and Lv, 2008).

Case 3 uses an unsupervised algorithm in the first stage instead of the supervised algorithm of Case 1. This may have speed advantage, since some of the unsupervised algorithms are faster than the supervised filters. See, e.g., (Drineas et al., 2010; Dasgupta et al., 2007; Paul and Drineas, 2016; Muthukrishnan, 2011).

Case 8 solves the unsupervised case by concatenating two unsupervised algorithms. To the best of our knowledge an algorithm of this type was first proposed in (Boutsidis et al., 2009).

The results presented in this dissertation improve previous results in Case 8. They are also applied successfully in cases 2 and 4. We are not aware of any previous work on these cases, where the original supervised problem is solved with an unsupervised second stage algorithm.

The main idea

We begin with the idea of (Boutsidis et al., 2009) that strongly influenced our approach. Consider their solution to the unsupervised problem of selecting k out of the n columns from the matrix X.

a. Compute a probability p_i for each column x_i.

b. Select $k_1 \geq k$ columns according to this distribution.

c. If x_j is selected, compute its weight as $w_j = 1/\sqrt{p_j}$.

d. Create the matrix X_1 from the k_1 columns $w_j x_j$.

e. Run unsupervised method to select k columns from X_1.

The key idea of Boutsidis et. al. is that with a smart choice of the p_i the matrix X_1 approximates X "in some sense". From this it follows that a good selection from X_1 should also be a good selection from X.

The main idea behind our result is the following: Regardless of how the columns are selected in Step b, it is always possible to compute the "best" weights so that X_1 approximates X. We show that finding the best weights involves quadratic optimization. Furthermore, in this setting the columns of X_1 do not have to be columns of X. Thus, it is possible to consider a generalized setting where k columns of X are chosen to approximate a different matrix Y, consisting of label information.

As we show, this generalized setting includes unsupervised as well as supervised feature selection. The supervised case includes both the standard setting where there is a single label associated with each data item as well as the more general case where many labels are associated with each data item.

Chapter organization

The chapter is organized as follows. Notation and the algorithm are introduced in Section 6.2. The computation of the weights is discussed in Section 6.3. An upper bound on the algorithm accuracy in terms of the errors that are explicitly minimized is shown in Section 6.4. Runtime analysis is performed in Section 6.5. Experimental results are discussed in Section 6.6.

6.2 Problem statement and notation

There are two main approaches to unsupervised feature selection. The first approach considers a feature to be redundant if it is similar to another feature that has already been selected (e.g., (Li et al., 2012)). The second approach considers a feature to be redundant if it can be approximated by a linear combination of other features that have already been selected (e.g., (Golub and Van-Loan, 2013; Jimenez-Rodriguez et al., 2007)). The second approach is sometimes called the *Column Subset Selection Problem (CSSP)*. We follow the second approach as it fits into the following general setting:

Let X be a "dictionary" matrix of m rows and n columns. Let Y be the labels matrix of m rows and N columns. Let S_1 be matrix of size $m \times k_1$ formed from the columns selected from X in the first stage. Let S_2 be matrix of size $m \times k_2$ formed from the columns selected from S_1 in the second stage. We wish to minimize the error of approximating the matrix Y in terms of S_2:

$$e = \min_{A} |Y - S_2 A|_F^2 \tag{6.1}$$

Observe that unsupervised feature selection (CSSP) is the special case where Y=X, and the standard supervised feature selection corresponds to the case where Y is a single column (N=1). With

Input: X, Y, k_2.

Design parameters: k_1, Alg1, Alg2.

Output: S_2.

1. Use Alg1 with input X, Y, k_1 to compute S_1.
(if Alg1 is unsupervised, it ignores Y.)

2. Use S_1, Y to compute the weights $w_1, \ldots, w_{k_1}$.
Create the $m \times k_1$ matrix X_1 as follows:
Column j of X_1 is column j of S_1 scaled by w_j.

3. Use Alg2 with input X_1, k_2 to compute S_2.

Figure 6.1: The two-stage algorithm.

$N>1$ we have the multi-label case that has drawn a lot of interest recently (e.g., (Tsoumakas et al., 2011)). Other studies that use and analyze the case where columns of X are selected to approximate an entire matrix Y include (Boutsidis et al., 2013; Tropp et al., 2006; Chen and Huo, 2006; Cotter et al., 2005).

In describing the algorithm we assume the availability of two algorithms. Alg1 is used in the first stage. It can be either supervised or unsupervised. Its input is the matrices X, Y and the parameter k_1, and its output is the selected columns S_1. (If Alg1 is unsupervised, it does not use the matrix Y.) Alg2 is the unsupervised algorithm of the second stage. It takes as input the matrix X_1 and the parameter k_2, and returns the selected columns S_2. The two-stage algorithm is shown in Figure 6.1. **The problem considered here is how to calculate the weights in Step 2.** Our solution is described in Section 6.3.

6.3 Computing the weights

This section describes our main result: an algorithm for computing the weights in Step 2 of the two-stage algorithm. We begin by observing that the approximation error of (6.1) depends only on YY^T and not directly on Y. This implies that the weights should be chosen so that the matrix X_1 in the algorithm satisfies: $X_1 X_1^T \approx YY^T$. Measuring the quality of the approximation in the

Frobenius norm we show that the optimal weights can be computed by quadratic optimization. In Section 6.4 we prove that minimizing the approximation error in Steps 2 and 3 of the two-stage algorithm indeed minimizes the desired error e of Equation (6.1).

Proposition-1. Let Y, S be two matrices. Suppose $e = \min_A \|Y - SA\|_F^2$. Then for any matrix Q with orthonormal columns that forms a basis to the column space of S:

$$e = \text{Trace}\{(I - QQ^T)YY^T(I - QQ^T)\}$$

Proof: Since S and Q have the same column space we have: $e = \min_A \|Y - QA\|_F^2$. In this expression the minimizing A is given by: $A = Q^TY$, so that:

$$e = \|Y - QQ^TY\|_F^2 = \|(I - QQ^T)Y\|_F^2$$

The proposition now follows from the relation between the Frobenius norm and the trace. (For any matrix M, the following relation holds: $\|M\|_F^2 = \text{Trace}\{MM^T\}$.) ■

From Proposition-1 it follows that the error e depends on YY^T and not directly on Y. Therefore, any matrix X_1 (not necessarily created from Y columns) which satisfies $X_1X_1^T = YY^T$ can be used instead of Y in Equation (6.1). This suggests that X_1 should be chosen satisfying $X_1X_1^T \approx YY^T$. (These arguments are quantified in Theorem-2.) Specifically, we propose to determine the weights needed in Step 2 of the two-stage algorithm by minimizing $\|X_1X_1^T - YY^T\|_F$. The minimization procedure can be done efficiently as explained in Theorem-1.

Theorem-1. Let $S_1 = \left(s_1, \ldots, s_{k_1}\right)$ be an arbitrary $m \times k_1$ matrix. Let $w = \left(w_1, \ldots, w_{k_1}\right)^T$ be a weights vector. Create the $m \times k_1$ matrix X_1 as follows: column j of X_1 is $w_j s_j$. Define the $k_1 \times k_1$ matrix H and the k_1 vector h by:

$$H = (H_{ij}), \;\; H_{ij} = (s_i^T s_j)^2, \quad h = (h_j), \;\; h_j = |Y^T s_j|^2$$

Let z be a k_1 vector. Consider the following two optimization problems:

$$w_{\text{opt}} = \arg\min_{w} \|X_1 X_1^T - YY^T\|_F^2$$

$$z_{\text{opt}} = \arg\min_{z} z^T H z - 2h^T z,$$

$$\text{subject to } \; z_j \geq 0, \; j = 1, \ldots, k_1$$

Then H is positive semidefinite and the coordinates of the two solutions are related by:

$$w_{\text{opt}}(j) = \sqrt{z_{\text{opt}}(j)}, \quad j = 1, \ldots, k_1$$

Proof: Define: $z_j = w_j^2$ and $B_j = s_j s_j^T$ for $j = 1, \ldots, k_1$. With this notation we have $X_1 X_1^T = \sum_{j=1}^{k_1} z_j B_j$. Plugging this into the first optimization problem gives:

$$\begin{aligned} e_1 &= \|\sum_{j=1}^{k_1} z_j B_j - YY^T\|_F^2 \\ &= \text{Trace}\{(YY^T)^2\} + \text{Trace}\{(\sum_{j=1}^{k_1} z_j B_j)^2\} - 2\text{Trace}\{\sum_{j=1}^{k_1} z_j B_j YY^T\} \end{aligned}$$

The first trace expression is independent of z. Manipulating the second trace expression gives:

$$\begin{aligned} \text{Trace}\{(\sum_j z_j B_j)^2\} &= \sum_{ij} z_i z_j \text{Trace}\{B_i B_j\} \\ &= \sum_{ij} z_i z_j (s_i^T s_j)^2 \\ &= z^T H z \end{aligned}$$

where we have used the easily provable identity: $\text{Trace}\{B_i B_j\} = (s_i^T s_j)^2$. Manipulating the third trace expression gives:

$$\begin{aligned} \text{Trace}\{\sum_j z_j B_j YY^T\} &= \sum_j z_j \text{Trace}\{Y^T s_j s_j^T Y\} \\ &= \sum_j z_j \text{Trace}\{s_j^T YY^T s_j\} \\ &= \sum_j z_j |Y^T s_j|^2 \\ &= h^T z \end{aligned}$$

This shows that

$$\min_{w} \|X_1 X_1^T - YY^T\|_F^2 = \text{Trace}\{(YY^T)^2\} + \min_{z} z^T H z - 2h^T z$$

which proves the second part of the theorem. It remains to show that H is positive semidefinite. Observe that the following matrix is positive semidefinite: $\sum_j z_j B_j$. This is also true for its square. Therefore, from the second trace expression $z^T H z$ is the trace of a positive semidefinite matrix, and must be nonnegative. ■

From the theorem it follows that the weights can be calculated as a solution to a constrained quadratic optimization problem. We have experimented with an implementation of (Goldfarb and Idnani, 1983), as well as with a simple heuristic that solves the unconstrained problem and zeroes out the negative coordinates. In our experiments there was little difference in the quality of the solutions obtained by these two methods. Observe that the unconstrained problem has a closed form solution. The minimizer z_{opt} can be found as the solution to system of linear equations $Hz = h$ (Golub and Van-Loan, 2013). This system of linear equations can be solved reliably and efficiently.

Since the matrix H may be singular the optimizing solution may not be unique. It is standard practice to add a regularization term which makes H positive definite, guarantees a unique solution, and increases the stability of the optimization.

The formulas in Theorem-1 give the algorithm shown in Figure 6.2, where a regularization term controlled by $\gamma > 0$ is added for stability. Observe that the computation of h is rearranged so that it is calculated in one pass over Y.

6.4 Error bounds

The algorithm error (6.1) is affected by the error of approximating YY^T and by the error of the algorithm in the Stage-2. In this section we investigate how these errors are related.

Input: $S_1 = (s_1, \ldots, s_{k_1})$, Y.

Design parameters: $\gamma > 0$ (the regularization parameter).

Output: the weights vector w.

0. Initialize the vector h to 0.

1. In one pass over Y, for $j = 1, \ldots, N$, read the vector y_j for $i = 1, \ldots, k_1$ increment h_i by $(s_i^T y_j)^2$.

2. for $i, j = 1, \ldots, k_1$ set $H_{ij} = (s_i^T s_j)^2$.

3. for $i = 1, \ldots, k_1$ increment H_{ii} by γ.

4. Solve the linear system $Hz = h$ for the unknown z.

5. for $j = 1, \ldots, k_1$ if $z_j < 0$ set $z_j = 0$.

6. for $j = 1, \ldots, k_1$ $w_j = \sqrt{z_j}$.

Figure 6.2: Algorithm for calculating the weights.

Theorem-2. Let ϵ_1 be the error minimized by the quadratic optimization step for computing the weights. Let ϵ_2 be the error minimized by the unsupervised algorithm in the second stage. Let ϵ be the error of the two-stage algorithm. Let m be the number of rows of X. Then:

$$\epsilon \leq \sqrt{m\epsilon_1} + \epsilon_2$$

Proof: Using the notation of Section 6.3 we have: $\epsilon_1 = \|E_1\|_F^2$, where $E_1 = X_1 X_1^T - YY^T$. Let the matrix Q_2 be an orthogonal basis to the column space of S_1. From Proposition-1 we have: $\epsilon_2 = \text{Trace}\{(I - Q_2 Q_2^T) X_1 X_1^T (I - Q_2 Q_2^T)\}$. Similarly, ϵ can be written as:

$$\begin{aligned}
\epsilon &= \text{Trace}\{(I - Q_2 Q_2^T) YY^T (I - Q_2 Q_2^T)\} \\
&= \text{Trace}\{(I - Q_2 Q_2^T)(X_1 X_1^T - E_1)(I - Q_2 Q_2^T)\} \\
&= \epsilon_2 - \text{Trace}\{(I - Q_2 Q_2^T) E_1 (I - Q_2 Q_2^T)\} \\
&= \epsilon_2 - \text{Trace}\{Q_3^T E_1 Q_3\}
\end{aligned}$$

where Q_3 is the orthogonal complement of Q_2. Let $\lambda_1, \ldots, \lambda_m$ be the eigenvalues of E_1. It is known that $\text{Trace}\{Q_3^T E_1 Q_3\} \geq -\sum_{j=1}^m |\lambda_j|$. (See, e.g., (Coope and Renaud, 2009)). Therefore,

m	-	number of rows of both X and Y.
n	-	number of X columns
N	-	number of Y columns
k_1	-	number of columns selected in Stage-1
k_2	-	number of columns selected in Stage-2
Z_S	-	number of nonzero values in Stage-1 selection.
Z_X	-	number of nonzero values in X.

Figure 6.3: The relevant complexity parameters.

to prove the theorem it remains to show that $\sum_{j=1}^{m} |\lambda_j| \leq \sqrt{m\epsilon_1}$. Since $\epsilon_1 = \|E_1\|_F^2 = \sum_{j=1}^{m} \lambda_j^2$ it is enough to establish the inequality: $\sum_{j=1}^{m} |\lambda_j| \leq \sqrt{m \sum_j \lambda_j^2}$. This inequality follows from one of the generalized means inequalities (Hardy et al., 1952) which states that:

$$\frac{\sum_{j=1}^{m} |\lambda_j|}{m} \leq \sqrt{\frac{\sum_{j=1}^{m} \lambda_j^2}{m}}$$

■

6.5 Runtime analysis

In this section we analyze the running time of the two-stage algorithm. The relevant parameters are shown in Figure 6.3. We consider both the dense and the sparse case. In the sparse case the "dictionary" matrix X sparse, containing only Z_X nonzero values ($Z_X \ll mn$). In this case the selection S_1 in the first stage would have Z_S nonzero values ($Z_S \ll mk_1$).

6.5.1 Stage-1 feature selection

In typical large datasets the most important complexity parameter is n. The first stage reduces n to k_1. Several options for fast algorithms that we experimented with and their complexity, are listed in Table 6.2.

The "Random" algorithm performs the selection uniformly at random. In typical storage models that allow skipping (e.g. hard drives which allow seeks) there is no need to read the entire data.

Table 6.2: Some Stage-1 algorithms. The "s/u" label indicates whether or not the algorithm is supervised.

Algorithm	Dense	Sparse	passes
Random-u	$O(k_1)$	$O(k_1)$	1
FKV-u	$O(mn)$	$O(Z_X)$	1
Leverage-u	$O(k_1mn)$	$O(k_1Z_X)$	1
Pearson-s	$O(mn)$	$O(Z_X)$	1
Filter-s	$O(Nmn)$	$O(NZ_X)$	1
QRP-u	$O(k_1mn)$	$O(k_1mn)$	k_1

This is, by far, the fastest approach. The FKV algorithm (Frieze et al., 2004) is unsupervised. It selects candidates with probability proportional to their squared norm. The Leverage Score algorithm is unsupervised. It uses probabilities computed from a truncated SVD of X (Boutsidis et al., 2009). The complexity we specify assumes the availability of fast truncated SVD algorithms (e.g., (Halko et al., 2011)). Pearson is a classic filter technique (e.g., (Guyon and Elisseeff, 2003)). In its standard form it can only be applied to one label. More general filters, similar to Pearson, can be constructed to handle the multiple label case. Their dominant computational step is the product of the matrix Y with each one of X columns. They are efficient for a small N (number of labels), but otherwise become quite expensive. The one we implemented is a single iteration of the Orthogonal Matching Pursuit algorithm (Mallat, 1999). The last algorithm, the QRP, performs the selection using the pivoted QR factorization (Golub and Van-Loan, 2013). Unlike the other algorithms it removes redundancies in the initial selection, but requires more than a single pass.

6.5.2 Stage-2 feature selection

The unsupervised algorithm in Stage-2 performs selection of k_2 columns from the matrix S_1 of size $m \times k_1$. Here the most relevant parameters are k_1 and m. We assume that k_1 is small enough so that S_1 can fit in memory and the number of passes is no longer relevant.

Some of the algorithms and their complexity are listed in Table 6.3. We also list some classic supervised algorithms that can be used in Stage 2 without the calculation of weights. (They are the "competitors".)

Table 6.3: Some Stage-2 algorithms. The "s/u" label indicates whether or not the algorithm is supervised.

Algorithm	s/u	Dense	Sparse
GKS	u	$O(k_1 k_2 m)$	$O(k_2 Z_S)$
GE	u	$O(k_1 m^2)$	$O(k_1 m^2)$
FS	s	$O(k_1 k_2 m N)$	$O(k_1 k_2 m N)$
SVM-RFE	s	$O(m^3 k_1^2)$	$O(m^3 k_1^2)$

The GKS algorithm (Golub and Van-Loan, 2013) is unsupervised. It performs pivoted column selection in the leveraged space. The GE algorithm is unsupervised. It was established in (Gu and Eisenstat, 1996), and has the best guarantee on accuracy when measured in the spectral norm.

The FS (Forward Selection) algorithm is supervised. Even though it is greedy, it appears to perform very well in practice. There are recent theoretical and experimental results that explain the good performance and show that under some conditions it produces a selection very close to the optimal. See (Das and Kempe, 2011), and extended version in (Das and Kempe, 2011).

The SVM-RFE was established in (Guyon et al., 2002). The expensive step is the solution of a quadratic programming problem, that takes in the worst case $O(m^3 k_1)$ (see (Chapelle, 2007)). This has to be repeated k_1 times.

6.5.3 Computing the weights

Calculating the weights involves calculating the matrix H, the vector h, and solving a quadratic optimization problem. Calculating H takes $k_1(k_1+1)/2 = O(k_1^2)$ dot products of m vectors. Calculating h takes Nk_1 dot products of m vectors. Solving the constrained quadratic optimization takes $O(k_1^4)$. The unconstrained case takes $O(k_1^3)$. This shows that solving the exact quadratic optimization dominates the complexity when $k_1^2 > m$, which is very likely. As discussed in Section 6.3 we use the unconstrained solution which gives the following running times:

$$\text{Dense running time:} \quad O(k_1 m(N + k_1))$$

$$\text{Sparse running time:} \quad O((k_1 + N)Z_S + k_1^3)$$

Table 6.4: Dataset description.

Dataset	Size	Labels	Availability
CNAE-9	$1{,}080 \times 857$	-	UCI
TechTC01	$163 \times 29{,}261$	-	Technion
Day1	$20{,}000 \times 3{,}231{,}957$	-	UCI
Isolet5	$1{,}559 \times 618$	1	UCI
Gisette	$6{,}000 \times 5{,}000$	1	UCI
Arcene	$100 \times 10{,}000$	1	UCI
Enron	$1{,}702 \times 1{,}001$	53	Mulan
Medical	$978 \times 1{,}449$	45	Mulan

When the GKS is used in Stage-2, the running time is dominated by the computation of the weights. This gives a running time much faster than the FS for sparse case, and for the dense case with large values of N. It should also be noted that for very sparse data the value of Z_S is $O(k_1)$, so that the running time is dominated by the solution of the system of linear equations $Hz = h$, with a positive definite H. For this case there are known iterative techniques that may obtain an approximate solution much faster. See, e.g., (Golub and Van-Loan, 2013).

6.6 Experimental evaluation

We conducted an extensive set of experiments to evaluate the proposed two-stage approach. Some of these results are shown here and some others are available as additional material. The datasets we used are publicly available, with the details shown in Table 6.4:

The first experiment is for unsupervised feature selection. The results are shown in Table 6.5. Our goal was comparing our method of computing weights to the one described in (Boutsidis et al., 2009) which we call the BMD. The BMD was implemented as suggested by the authors, computing the best result in 40 random runs. In all cases the Stage-1 classifier was Leverage Score, and the Stage-2 classifier was the GE. The reported error is the ratio between the computed error and the initial error with no features selected. The only difference between the BMD and our algorithm was the way in which the weights were calculated. It is clear that our results are more accurate in all test cases.

Table 6.5: Comparison between the BMD two-stage method of Boutsidis et al. and our method. $k_1 = 4k_2$.

k_2	CNAE-9		TechTC01		Day1	
	BMD	ours	BMD	ours	BMD	ours
25	0.110	**0.093**	0.057	**0.050**	0.225	**0.208**
50	0.082	**0.068**	0.023	**0.018**	0.195	**0.176**
75	0.068	**0.052**	0.008	**0.007**	0.174	**0.158**
100	0.057	**0.042**	**0.003**	**0.003**	0.164	**0.146**

The second experiment is for the supervised case, where there is a single label associated with each data item (N=1). The results are shown in Table 6.6. The algorithm used in Stage-1 is shown in the first row, and the algorithm used in Stage-2 is shown in the second row. The first two columns describe "competitors", where Stage-2 is implemented with supervised algorithms. Observe that the last two columns correspond to the case where both the Stage-1 and the Stage-2 algorithms are unsupervised. While there are many cases where the competitors beat our method, the results are very similar.

The third experiment is for the multi-label case, where there are multiple labels associated with each data item. The results are shown for the Enron dataset (in Figure 6.4) and for the Medical dataset (in Figure 6.5). Each Stage-1 algorithm was tested with two Stage-2 algorithms: the GKS and the FS. The results obtained with the FS are our competitors. Since the "Random" and the "FKV" methods depend on random drawings, they were repeated 20 times and averaged.

It is clear that typically using the FS in the second stage gives more accurate results, but the difference is very small. As shown in Section 6.5 using the FS comes with the heavy price of a much slower run time.

6.7 Discussion

In situations where feature selection is too slow one may be forced into compromising accuracy for speed. This leads to the widely used two-stage approach, where an initial pass over the data

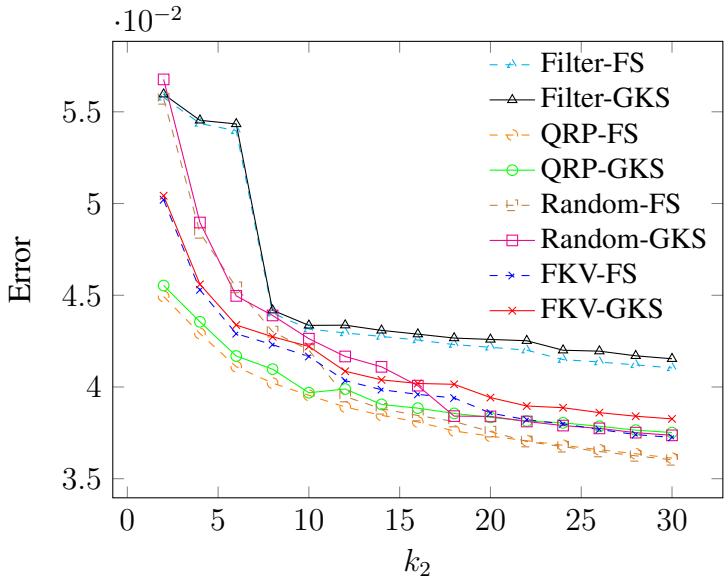


Figure 6.4: Error comparison among different algorithms on the Enron dataset. $k_1 = 10k_2$.

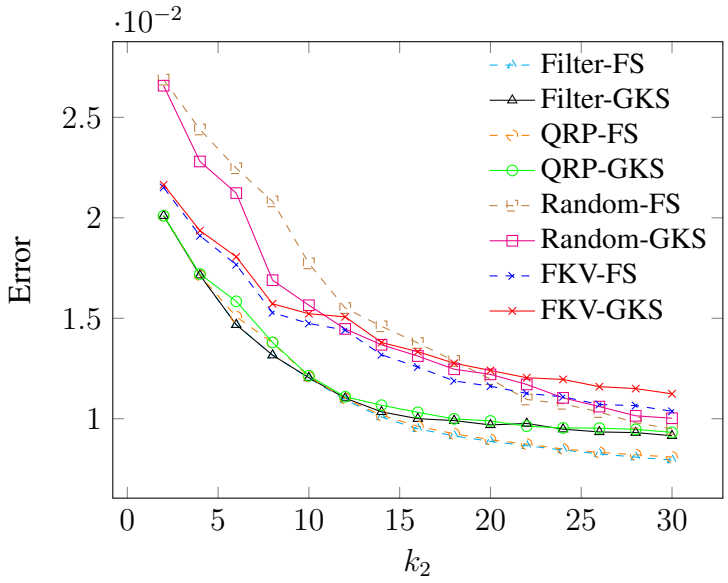


Figure 6.5: Error comparison among different algorithms on the Medical dataset. $k_1 = 10k_2$.

Table 6.6: Experiments with supervised feature selection, predicting a single label. $k_1 = 4k_2$,

k_2	Pearson				FKV	LEV
	SVM	FS	GE	GKS	GE	GE
Isolet $(1,559 \times 618)$						
10	-	0.249	0.279	0.289	**0.188**	0.207
20	0.179	**0.163**	0.244	0.220	0.172	0.192
30	0.212	**0.136**	0.200	0.173	0.155	0.168
40	0.195	**0.125**	0.199	0.168	0.137	0.162
50	0.145	**0.117**	0.192	0.162	0.125	0.150
Gisette $(6,000 \times 5,000)$						
10	-	0.810	0.828	0.826	0.600	**0.456**
20	-	0.679	0.772	0.771	0.510	**0.363**
30	-	0.626	0.741	0.742	0.445	**0.341**
40	-	0.590	0.709	0.709	0.378	**0.319**
50	-	0.573	0.692	0.698	0.390	**0.297**
Arcene (100×10000)						
10	0.812	0.656	0.850	0.850	0.665	**0.615**
20	0.595	0.490	0.501	0.501	0.471	**0.401**
30	0.414	**0.257**	0.412	0.410	0.388	0.283
40	0.332	**0.168**	0.376	0.394	0.312	0.214
50	0.233	**0.094**	0.246	0.210	0.234	0.140

selects candidate features that are evaluated more thoroughly in a second stage. The obvious reason why one may want to use unsupervised feature selection in the second stage is that unsupervised algorithms may be faster than supervised algorithms. However, it was not previously known that it is possible to use unsupervised algorithms in a general setting. The results of (Boutsidis et al., 2009) have shown such a case for a particular (randomized) algorithm in Stage-1, and the approach appeared to be closely tied to that particular algorithm.

In this chapter we describe a general approach that enables combining two off-the-shelf algorithms for performing the selection. (The algorithm for the second stage must be unsupervised.) As we show, this requires the nontrivial computation of weights. The runtime analysis shows the advantage of this approach, especially for the multi-label case and for highly sparse data. In terms of accuracy, we compared the results to a typical setting where Stage-2 is implemented with a

supervised algorithm. Our results show that using unsupervised Stage-2 algorithms with weights calculated by our method is almost as accurate as using supervised algorithms, while at the same time it may be significantly faster.

At the moment it is not clear what particular choice of selection algorithms should work well with our approach. We expect additional studies to discover algorithm pairs that work well together. We also expect that using more accurate feature selection algorithms for the second stage may allow significant improvement in the overall accuracy.

CHAPTER 7

FORWARD SELECTION FOR MULTI-LABEL CLASSIFICATION WITH UNIVARIATE MONOMIAL KERNELS

In multi-label learning, each sample is associated with a set of labels. It has attracted a lot of interest in recent years. As an effective data preprocessing step, feature selection plays an important role as removing irrelevant and redundant features will enhance the performance of classifiers. Most current multi-label feature selection algorithms involve a regularization term in the objective function to achieve feature selection. The solution is obtained iteratively until convergence. In this chapter, we describe an algorithm to address multi-label classification problem by selecting features using forward selection. It is numerically stable and efficient. The best number of features to use can be determined by cross-validation. The forward selection algorithm is further generalized by adding monomial kernels which depend on a single variable. Experimental results on real-world data demonstrate the effectiveness of the proposed algorithms.

7.1 Introduction

Nowadays, multi-label classification has attracted great interest where each instance is associated with multiple labels simultaneously. It is involved in the areas such as image annotation (Boutell et al., 2004), protein function classification (Diplaris et al., 2005) and text categorization (Katakis et al., 2008). Multi-labeled data often has noisy, irrelevant and redundant features of high dimensionality. Feature selection can be used to reduce the high dimensional space. Previous studies utilize a regularization term to achieve feature selection. In (Nie et al., 2010), the robust feature selection method minimizes the joint $\ell_{2,1}$-norm on both loss function and regularization. It is robust to outliers in data points and is able to select features across all data points with a joint sparsity. In (Jian et al., 2016), it exploits the label correlations in the output space to find features that are shared across multiple labels. A novel convex algorithm for large-scale multi-label feature selection is proposed in (Chang et al., 2014). These solutions are solved iteratively until convergence.

Input: X, Y, k. Output: S.

0. $S = \{\}$

1. Run iteration $j = 1, \ldots, k$:
  1.1 For $i = 1, \ldots, n$:
    compute $f(x_i)$, a criterion to evaluate x_i.
  1.2 $x_{s_j} = \arg\max f(x_i)$. Add x_{s_j} to S.

Figure 7.1: The greedy framework for feature selection.

By contrast, we propose an algorithm that achieves feature selection using a greedy approach named "forward selection", which is numerically stable and efficient. The best number of features to use can be determined by cross-validation. The proposed algorithm is further generalized by adding monomial kernels which depend on a single variable. Experimental results show that with the monomial kernels, less features are needed and the accuracy is comparable to current state of the art.

7.2 Forward selection for multi-label classification

7.2.1 Forward selection

Let X be a dictionary matrix of size $m \times n$. Let Y be a data matrix of size $m \times N$. The problem we consider is to select k columns from X to approximate the matrix Y, where k is a relatively small number. In Frobenius norm the approximation error is:

$$\min_{A} \|Y - SA\|_F^2 \tag{7.1}$$

Here S is the selected subset and A is the coefficients matrix.

In many situations, the optimal solution is known to be NP-hard (Çivril, 2017; Shitov, 2017). Common approaches compute an approximate solution using greedy strategy. Among the classic greedy algorithms, forward selection is slightly expensive but more accurate than Matching Pursuit (Mallat and Zhang, 1993) and Orthogonal Matching Pursuit (Davis et al., 1994). In literature,

the naming of the various greedy algorithms is a little bit confusing. Forward selection is also called OMP(e.g.(Gharavi-Alkhansari and Huang, 1998)), OOMP(e.g. (Rebollo-Neira and Lowe, 2002)) and OLS(e.g. (Chen et al., 1989)) .

The general framework of the greedy algorithms is shown in Figure 7.1. It run k iterations to select k features. At each iteration, evaluate each feature based on a certain criteria f and select the best among all the features. Specifically, in forward selection the measurement $f(x_i)$ is defined as the error reduction to approximate Y after adding the feature x_i.

Let x_{s_j} be the selected column at iteration j. Let S_j be the selected columns after iteration j, define $S_j = (S_{j-1}, x_{s_j})$. Let Q_j be the orthogonal basis of S_j and let q_j be the last column of Q_j, so that $Q_j = (Q_{j-1}, q_j)$. Based on the definition, q_j is the vector $(I - Q_{j-1}Q_{j-1}^T)x_{s_j}$ after normalization. Let R_j denote the remaining matrix needs to be approximate after iteration j. Then:

$$\begin{aligned} \|R_j\|_F^2 &= \min_A \|Y - S_j A\|_F^2 = \|Y - Q_j Q_j^T Y\|_F^2 = \|Y - Q_{j-1}Q_{j-1}^T Y - q_j q_j^T Y\|_F^2 \\ &= \|R_{j-1}\|_F^2 - \|q_j^T Y\|_F^2 = \|R_{j-1}\|_F^2 - \|q_j^T R_{j-1}\|_F^2 \end{aligned} \tag{7.2}$$

Before the iteration j starts, $j-1$ columns have already been selected. Thus R_{j-1} is fixed. Equation 7.2 shows that the criterion $f(x_i)$ should be defined as $\|q_i^T R_{j-1}\|_F^2$. The best selection x_{s_j} satisfies the condition that $x_{s_j} = \arg\max_{x_i} \|q_i^T R_{j-1}\|_F^2$. After adding the feature x_{s_j}, the error to approximate matrix Y is decreased by $\|q_j^T R_{j-1}\|_F^2$.

7.2.2 Forward selection for multi-label classification

The procedure of the training phase for multi-label classification using forward selection is shown in Figure 7.2. The training data are the features matrix X_{train} and labels matrix Y_{train}. A forward selection algorithm is then applied on the training data to select a certain number of features. With the selected features, the linear regression models A in Equation.7.1 can be solved easily. Each column in A can be viewed as a predictor. The optimal thresholding is then applied to determine the threshold for label classification.

Input: X_{train}: the feature matrix for training, size $m \times n$
Y_{train}: the label matrix for training, size $m \times N$
k: number of features to be selected.

Output: The selection S and N predictors for each label.

1. Run the Forward Selection Algorithm on X_{train}, Y_{train} to obtain k selected features.

2. $S = \{$the indices of selected features$\}$.

3. For each target attribute y_i in Y_{train}:
Train a predictor using y_i and the k selected features from X_{train}.

4. Output S and the N predictors.

Figure 7.2: Training phase for multi-label classification using forward selection.

Input: x_{test}: a test sample, size $1 \times n$
S: the indices of selected features
N predictors for each label

Output: Prediction for the N labels, size $1 \times N$

1. Use S to get the selected features in x_{test}.

2. For $i = 1, \ldots, N$:
Predict a label attribute using the selected features in x_{test} and predictor i.

3. Output the prediction for the N labels.

Figure 7.3: Testing phase for multi-label classification using forward selection.

Figure 7.3 shows the procedure of the testing phase. The prediction is obtained using the selected features in the test samples.

The best number of features to use can be determined by cross validation. Given a set of examples, one round of cross-validation involves partitioning the data into complementary subsets, performing the training on one subset following the steps in Figure 7.2 , and validating the analysis on the other subset as shown in Figure 7.3. To reduce variability, multiple rounds of cross-validation are performed using different partitions and the validation results are averaged over the rounds. The k that produces the smallest classification error is the desired best number of features to use.

Table 7.1: Dataset description.

Dataset	Instances	Features	Labels
emotions	391	72	6
medical	978	1449	45
scene	1211	294	6
yeast	1500	103	14

We show the fine-tuning process by cross validation on benchmark datasets from Mulan repository (Tsoumakas et al., 2011). The details of these datasets are shown in Table 7.1. The classification error is evaluated in hamming loss. It computes the fraction of labels that are incorrectly predicted, defined as follows.

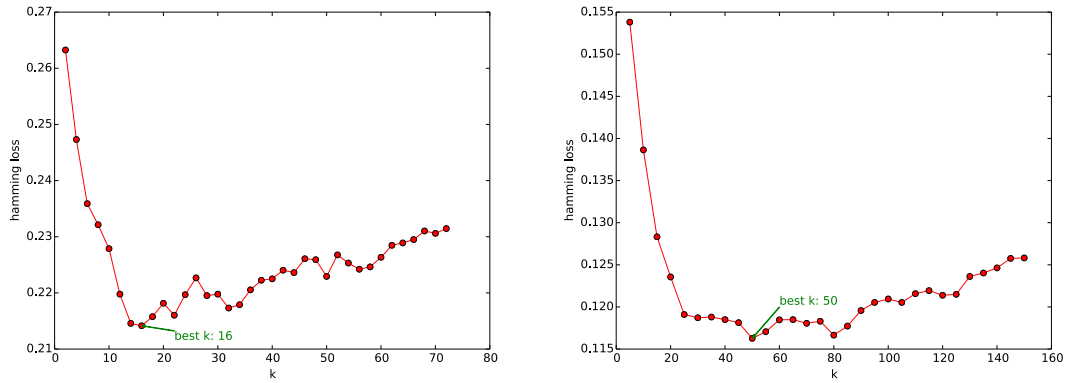
$$\text{HammingLoss}(y, \hat{y}) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{\text{xor}(y, \hat{y})}{|N|}$$

Here $|D|$ is the number of examples, N is the number of labels, y are the ground truth labels and $\hat{y}$ are the predicted labels.

Figure 7.4 shows that with the increasing number of selected features the classification error goes down first, then up to a certain point, the error becomes bigger and bigger. It is the classic overfitting problem in machine learning, that the model fits the training data well but fails to generalize to the unseen data. Therefore, it is very important to determine the best number of features to use. Typically using a small number of features is better than using the entire feature set.

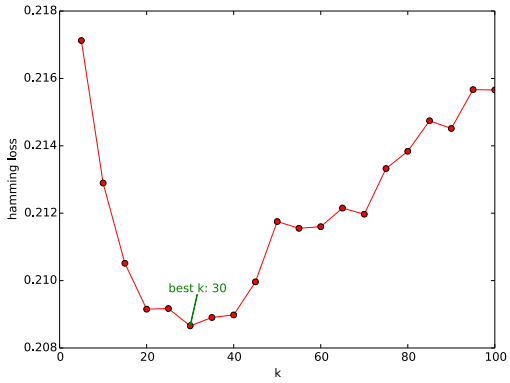
We evaluate the algorithm with the best k obtained in the fine-tuning process. The competitors are the current state-of-the-art feature selection methods MIFS (Jian et al., 2016) and RFS (Nie et al., 2010). For a fair comparison, we tune the regularization parameters for the two methods with a grid-search strategy. Parameter values are varied in the range of {0.0001, 0.001, 0.01, 0.1, 0.2, 0.4, 0.6, 0.8, 1, 10}. We report the results with the best parameters. Experiments are run with 5-fold cross validation and repeated 10 times with different shuffles then averaged. The labels are classified by linear regression models with optimal thresholding.

The results are shown in Table 7.2. Forward selection is comparable with these methods. We discuss an algorithm that improves the performance next section.

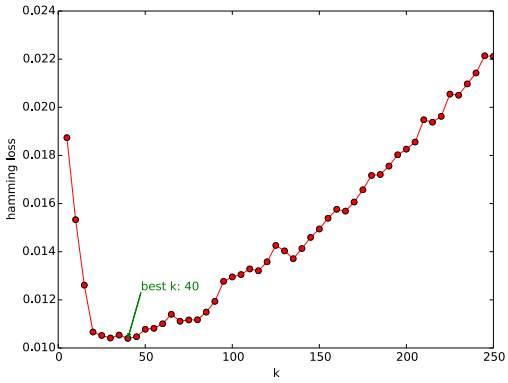


(a) dataset emotions

(b) dataset scene



(c) dataset yeast



(d) dataset medical

Figure 7.4: Fine-tune the best number of features with Forward Selection on Mulan datasets.

Table 7.2: Comparison with MIFS and RFS.

Dataset	MIFS	RFS	Forward Selection
emotions	0.197	0.215	0.207
medical	0.009	0.044	0.010
scene	0.109	0.120	0.107
yeast	0.197	0.197	0.203

Input: X, Y, k, p. Output: S.

0. $S = \{\}$

1. Run iteration $j = 1, \ldots, k$:
  1.1 For each feature x_i:
    1.1.1 Create monomial features $x_i^2, x_i^3, \ldots, x_i^p$ from x_i. Let $T_i = \{x_i, x_i^2, x_i^3, \ldots, x_i^p\}$.
    1.1.2 Orthogonalize the features in T_i.
    1.1.3 For each feature x_i^t in T_i: compute $f(x_i^t)$.
  1.2 $x_{s_j} = \arg\max \sum_{t=1}^{p} f(x_i^t)$. Add $x_{s_j}, x_{s_j}^2, \ldots, x_{s_j}^p$ to S.

Figure 7.5: Improved Forward selection with univariate monomial kernels.

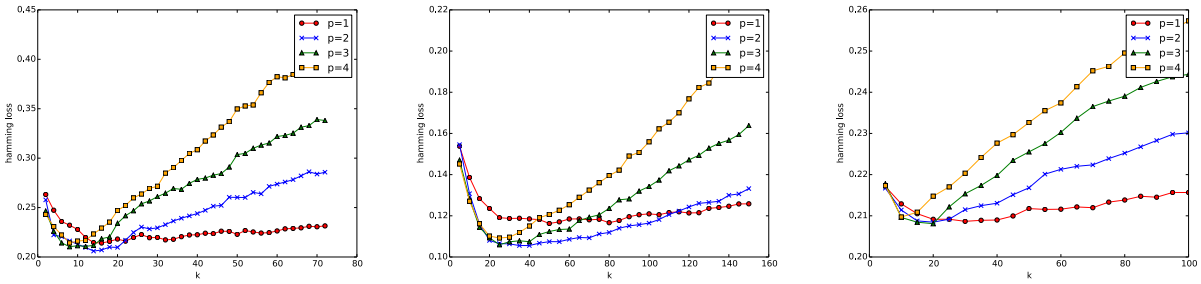
7.3 Forward selection for multi-label classification with univariate monomial kernels

Kernel features are widely used in many machine learning algorithms, e.g. support vector machine (Suykens and Vandewalle, 1999) and neural network (Demuth et al., 2014). We consider creating monomial kernel features to enhance the performance of the classifiers. Once a feature is selected, the monomial features generated by it can also be used. Specifically, these monomial features are computed by taking the values of the feature vector to a certain power p element-wise.

As shown in Equation 7.2, at one iteration, the feature that reduces the error the most to approximate the remaining matrix will be selected. With the monomial kernels, we proposed a new selection criterion. The importance of a feature is evaluated by itself and the monomial features generated by it.

The procedure of the improved forward selection with univariate monomial kernels is shown in Figure 7.5. For a specific feature x_i, let T_i be the feature set contains feature x_i and all the monomial features generated by x_i. The key point is to orthogonalize the monomial features in T_i. Under such condition, the contribution of T_i for reducing the approximation error can be computed as the summation of the contribution of each individual feature in the set T_i.

Fine-tune the best number of features to use and the best power p to generate univariate monomial kernels proceed similarly as before. The results are shown in Figure 7.6. When p is set to be 1, it indicates that no monomial kernels are used. The best k becomes smaller with a larger p. A



(a) dataset emotions (b) dataset scene (c) dataset yeast

Figure 7.6: Fine-tune the parameters for the improved forward selection algorithm on Mulan datasets.

Table 7.3: Comparison with RFS, MIFS and FS.

Dataset	RFS	MIFS	FS	Improved FS
emotions	0.199	0.211	0.211	**0.198**
scene	0.113	0.129	0.111	**0.099**
yeast	**0.198**	0.199	0.206	0.202

higher power typically makes the problem of overfitting worse. For dataset emotions and scene, the best power to use is 2. For dataset yeast, the best power to generate the monomial features is 3.

Table 7.3 shows the comparison with RFS, MIFS and Forward Selection (FS) with the fine-tuned parameters. The results are averaged over 10 runs of 5-fold cross validation. The proposed algorithm performs the best on the datasets emotions and scene. For dataset yeast, it outperforms than forward selection and is comparable with RFS and MIFS. The advantage of the improved forward selection algorithm over RFS and MIFS is that it does not require regularization to achieve feature selection, thus in practice it runs much faster.

7.4 Discussion

In this chapter, we proposed an algorithm that can effectively select features for multi-label classification. Unlike other studies that achieve feature selection by regularization, our method solves an optimization problem which is numerically stable. We come up with the idea that the best number of features to use can be determined by cross validation. Experimental results show that using a

small set of features outperforms using the entire feature set. The proposed algorithm is further generalized by using univariate monomial kernel features. Specifically, these monomial features are generated from a single feature vector and can be viewed as almost "free" to use. The best number of features to use is further reduced. The proposed algorithms are shown experimentally to be comparable to the current state of the art.

CHAPTER 8

GREEDY SPARSE APPROXIMATION OF A MATRIX IN TERMS OF ANOTHER MATRIX UNDER UNITARILY INVARIANT CRITERIA

Chapter 5 describes the solution for column subset selection under unitarily invariant criteria. Here we study a more general case that approximates a matrix in terms of a small number of columns selected from a dictionary under such error criteria. The column subset selection problem can be viewed as a special case when the dictionary is the same as the given matrix. There are known greedy algorithms for the problem that minimize the error in Frobenius norm. We generalize this approach and describe a fast greedy algorithm for a large family of unitarily invariant criteria. In particular we show minimizing Schatten p-norms with $0 < p < 1$ can be used to solve the outlier-robust PCA problem. The standard methods for rank minimization are gradient-based, using convex nuclear norm. Several recent studies utilize non-convex Schatten p-norms to approximate the rank function. The greedy approach we proposed is different from the previous work. The algorithm is very efficient and only requires a single pass of eigenvalue decomposition. Experimental results show that our algorithm outperforms other convex-based outlier-robust PCA method. We also show the application to multi-label classification problem using the proposed approach.

Portions of this chapter are the results of collaboration with Tongyi Cao.

8.1 Introduction

Let X be a dictionary matrix of size $m \times n$ and let Y be a data matrix of size $m \times N$. The problem that we consider is to select a subset S from X to approximate the matrix Y, where S consists of k columns.

$$Y \approx SA \tag{8.1}$$

Each column of Y is approximated by a linear combination of the k columns in S, and A is the coefficients matrix. Let $R = Y - SA$ be the error matrix of the approximation (8.1). The most

common criterion to evaluate the approximation in Equation (8.1) is using $\|R\|_F$, the Frobenius norm of R. Find the subset S which gives the smallest $\|R\|_F$ is known to be NP-hard even for $N = 1$ (Natarajan, 1995). A common approach is to obtain an approximate solution using greedy algorithm. Previous studies that use this approach includes (Soussen et al., 2013), which analyzes cases where the greedy approach is capable of recovering the exact signal. (Das and Kempe, 2011) shows that the greedy forward selection produces nearly optimal results. (Maung and Schweitzer, 2015) shows how to implement the forward selection algorithm efficiently.

We are interested in error criteria other than the Frobenius norm. For example, the nuclear norm attracts a lot of attention as convex surrogate for rank minimization problem (Candès and Tao, 2010). It shows promising results in Robust PCA (Candès et al., 2011) (Xu et al., 2010). However, since nuclear norm is the ℓ_1 norm of the singular values, it tends to over-penalize large entries. A more accurate approximation of the rank function can be obtained by Schatten p-norms with $0 < p < 1$. It outperforms the nuclear norm in various problems including matrix completion (Nie et al., 2012) (Marjanovic and Solo, 2012) and images recovery (Lu and Zhang, 2015).

These studies mentioned above avoid the discrete rank function and sparsity constraints by relaxing them to obtain a continuous regularization problem. The continuous problem is then solved using gradient-based optimization approaches, either convex or non-convex. We adopt a different approach and show how the greedy selection algorithm can be generalized to minimize a general class of unitarily invariant criteria, including the Schatten p-norms. In particular we show that our algorithm can be applied to outlier-robust PCA problem (Xu et al., 2010), i.e. to separate a matrix to a low rank component and a column sparse component. To the best of our knowledge, there is no greedy method of Schatten p-norms minimization for Robust PCA. Our algorithm is very efficient and only requires a single pass of Eigenvalue Decomposition (EVD), while most of the gradient-based optimization methods require SVD or EVD for every iteration.

The problem being addressed

Our goal is to compute a selection S that gives good approximation in Equation (8.1) with various

$\lambda(A)$	-	eigenvalues of a matrix A.
$\|\cdot\|_*$	-	nuclear norm
$\|\cdot\|_p$	-	Schatten p-norms
$\|\cdot\|_{p,q}$	-	$\ell_{p,q}$ norm
X^+	-	generalized inverse of matrix X
S	-	a subset of matrix X

Figure 8.1: The relevant notation.

error criteria. Let Θ be an error function $\Theta : \mathbb{R}^{m\times n} \mapsto \mathbb{R}$. We wish to determine a selection S from the dictionary matrix X such that $\Theta(R)$ is small, where $R = Y - SA$. This problem can be written in an equivalent form.

$$\begin{aligned} &\min_{C} \ \Theta(Y - XC) \\ &\text{s.t.} \quad \|C\|_{\text{row-0}} \leq k \end{aligned} \tag{8.2}$$

Here $\|C\|_{\text{row-0}}$ stands for the number of non-zero rows in C. Each non-zero row corresponds to a column in the selection S. Clearly, Θ can be any matrix norm, but one may be interested in error criteria that are not norms. The function can be non-smooth and even discontinuous such as the unitarily invariant monotonic functions defined in Chapter 5. The approach that we proposed is to generalize the greedy algorithm used in previous studies to such error criteria.

Notation

The relevant notation in this chapter is shown in Figure 8.1.

Chapter organization

The chapter is organized as follows. First we introduce a classic greedy method named Forward Selection. Then we generalized the Forward Selection algorithm and describe an efficient method for calculating the selection. Lastly, we show an application of the proposed algorithm on outlier robust PCA problem and present experimental results with comparison to the state-of-the-art method.

8.2 Generalized forward selection

8.2.1 Forward selection

The greedy approach we consider here is the Forward Selection algorithm introduced in Chapter 7 Section 2. Let x_{s_j} be the selected column at iteration j. Let S_j be the selected columns after iteration j, define $S_j = (S_{j-1}, x_{s_j})$. Let Q_j be the orthogonal basis of S_j and let q_j be the last column of Q_j, so that $Q_j = (Q_{j-1}, q_j)$. Based on the definition, q_j is the vector $(I - Q_{j-1}Q_{j-1}^T)x_{s_j}$ after normalization. Let R_j denote the remaining matrix needs to be approximate after iteration j. Recall that the problem is:

$$\begin{aligned} \|R_j\|_F^2 &= \min_A \|Y - S_j A\|_F^2 \\ &= \|Y - S_j(S_j^+ Y)\|_F^2 = \|Y - Q_j Q_j^T Y\|_F^2 \\ &= \text{Trace}((R_{j\text{-}1} - q_j^T R_{j\text{-}1})(R_{j\text{-}1} - q_j^T R_{j\text{-}1})^T) \\ &= \|R_{j\text{-}1}\|_F^2 - \|q_j^T R_{j\text{-}1}\|_F^2 \end{aligned} \tag{8.3}$$

Observe that we do not need to calculate the remaining matrix R_j explicitly and the cost of evaluating a feature is simply a dot product. This is due to the fact that the Frobenious norm can be expressed as trace, which is linear.

To generalize the Forward Selection algorithm to unitarily invariant criteria, two issues need to be solved. First, the solution of the optimization problem in equation 8.3 relies on the generalized inverse. It is true when the error is measured in Frobenious norm. For other error criteria such as unitarily invariant functions, it is proved in Chapter 5 Section 2 that the generalized inverse is the optimal solution. Second, error functions other than the Frobenious norm typically do not hold the property of linearity. Compute the remaining matrix R_j and its singular values for each x_i is pretty expensive. We describe a fast algorithm for computing the singular values without explicitly calculating R_j by exploiting a special structure.

8.2.2 The efficient calculation of the eigenvalues

We use a similar technique to (Arai et al., 2015) that computes the singular values efficiently without calculating R_j. The main idea is to perform the calculation on a matrix related to R_j that has a special structure which allows fast calculation. The method utilizes the information in the previous iteration and only requires a single pass of eigenvalue decomposition in the beginning.

There are three different types of eigenvalue calculation involved. The first is calculated once for the initialization step, the second is calculated after a column is selected, and the third is calculated for the evaluation of each column.

Initial eigenvalue decomposition

In the beginning we compute the eigenvalues and the eigenvectors of the matrix YY^T.

$$YY^T = V\Lambda V^T$$

Here V are the eigenvectors, and Λ are the corresponding eigenvalues. If the rank of Y is r, then the dimension of V is $m \times r$ and Λ is $r \times r$.

Eigenvalue decomposition for selected columns

This eigenvalue and eigenvector calculation is performed at step 1.2 of the algorithm shown in Figure 7.1, note that it is calculated after a column is selected. To prepare for the next iteration, we need both the eigenvectors and the eigenvalues of the matrix $R_j R_j^T$.

Eigenvalues for evaluating columns

The evaluation of columns x_i at step 1.1 shown in Figure 7.1 is the most time consuming part of the algorithm. We show how to compute these values efficiently from the eigenvalue decomposition in the previous iteration.

Let the eigenvectors and eigenvalues calculated in the j-1 iteration be $V_{j\text{-}1}$ and $\Lambda_{j\text{-}1}$. At iteration j we have:

$$\begin{aligned}\lambda(R_j R_j^T) =& \lambda\Big((I - q_j q_j^T) R_{j\text{-}1} R_{j\text{-}1}^T (I - q_j q_j^T)\Big) \\ =& \lambda\Big((I - q_j q_j^T) V_{j\text{-}1} \Lambda_{j\text{-}1} V_{j\text{-}1}^T\Big) \\ =& \lambda(\Lambda_{j\text{-}1} - z z^T)\end{aligned}$$

Here $z = \Lambda_{j\text{-}1}^{1/2} V_{j-1}^T q_j$. The derivation above is straight forward considering the invariance of eigenvalues under cyclic permutation and q_j is a unit vector. Since the matrix $R_j R_j^T$ can be expressed as rank-1 updates to a diagonal matrix, its eigensystem can be calculated fast by specialized routines. See, e.g (Bini and Robol, 2014; Melman, 1998; Golub, 1973). For evaluating the column we only need the eigenvalues. After a column is selected we calculate its eigenvalues also by the specialized routines to prepare for the next iteration. Our implementation uses Gragg's method (Melman, 1998). This is an iterative procedure with fast convergence speed.

8.2.3 Pruning

The Gragg's method (Melman, 1998) calculates the eigenvalues of the children in descending order. Thus we have a lower bound on the function value of a column when a portion of the eigenvalues are calculated. Note that we only need to select the column with the smallest value. If the lower bound is larger than the smallest value in the previously calculated columns, we are sure that this column will not be selected, and the rest of the calculation for that column can be pruned.

8.3 Outlier-Robust PCA

We proceed to show an application of the Generalized Forward Selection algorithm on the outlier-robust PCA problem.

Principle component analysis is one of the most important tools for high dimensional data analysis and is well known for its sensitivity to outliers. A classic result shows that it is possible to

recover the low rank structure from an arbitrarily but sparsely corrupted data matrix (Candès et al., 2011). However this method fails when the entire data point or columns are corrupted, which is common due to the error in data collection process or because a few points do not conform with the low rank subspace of the majority. This problem was first suggested in (Xu et al., 2010), where they named these columns outliers. They present an algorithm named Outlier Pursuit that adopts a convex optimization approach. The algorithm has been successfully applied on various problems including collaborative filtering (Chen et al., 2011) and image annotation (Dong et al., 2013). New theoretical analysis shows that the algorithm has a more extensive working range than it was shown previously (Zhang et al., 2015).

We show how the Generalized Forward Selection algorithm can be used to partition the matrix into a low-rank and a column-sparse component by Schatten p-norms minimization. It is different from the Outlier Pursuit algorithm in two aspects. First, it uses Schatten p-norms to approximate the rank function while the Outlier Pursuit uses the nuclear norm. The Schatten p-norms has been shown to outperform the nuclear norm in other rank minimization problems including matrix completion (Nie et al., 2012) (Marjanovic and Solo, 2012) and images recovery (Lu and Zhang, 2015). Second, the previous work in Robust PCA (e.g.(Candès et al., 2011) (Xu et al., 2010)) relax the discrete sparsity constraints to continuous norms that can promote sparse solutions, and solve the regularization problem using gradient-based optimization methods. We adopt a different approach and enforce the sparsity by selecting a few columns greedily.

The Schatten p-norms are defined as the p norms of the vector of the singular values of a matrix:

$$\|A\|_p = \left(\sum_{i=1}^{rank(A)} \sigma_i^p(A) \right)^{1/p} \tag{8.4}$$

It contains some common norms such as Frobenius norm ($p = 2$), nuclear norm ($p = 1$) and spectral norm ($p \to \infty$). For $p < 1$ the triangular inequity is violated and it is no longer a norm. But clearly for any $p \in (0, \infty)$, equation 8.4 is a unitarily invariant monotonic function. It can be used as error criteria for the Generalized Forward Selection algorithm.

8.3.1 From sparse approximation to robust PCA

The classic Frobenious norm Forward Selection is an algorithm for sparse approximation. The assumption for sparse approximation is that the data matrix Y can be well approximated by linear combination of a few columns in the dictionary X. We minimize the Frobenius norm of the remaining matrix as the noise is expected to be small in magnitude. When the Schatten p-norms are used as error criteria, instead of expecting the remaining matrix to be small in magnitude, we expect it to be low rank. The subset is selected to let the remainder matrix become low rank. Both the selection and the remaining matrix can be arbitrary in magnitude, and the selection part is column sparse. This is the same idea as outlier-robust PCA.

8.3.2 Problem setup

Assume the data matrix M consists of a true low rank component L and a column sparse outliers K. (Conventionally the sparse component is denoted as S. Here it has been used for the selection.)

$$M = L + K \tag{8.5}$$

The goal is to recover the column space of L and the column support of K. To achieve this goal the Outlier Pursuit algorithm (Xu et al., 2010) uses the following objective function:

$$\begin{aligned} &\min_{L,K} \|L\|_* + \lambda \|K\|_{2,1} \\ &\text{s.t.} \quad M = L + K \end{aligned} \tag{8.6}$$

The nuclear norm and the $\ell_{2,1}$ norm are the convex surrogates of the rank and the number of non-zero columns of a matrix respectively. We can recast the regularization problem as the following problem:

$$\begin{aligned} &\min_{L} \|L\|_* \\ &\text{s.t.} \quad M = L + K, \quad \|K\|_{2,0} \leq k \end{aligned} \tag{8.7}$$

We consider the above formulation is more clear to represent the problem. The separation of a low-rank and a column sparse matrix suffers from identifiability problem. A matrix with only one non-zero column is both column sparse and low rank. The column incoherence property used in (Xu et al., 2010) dose not clear this ambiguity. Besides it is not clear what value a column in L will have the corresponding non-zero columns in K. Formulation 8.7 does not have such problem and it is always well defined. Since only L is involved in the objective function, the columns in L that correspond to non-zero columns in K will be zero. Thus the non-zero columns of K are the same as the corresponding columns in M.

We have the following observation. Let e_i be the ith coordinate vector. L can be written as

$$L = (I - \sum_{i \in \text{col-supp}(K)} e_i e_i^T) M \tag{8.8}$$

Here col-supp(K) is the set of the indices of the non-zero columns in K. This is equivalent to selecting a subset from an $n \times n$ identity matrix I. Finally, we use the Schatten p-norms with $p < 1$ to approximate the rank function, replacing the nuclear norm:

$$\begin{aligned} &\min_C \|M - IC\|_p \,, \\ &\text{s.t.} \quad |C|_{\text{row-0}} \leq k \end{aligned} \tag{8.9}$$

It has the same form as the goal function 8.2. We can use the Generalized Forward Selection algorithm to solve this problem. The selection corresponds to outliers and the remaining matrix R_k corresponds to the low rank component L.

There are two points to notice. First, after selection, the singular values of the remaining matrix are already calculated, so the PCA of the low rank matrix L comes with no extra cost. Second, the dictionary used here is orthogonal. For standard greedy selection under the Frobenius norm, this means that the selection can be found in one pass and it is optimal. But it is no longer true for general error criteria due to the non-linearity.

8.3.3 The number of selection

The goal of outlier-robust PCA is to recover the column space of L while identifying the outliers. These two tasks are complementary. Given the outliers, the column space is the PCA on the remainder of the matrix. Given the true column space, setting a simple threshold on the approximation error of each column will give the outliers.

The key point is that the number of selection k does not need to be exact. The true column spaces can be recovered as long as k is an upper bound of the number of outliers. A post-processing threshold can be applied to identify the outliers. Note that the Outlier Pursuit algorithm also requires applying threshold to the column sparse matrix K to identify outliers, since for the real world data none of the columns in K is exactly zero.

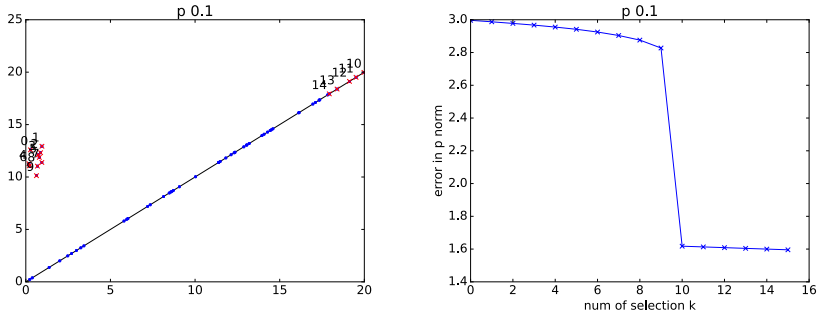
8.3.4 Experimental results

We perform various experiments to see the performance of the proposed algorithm for outlier-robust PCA problem. We compare our algorithm with Outlier Pursuit algorithm on synthetic datasets.

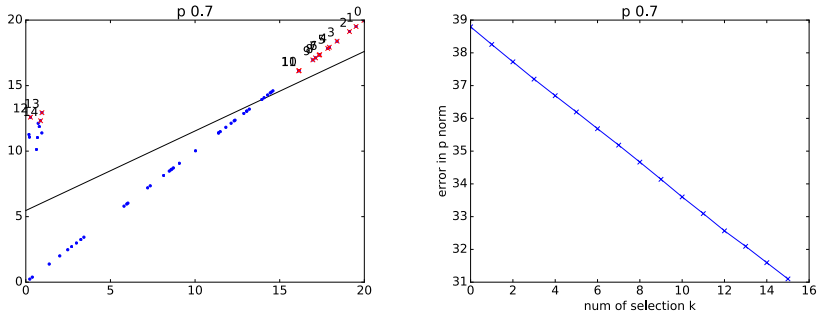
A toy example

First, we demonstrate the selection of outliers and the minimization of the objective function by running the algorithm on data from a 2D line with a few outliers. We test on both noiseless case and noisy cases. There are 50 normal points lying on the true subspace and 10 ouliers that are not. We set $p = 0.1$. k is set to be 50% more than the true number of outliers. The result is shown in Figure 8.2. We show the selection points with the order in which they are selected, and show the decrease of the error in Schatten p-norms for each iteration.

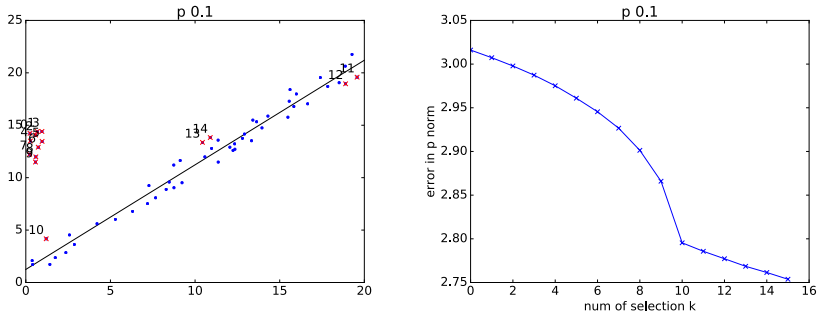
The selection minimizes the objective function in both cases. We can see that $p = 0.1$ is a very good approximation of the rank function. There is a steep jump after the outliers are all selected in the noiseless case, as the rank is reduced from 2 to 1. For the noisy case, there is no jump but



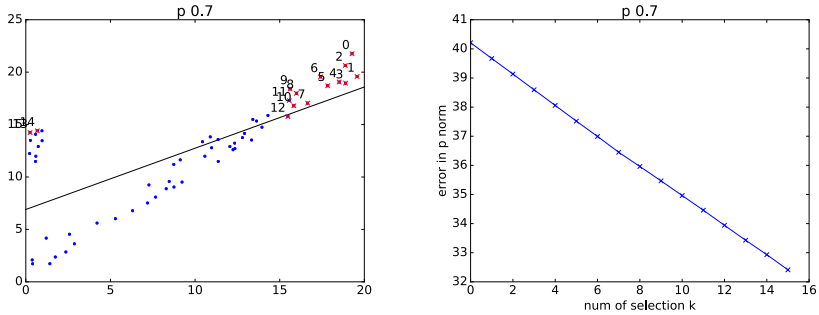
(a) Selection (b) Error



(c) Selection (d) Error



(e) Selection (f) Error



(g) Selection (h) Error

Figure 8.2: The selection of outliers in a noiseless and noisy lines with different p. The selected points are marked with "cross" and the order in which they are selected is shown. We show how the value of the error function decreases as the points are selected.

we can still see the difference before and after the selection. In both cases, even when we select more than the real number of outliers, the true subspace is still recovered. Putting a threshold on the approximation error will give the true outliers. Apparently the value of p is crucial for the algorithm. When $p = 0.7$, the algorithm fails to select all the outliers, and error decrease stably.

Phase transition property

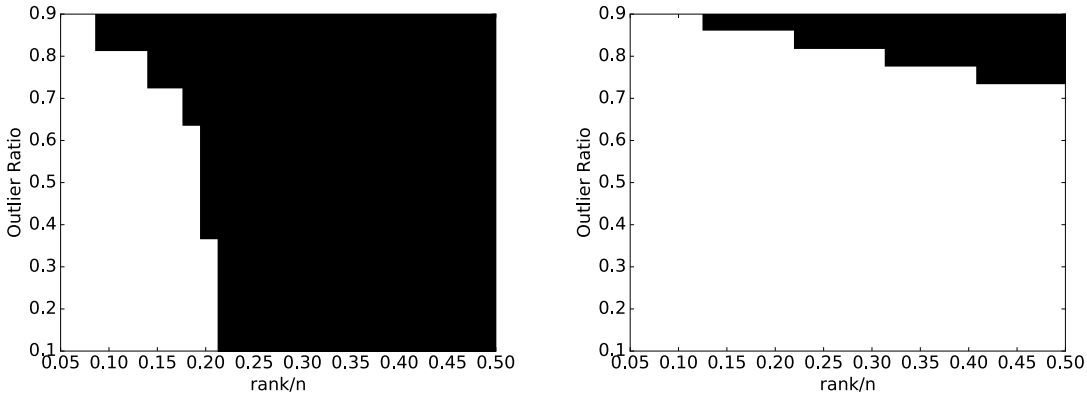
The phase transition property shows the working range of the algorithm. We adopt the same setting as in (Zhang et al., 2015). We fix $m = n = 400$ and test on different ratio of outliers λ and ratio of the dimension of the subspace r. The low rank component is composed as $L = GH^T$, where each entry in $G, H \in \mathbb{R}^{n\times r}$ is drown independently from the distribution $\mathcal{N}(0, 1)$. The sparse component $S \in \mathbb{R}^{m\times n}$ has each entry drown from $\mathcal{N}(0, 1)$ for λn columns and the rest are zero. The data matrix $M = L + S$.

We try different values of p and the results are compared to the Outlier Pursuit algorithm. We set the regularization parameter $\lambda = 1/\sqrt{\log(n)}$ for the Outlier Pursuit algorithm as suggested by the new theoretical analysis (Zhang et al., 2015) and obtain similar result as they reported. For greedy selection we count it as a failure if any column is selected before an outlier. The experiment is repeated for 10 times. The area is marked as black if any failure occurs. The results are shown in Figure 8.3.

The proposed algorithm has significantly larger working range than the Outlier Pursuit Algorithm. It can find all the outliers when the true subspace is 50% of the ambient space and the outliers consist of 70% of the data points with $p = 0.1$. When p is increased, working range becomes smaller.

8.4 Application to multi-label classification problem

In Chapter 7, features are selected by forward selection under Frobenius norm for multi-label classification. It is interesting to see the performance when selecting features under the unitarily invari-



(a) Outlier Pursuit (b) $p = 0.1$

(c) $p = 0.3$ (d) $p = 0.7$

Figure 8.3: Phase transition property of the Outlier Pursuit algorithm (a), and of the outlier selection algorithm for different p values (b) (c) (d).

ant criteria. Experiments are performed on benchmark datasets from Mulan repository (Tsoumakas et al., 2011). The properties of these datasets are shown in Table 8.1. Experimental results are obtained by 10 folds cross-validation using linear regression as classifier. The error is evaluated in hamming loss, which measures the percentage of wrongly classified labels. Hamming loss is defined as follows:

$$HammingLoss = \frac{1}{mN} \sum_{i=1}^{m} \sum_{j=1}^{N} I(\hat{y}_j^i \neq y_j^i)$$

Table 8.1: Dataset description.

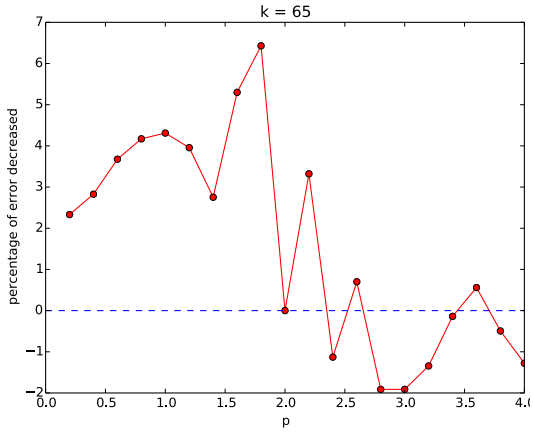
Dataset	Instances	Features	Labels
emotions	593	72	6
medical	978	1449	45
enron	1702	1001	53
scene	2407	294	6

where m is the number of examples, N is the number of labels, $\hat{y}$ is the predicted labels and y is the ground truth labels.

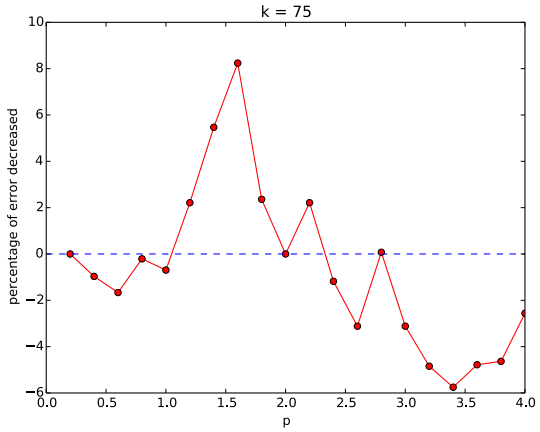
We consider the error using Frobenius norm ($p = 2$) as baseline. For all the other p, we compute the percentage of error decreased compared to the baseline. From the results shown in Figure 8.4, we can see that the percentage of the decreasing error is 8% for dataset medical when k is set to be 75 and p is 1.6. Figure 8.5 shows that the percentage of the decreasing error is around 6% for dataset emotions when k is 4 and p is 4.8. For dataset enron and scene (Figure 8.6, 8.7), the reduced error is not that significant. It is roughly 1% and 2% respectively. For dataset emotions, when k is 4, using a bigger p reduces the error significantly. However, the best p is about 1.2 when k is 8. It shows that best p is dependent on the number of selected features k. For dataset enron, when k is 14, the performance becomes better with an increasing p. While for dataset emotions, a smaller p produces better results. The best p can be determined by cross-validation.

8.5 Discussion

We generalized a classic Forward Selection algorithm and describe a fast algorithm to handle a general class of unitary invariant error criteria. We show that the algorithm can be used to solve the outlier-robust PCA problem by Schatten p-norms minimization. The solution is novel and is very different from previous approaches. Experimental results show that our algorithm outperforms the convex optimization based method. The algorithm also can be applied to address the multi-label classification problem. Experimental results show that the best subset of features are not necessarily selected under Frobenius norm. Instead, using Schatten p-norms performs better. The

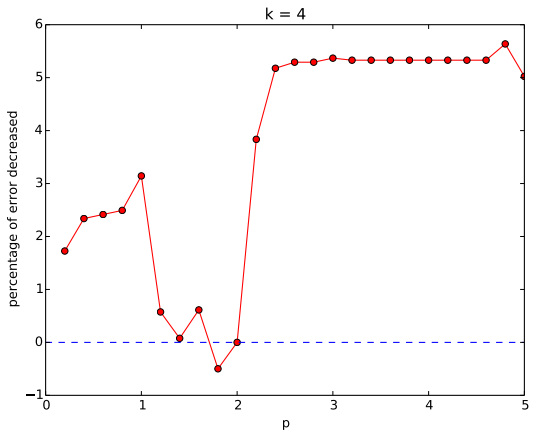


(a) k = 65

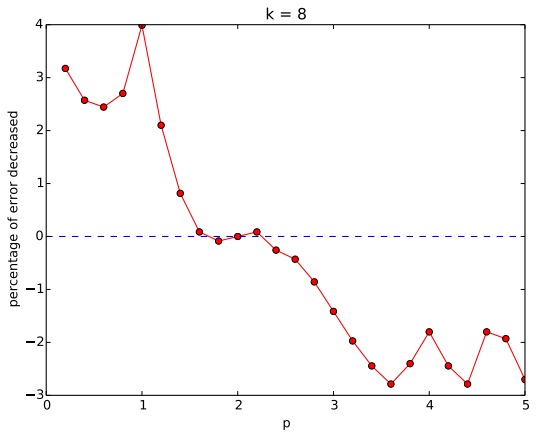


(b) k = 75

Figure 8.4: Performance on dataset medical with different p.

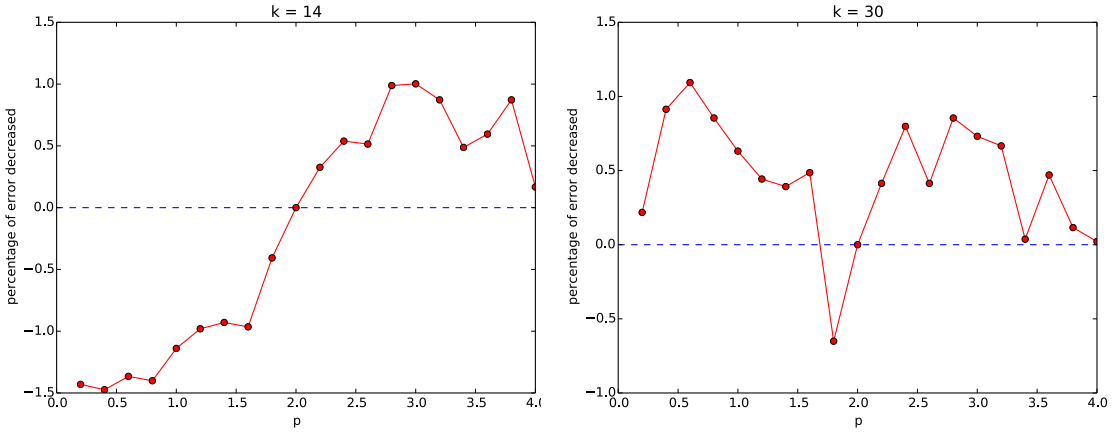


(a) k = 4



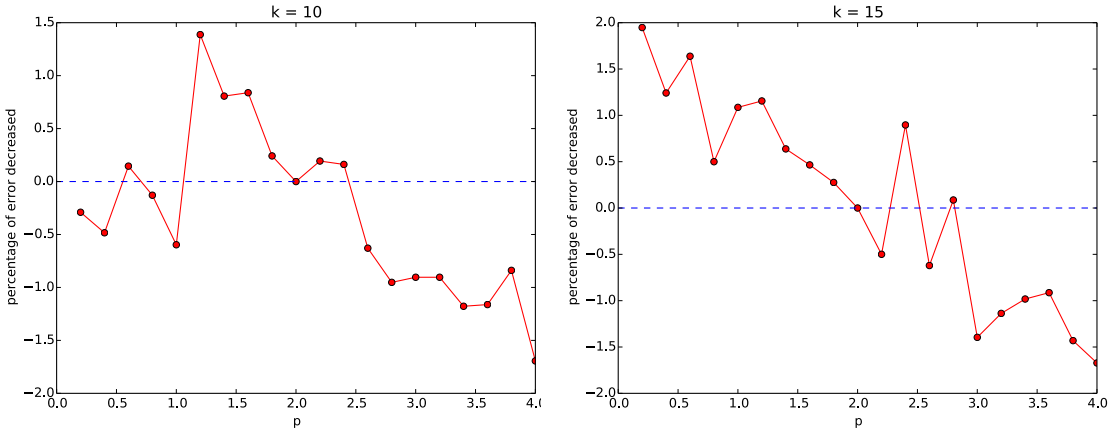
(b) k = 8

Figure 8.5: Performance on dataset emotions with different p.



(a) k = 14 (b) k = 30

Figure 8.6: Performance on dataset enron with different p.



(a) k = 10 (b) k = 15

Figure 8.7: Performance on dataset scene with different p.

work described here can be extended in several ways. It is interesting to identify other unitary invariant functions that allow new application of the greedy method. It is also interesting to study the theoretical guarantee for the greedy method under different error criteria.

CHAPTER 9

CLEANING THE NULL SPACE: A PRIVACY MECHANISM FOR PREDICTORS

In standard machine learning and regression setting feature values are used to predict some desired information. The privacy challenge considered here is to prevent an adversary from using available feature values to predict confidential information that one wishes to keep secret. We show that this can sometimes be achieved with almost no effect on the quality of predicting desired information. We describe two algorithms aimed at providing such privacy when the predictors have a linear operator in the first stage. The desired effect can be achieved by zeroing out feature components in the approximate null space of the linear operator.

Portions of this chapter are the results of collaboration with Tongyi Cao and Swair Shah. The results are published in (Xu et al., 2017).

9.1 Introduction

Consider a company that develops technology for predicting desired information from raw data. As a trivial example the raw data may be the height, weight, and sex of a child, with the desired information being the child's age. The company may obtain the data from a client, use it to predict the desired information, and send that information back to the client.

The privacy concern we address is the potential inappropriate use of the client data to predict confidential information that the client does not wish to expose. In our trivial example such information may be the child weight (which is part of the data), or whether or not the child is obese.

It appears that the current approach in similar situations is to encrypt the raw data sent to the company so that it requires a secret password to be accessed. This, however, does not provide significantly improved privacy. Someone who works at the company and knows the password may still access the data, and an adversary may still gain access to both the encrypted data and the password.

Another possible solution is to provide the client with the software that computes the desired information, so that the raw data need not be sent to the company. This solution is sometimes inappropriate because of the following two reasons. First, the software may be too big and require special hardware. Secondly, giving the program away may allow the client to use it in cases not agreed upon by the company. Similar arguments are made in (Tramèr et al., 2016).

We propose a privacy mechanism to minimize the exposure of confidential information that the client may wish to keep private even from the company. We show how to "clean" the data before sending it to the company in such a way that allows its usage for predicting the desired information. At the same time the accuracy of predicting confidential information from the cleaned data is significantly reduced.

The main idea behind our approach can be easily understood in terms of feature selection. Clearly, it makes no sense to provide the company data that is not needed for predicting the desired information. Unfortunately, if accurate prediction of desired labels is necessary, the removal of an entire feature may not be appropriate. Even "mostly irrelevant" features may contribute "something" to the prediction. Instead, we show how to remove unnecessary feature components.

Our idea is to use knowledge about the predictor of desired information to help hide confidential information. In particular, if the predictor has a linear operator in the first stage then there is a transformation of the data that reveals feature components not needed for the prediction. Specifically, combinations of features that lie in the operator null space do not affect the prediction and need not be provided. Removing this information is what we call *cleaning*.

9.2 Problem statement

The problem we consider involves three entities: the ***cleaner***, the ***ally***, and the ***adversary***. In the example discussed earlier the cleaner is the client and the ally is the company. The cleaner has a single feature vector denoted by x. In addition to x the cleaner has knowledge of the linear operator

of the predictor used by the ally. The cleaner cleans x and produces $\tilde{x}$. We write the computation performed by the cleaner as:

$$\tilde{x} = \text{clean}(x) \tag{9.1}$$

The ally has a predictor f_d that can predict desired information, denoted by y_d, from the uncleaned data x. This can be expressed as: $y_d = f_d(x)$. The ally receives the cleaned feature vector $\tilde{x}$ from the cleaner without knowledge of whether or not it was cleaned, or how it was cleaned. The computation performed by the ally is always the same, applying the predictor to the data:

$$\tilde{y}_d = f_d(\tilde{x}) \tag{9.2}$$

The adversary attempts to predict confidential information denoted by y_c. It does not know x, but it knows everything else. In particular it knows the cleaned feature vector $\tilde{x}$, the method clean() which was used by the cleaner, and the method $f_d()$ which was used by the ally. In addition, the adversary may have training data in the form of (x^i, y_c^i), relating uncleaned feature vectors to confidential information. We assume that the training data is good enough to infer an accurate predictor of the form: $y_c = f_c(x)$. In order to discover the confidential information the adversary calculates a predictor $\tilde{f}_c$ and produces the following approximation:

$$\tilde{y}_c = \tilde{f}_c(\tilde{x}) \tag{9.3}$$

9.2.1 Evaluation criteria

We wish to obtain algorithms for "clean()" such that the vector $\tilde{x}$ produced by the cleaner satisfies:

$$\begin{aligned} f_d(\tilde{x}) = \tilde{y}_d \approx y_d, \quad & e_{\text{utility}} = |\tilde{y}_d - y_d|^2 \\ \tilde{f}_c(\tilde{x}) = \tilde{y}_c \not\approx y_c, \quad & e_{\text{privacy}} = |\tilde{y}_c - y_c|^2 \end{aligned} \tag{9.4}$$

We say that the cleaning method has *high utility* if e_{utility} is guaranteed to be small, and that it has *high privacy* if e_{privacy} is guaranteed to be large. We describe algorithms that provide a guarantee

on utility but not for the worst case privacy. Specifically, for any given ϵ and x the algorithms guarantee cleaning that satisfies:

$$e_{\text{utility}} = |\tilde{y}_d - y_d|^2 \leq \epsilon \tag{9.5}$$

In our model, the adversary knows everything except for the feature vector x. It is impossible to guarantee privacy in the worst case because of the following argument. If $y_c = y_d$ then the adversary can use the predictor f_d to predict the confidential information. Therefore, in this worst case high utility would imply low privacy. Instead of considering the worst case we attempt to maximize e_{privacy} while still satisfying the constraint on e_{utility}. We present two algorithms. The first maximizes the "expected privacy error" that will be defined later. The second attempts to learn the predictor used by the adversary and maximize the privacy protection against that particular predictor.

9.2.2 Our results

Our cleaning algorithms require that the predictor $f_d()$ starts with a linear operator. This means that it can be written as: $f_d(x) = f_1(A_d^T x)$, where A_d is a matrix. (Extensions to the nonlinear case are nontrivial.) Here are several examples of commonly used predictors that can be expressed in this way.

1. Linear regression (e.g., (Miller, 2002; Hastie et al., 2009)).

2. Many approaches to multi-label classification start with linear regression. The regression results are then followed by thresholding or other classification schemes, converting the real valued data to discrete values. See, e.g. (Zhang and Zhou, 2014; Tsoumakas et al., 2011).

3. Multi-layer neural nets start with a linear operator applied to the input data. (See, e.g., (Hastie et al., 2009)).

4. Algorithms that use linear dimensionality reduction such as PCA. See, e.g., (Jolliffe, 1986).

We describe two cleaning algorithms that work intuitively as follows. The algorithms take as input the feature vector x and the matrix A_d (used by the ally in the initial linear step). The cleaning

Table 9.1: A toy example.

(x_1, x_2)	cleaned (x_1, x_2)	y_d	y_c
(3,1)	(1,-1)	2	5
(4,2)	(1,-1)	2	8
(5,1)	(2,-2)	4	7
(6,5)	(0.5,-0.5)	1	16

of x is achieved by subtracting from it projections on the approximate null space of A_d^T. These projections are irrelevant to the prediction of the desired information. Both algorithms remove components in the exact null space. They behave differently in identifying the approximate null space.

9.2.3 A toy example

To illustrate the main ideas behind our approach consider the toy example show in Table 9.1. There are two features x_1, x_2, one desired label y_d, and one confidential label y_c. Both y_d and y_c can be calculated exactly from x_1, x_2:

$$y_d = x_1 - x_2, \quad y_c = x_1 + 2x_2$$

Clearly, the desired information y_d can be predicted exactly from x_1, x_2 with zero error. However, y_c is exposed. Here the prediction model of y_d is $A_d^T = (1, -1)$, with the vector $(1, 1)$ in its null space. Thus, the vector (x_1, x_2) can be cleaned by zeroing out projections on the direction $(1, 1)$. This amounts to subtracting the mean of x_1 and x_2 from each coordinate.

After cleaning, the same linear model can still be used to predict y_d from the cleaned features. On the other hand there is no predictor for computing y_c exactly from the cleaned features, since two different values of y_c must be inferred from two identical cleaned feature vectors (lines 1,2).

9.2.4 Relation to previous work

Adapting machine learning terminology of a learning and a testing phase, our goal is protecting the privacy of information during the testing phase. It is different from studies concerned with the privacy of training data, such as differential privacy e.g., (Dwork and Roth, 2014; Sarwate and Chaudhuri, 2013), where noise is added to blur the distinction between individual items in the training data.

There are several studies that investigate feature selection as a tool for obtaining privacy of training data. See, e.g., (Pattuk et al., 2015; Jafer et al., 2015; Banerjee and Chakravarty, 2011). The idea is not to release the entire information in the data, but only selected features.

Unlike these studies we consider the privacy of information that can be extracted from a single feature vector. Clearly, to increase privacy we can add noise to the feature vector or use fewer features. However, with no distinction between desired and and confidential information the increase in privacy would imply a decrease in accuracy.

Recent studies (Enev et al., 2012; Hamm, 2015; Whitehill and Movellan, 2012) consider a setting similar to ours. They also make the distinction between desired and confidential information. In (Hamm, 2015), the ally (data aggregator) and the user (data contributor) participate in the data filtering. The goal is to prevent the adversary from predicting the private information of user while maintaining the utility of the data. The framework proposed in (Enev et al., 2012) transforms the data in a way that the covariance between the data and the desired information is increased, while the covariance between the data and confidential information is decreased. The most important difference between our approach and these approaches is in the treatment of the predictors. They make different (or no) assumptions about the predictors, while we assume knowledge of these predictors, which yields different algorithms.

The Fractional Knapsack

1. Input: $G_1, \ldots, G_n$, nonnegative $H_1, \ldots, H_n$, ϵ.
2. Output: $\alpha_1, \ldots, \alpha_n$ that solve the following problem:
3. Maximize $\sum_j \alpha_j G_j$.
4. Subject to: $\sum_j \alpha_j H_j \leq \epsilon$,
5. $0 \leq \alpha_j \leq 1$.

Figure 9.1: The Fractional Knapsack Problem.

1. If $H_j = 0$ set $\alpha_j = 1$.
2. Sort the pairs (G_j, H_j) in decreasing order of G_j/H_j. Let (j) be the location of (G_j, H_j) in this order.
3. Compute the index t such that:
$$\sum_{(j)=1}^{t} H_{(j)} < \epsilon, \quad \sum_{j=1}^{t+1} H_{(j)} \geq \epsilon$$
4. Set $r = (\epsilon - \sum_{(j)=1}^{t} H_{(j)})/H_{(t+1)}$
5. Set $\alpha_{(j)} = \begin{cases} 1 & 1 \leq (j) \leq t \\ r & (j) = t+1 \\ 0 & \text{otherwise} \end{cases}$

Figure 9.2: Solving the Fractional Knapsack.

9.3 Optimization tool

The algorithms we propose require an optimization tools that is detailed in this section. The problem to be optimized is a variation on the classical fractional Knapsack problem (in Figure 9.1), originally discussed by Dantzig in (Dantzig, 1957). See also (Goodrich and Tamassia, 2002).

The standard algorithm for the optimal solution to this problem is shown in Figure 9.2. We need the following variation of the fractional knapsack.

Augmented objective function and equality constraint. Here both the objective function in line 3 and the equality constraint in line 4 (Figure 9.2) are replaced by:

Input: x: the feature vector to be cleaned,
A_d: the known predictor,
ϵ: the desired value of e_{utility}.

Output: the cleaned feature vector $\tilde{x}$.

1. Compute eigenvectors/eigenvalues of $B_d = A_d A_d^T$.

2. For each eigenvector/eigenvalue pair (v_j, λ_j) compute: $a_j = v_j^T x$, $\delta_j = \lambda_j a_j^2$.

3. For $j = 1, \ldots, n$, solve the Augmented Fractional Knapsack problem $G_j = 1$, $H_j = \delta_j$, ϵ to determine α_j.

4. Output $\tilde{x} = x - \sum_{j=1}^{n} \alpha_j a_j v_j$.

Figure 9.3: Algorithm 1 for cleaning a feature vector x.

3. Maximize $\sum_j \alpha_j^2 G_j$.

4. Subject to: $\sum_j \alpha_j^2 H_j \leq \epsilon$.

The only change needed in the algorithm of Figure 9.2 is in line 4.

4. Set $r = \sqrt{(\epsilon - \sum_{(j)=1}^{t} H_{(j)})/H_{(t+1)}}$

9.4 Cleaning

In this section we describe the algorithms for cleaning the feature components in the approximate null space of the linear predictor. As in the previous section we denote the feature vector by x, and the cleaned feature vector by $\tilde{x}$. To simplify the discussion and the notation we assume that the predictor f_d is linear, so that $f_d(x) = A_d^T x$.

9.4.1 Algorithm 1

The first algorithm is shown in Figure 9.3. It identifies t eigenvectors and a fraction α_{t+1} of another eigenvector that are in the approximate null space of A_d. Zeroing out the projection of the feature vector x on these eigenvectors produces the desired cleaned feature vector.

Theorem 1: Let $\tilde{x}$ be the result of cleaning x by Algorithm 1. Set y_d=$A_d^T x$, $\tilde{y}_d$=$A_d^T \tilde{x}$, and $e_{\text{utility}} = |y_d - \tilde{y}_d|^2$. Then $e_{\text{utility}} \leq \epsilon$.

Proof: With the notation of Algorithm 1:

$$y_d - \tilde{y}_d = A_d^T x - A_d^T \tilde{x} = A_d^T (x - \tilde{x}) = A_d^T \sum_{j=1}^{n} \alpha_j a_j v_j$$

Therefore:

$$\begin{aligned} e_{\text{utility}} = |y_d - \tilde{y}_d|^2 &= (\sum_{i=1}^{n} \alpha_i a_i v_i^T) B_d (\sum_{j=1}^{n} \alpha_j a_j v_j) \\ &= \sum_{i=1}^{n} \lambda_i \alpha_i^2 a_i^2 = \sum_{i=1}^{t} \lambda_i a_i^2 + \lambda_{t+1} a_{t+1}^2 r^2 \leq \epsilon \quad \blacksquare \end{aligned}$$

The last inequality follows from the promise of the Fractional Knapsack.

We proceed to analyze the privacy properties of Algorithm 1. Ideally, an algorithm should be designed with the following criterion:

$$\text{maximize} \quad \inf e_{\text{privacy}}$$

where the infimum is over all possible algorithms. While this cannot be shown in our model, we can show that Algorithm 1 maximizes the following related criterion:

$$\text{maximize} \quad \mathsf{Exp}\{e_{\text{privacy}}\} \tag{9.6}$$

where the expectation is over the probability distribution defined below. Suppose the adversary uses the *linear* model A_c to predict the confidential labels from x. Let $\mathbb{A}_{\mathcal{X}}$ be the set of all pairs that can be obtained from the pair (A_c, x) by rotation. Specifically, the pair (A_1, x_1) belongs to $\mathbb{A}_{\mathcal{X}}$ if there is an orthogonal matrix Q_1 such that $A_1 = Q_1 A_c$ and $x_1 = Q_1 x$. The probability distribution in (9.6) is the uniform distribution over the elements of $\mathbb{A}_{\mathcal{X}}$. (Multiplication by an orthogonal matrix can be viewed as a change of the coordinate system.)

Lemma 1: Let $v_1, \ldots, v_n$ be an orthogonal basis of $\mathbb{R}^n$, and let (A, x) be a random variable in $\mathbb{A}_{\mathcal{X}}$. Then:

$$\mathsf{Exp}\left\{x^T v_i v_i^T A A^T v_j v_j^T x\right\} \text{ is independent of } i, j,$$

where the expectation is with respect to the uniform distribution defined over the elements of $\mathbb{A}_{\mathcal{X}}$.

Proof: For any orthogonal matrix Q we have:

$$\begin{aligned}
&\mathsf{Exp}\left\{x^T v_i v_i^T A A^T v_j v_j^T x\right\} \\
&= \int x^T v_i v_i^T A A^T v_j v_j^T x \; \mathsf{prob}((A,x)) d((A,x)) \\
&= \int v_j^T x x^T v_i v_i^T A A^T v_j \; \mathsf{prob}((A,x)) d((A,x)) \\
&= \int v_j^T Q x x^T Q^T v_i v_i^T Q A A^T Q^T v_j \; \mathsf{prob}((QA,Qx)) d((QA,Qx)) \\
&= \int v_j^T Q x x^T Q^T v_i v_i^T Q A A^T Q^T v_j \; \mathsf{prob}((A,x)) d((A,x)) \\
&= \mathsf{Exp}\left\{v_j^T Q x x^T Q^T v_i v_i^T Q A A^T Q^T v_j\right\} \\
&= \mathsf{Exp}\left\{x^T Q^T v_i v_i^T Q A A^T Q^T v_j v_j^T Q x\right\} \qquad (9.7)
\end{aligned}$$

(The integrals are computed over $\mathbb{A}_{\mathcal{X}}$.) To prove the lemma for the case where i=j we need to show that $\mathsf{Exp}\left\{x^T v_i v_i^T A A^T v_i v_i^T x\right\} = \mathsf{Exp}\left\{x^T v_k v_k^T A A^T v_k v_k^T x\right\}$ for $1 \leq k \leq n$. This follows immediately by applying Equation (9.7) with any orthonormal matrix Q satisfying $v_k = Q^T v_i$.

For the case where $i \neq j$ we need to show that $\mathsf{Exp}\left\{x^T v_i v_i^T A A^T v_j v_j^T x\right\} = \mathsf{Exp}\left\{x^T v_k v_k^T A A^T v_l v_l^T x\right\}$ for $1 \leq k, l \leq n$, $k \neq l$. This follows immediately by applying Equation (9.7) with any orthonormal matrix Q satisfying $v_k = Q^T v_i$, $v_l = Q^T v_j$. Such orthogonal matrix Q always exists since there is always a "change-of-basis" orthogonal matrix that maps one basis of $\mathbb{R}^n$ to another. ■

Theorem 2: Suppose the predictor of confidential information from x is linear, represented by the matrix A_c. Let $\mathbb{A}_{\mathcal{X}}$ be the set of all rotations of the pair (A_c, x), and consider a uniform probability distribution over the elements of $\mathbb{A}_{\mathcal{X}}$. Then the cleaned vector produced by Algorithm 1 maximizes the expected privacy error.

Proof: As in the proof of Theorem 1 we have:

$$\begin{aligned} e_{\text{privacy}} &= |y_c - \tilde{y}_c|^2 \\ &= x^T(\sum_{i=1}^{n} \alpha_i v_i v_i^T) A_c A_c^T (\sum_{j=1}^{n} \alpha_j v_j v_j^T) x \\ &= \sum_{ij} \alpha_i \alpha_j x^T v_i v_i^T A_c A_c^T v_j v_j^T x \end{aligned}$$

Taking expectations gives:

$$\mathsf{Exp}\left\{e_{\text{privacy}}\right\} = b \sum_{ij} \alpha_i \alpha_j = b(\sum_{j} \alpha_j)^2$$

where $b = \mathsf{Exp}\left\{x^T v_i v_i^T A A^T v_j v_j^T x\right\}$ which according to Lemma 1 is independent of i, j. Therefore, maximizing $\mathsf{Exp}\left\{e_{\text{privacy}}\right\}$ in terms of the α_j requires solving the following optimization problem:

$$\text{Maximize } \sum_j \alpha_j \quad \text{s.t. } 0 \leq \alpha_j \leq 1, \ \sum_j \alpha_j^2 \delta_j \leq \epsilon$$

And this can be solved as the Augmented Fractional Knapsack. ■

9.4.2 Algorithm 2

The second algorithm is shown in Figure 9.4. The cleaner uses its own training data to estimate a linear predictor A_c for the confidential labels. It then proceeds similarly to Algorithm 1, but the approximate null space is computed in terms of generalized eigenvectors and eigenvalues.

Theorem 3: Let $\tilde{x}$ be the result of cleaning x by Algorithm 2. Set y_d=$A_d^T x$, $\tilde{y}_d$=$A_d^T \tilde{x}$, and $e_{\text{utility}} = |y_d - \tilde{y}_d|^2$. Then $e_{\text{utility}} \leq \epsilon$.

Proof: Identical to the proof of Theorem 1. ■

We proceed to analyze the privacy properties of Algorithm 2.

Input: x: the feature vector to be cleaned,
A_d: the known predictor,
ϵ: the desired value of e_{utility},
training data: (x^i, y_c^i).

Output: the cleaned feature vector $\tilde{x}$.

1. Use the training data to compute the predictor A_c so that $y_c^i \approx A_c^T x^i$.

2. For $B_d = A_d A_d^T$, $B_c = A_c A_c^T$, solve the generalized eigenvalue problem: $B_d v = \gamma B_c v$. The result is $(v_1, \gamma_1), \ldots, (v_n, \gamma_n)$.

3. For $j = 1, \ldots, n$: compute $a_j = v_j^T x$, λ_j^d=$v_j^T B_d v_j$, δ_j^d=$\lambda_j^d a_j^2$, λ_j^c=$v_j^T B_c v_j$, δ_j^c=$\lambda_j^c a_j^2$.

4. For $j = 1, \ldots, n$: solve the Augmented Fractional Knapsack for $G_j = \delta_j^c$, $H_j = \delta_j^d$, ϵ to determine α_j.

5. Output $\tilde{x} = x - \sum_{j=1}^n \alpha_j a_j v_j$.

Figure 9.4: Algorithm 2 for cleaning a feature vector x.

Theorem 4: If the Adversary uses the linear predictor A_c as computed by the cleaner then the cleaned vector produced by Algorithm 1 maximizes the privacy error.

Proof: The generalized eigenvalues γ_j satisfy:

$$\gamma_j = \lambda_j^d / \lambda_j^c, \quad \text{where } \lambda_j^d = v_j^T B_d v_j,\ \lambda_j^c = v_j^T B_c v_j$$

Following the same derivation as in Theorem 1 we have:

$$e_{\text{utility}} = \sum_{j=1}^n \alpha_j^2 \delta_j^d$$

$$e_{\text{privacy}} = \sum_{j=1}^n \alpha_j^2 \delta_j^c$$

Therefore, the optimization problem that needs to be solved to maximize the privacy error is:

$$\text{Maximize } \sum_{j=1}^n \alpha_j^2 \delta_j^c$$

$$\text{s.t. } 0 \le \alpha_j \le 1,\ \sum_j \alpha_j^2 \delta_j^d \le \epsilon$$

And this problem is solved by Algorithm 2 using the Augmented Fractional Knapsack. ■

Table 9.2: Dataset description.

Dataset	Instances(m)	Features(n)	Labels(N)
scene	2407	294	6
wq	1060	16	14
oes97	334	263	16
CAL500	502	68	174

9.5 Experimental results

We evaluated the proposed algorithms on benchmark datasets from Mulan repository (Tsoumakas et al., 2011) shown in Table 9.2. Experimental results are averaged over 10 runs. In each run 90% of the dataset is chosen randomly to compute the model of the ally. The rest is used as testing data, where each individual sample is evaluated independently of the other samples. We ran three experiments with each dataset, varying the number of desired and confidential labels:

1) one desired label and all the rest are confidential labels.

2) randomly select half of the labels as desired and the other half as confidential.

3) one confidential and all the rest are desired labels.

In the tables N_d and N_c denote the number of desired and confidential labels respectively. The results are shown in Table 9.3. Observe that in almost all cases e_{utility} is exactly ϵ and e_{privacy} is much bigger than e_{utility}. The special case happens on dataset CAL500 when N_d is 1 and N_c is 173.

To provide further insight into the performance of the cleaning mechanism we count the test cases for which the cleaning achieves "linear privacy". The criterion for ***linear privacy*** is defined as follows.

> If the error of predicting y_c using $\tilde{x}$ is greater than the error of predicting y_c from the mean feature vector, then ***linear privacy*** has been achieved.

From the definition of *linear privacy*, if the adversary takes random guess, then the probability that the prediction is better than the prediction computed from the mean feature vector is 0.5. Therefore, we consider 50% is the best possible *linear privacy*. In Table 9.3, LP denotes the percentage of test cases for which the cleaner achieves *linear privacy*. We observe that in most of the cases the

Table 9.3: Results with no attack, $\epsilon = 0.01$, Almost all the cases e_{utility} is exactly ϵ, and e_{privacy} is much bigger than e_{utility}. LP denotes the percentage of test samples which achieves *linear privacy*.

Dataset	N_d	N_c	Algorithm 1			Algorithm 2		
			e_{utility}	e_{privacy}	LP	e_{utility}	e_{privacy}	LP
scene	1	5	0.01	0.796	81.8%	0.01	0.796	81.8%
	3	3	0.01	1.358	80.6%	0.01	1.296	80.7%
	5	1	0.01	4.259	82.3%	0.01	3.659	82.3%
CAL500	1	173	0.009	0.845	85.7%	0.01	5.3E+10	100.0%
	87	87	0.01	0.156	63.5%	0.01	1.741	79.9%
	173	1	0.01	0.034	46.0%	0.01	2.188	59.6%
wq	1	13	0.01	2.654	52.6%	0.01	2.654	52.6%
	7	7	0.01	2.172	58.8%	0.01	1.956	57.8%
	13	1	0.01	2.265	74.8%	0.01	1.403	45.5%
oes97	1	15	0.01	5.11E+7	65.4%	0.01	5.11E+7	65.4%
	8	8	0.01	4.23E+7	63.0%	0.01	4.23E+7	63.0%
	15	1	0.01	6.02E+7	47.0%	0.01	6.02E+7	47.0%

adversary would be no more advantageous using the cleaned feature vector $\tilde{x}$ than using the mean feature vector.

In some cases, Algorithm 1 and Algorithm 2 have the same performance (e.g. dataset scene and oes97). In other cases (e.g. dataset CAL500 and wq), the two algorithms perform differently. It is because both the algorithms will clean feature vector x with the eigenvectors/generalized eigenvectors in the nullspace first. These vectors are identical to the two algorithms. After such cleaning e_{utility} maintains 0. To achieve the desired value of e_{utility}, both the algorithms will keep cleaning x with other vectors/generalized eigenvectors. This is the procedure where Algorithm 1 and Algorithm 2 diverges. Algorithm 2 selects vectors obtained by taking into consideration the predictor model A_c for confidential information, while Algorithm 1 selects vectors obtained without considering such information.

It is possible that the adversary uses a model other than A_c which may be better suited for predicting confidential labels. We discuss such a scenario next.

Input: x^t: a test example,
training data: (x^i, y_c^i).

Output: Attempted prediction of y_c^t from x^t.

1. Repeat for each feature vector x^i:
Apply the cleaning algorithm to get a cleaned feature vector $\tilde{x}^i$.

2. Use the cleaned training data to build a prediction model.

3. Use the model created in step 2 to predict y_c^t from x^t.

Figure 9.5: A proposed attack on the cleaning algorithms.

9.5.1 A proposed attack

We assume the adversary knows the cleaning algorithms. Therefore, the adversary can acquire a dataset and obtain cleaned features. With the information of confidential labels and features after cleaning, a new model can be inferred for predicting y_c. Given a test example, the adversary would apply the new model to make prediction. The procedure is shown in Figure 9.5.

The experiments with the proposed attack show that e_{utility} is still exactly ϵ for both algorithms in almost all the cases. Due to the limitation of table size, here e_{utility} is not shown. Other results are shown in Table 9.4. e_{privacy} is still considerably higher than e_{utility}. In some cases the rate of linear privacy is 40%. That shows the cleaned feature vector provides no more confidential information than the mean feature vector for 40% of the test samples.

It is to be noted that for some datasets e_{privacy} becomes extremely high (e.g. oes97) and for some datasets e_{privacy} is relatively small. This is dependent on the dataset as well as the desired and confidential labels. A higher privacy (at the expense of utility) can be obtained by increasing the value of ϵ.

9.5.2 Experiments with non-linear SVM

The adversary may use predictors other than the linear operators. We discuss the case when the adversary uses non-linear SVM as the prediction model. In particular, we show the results that

Table 9.4: Results with the proposed attack. $\epsilon = 0.01$. LP is the percentage of test cases that achieves *linear privacy*. Still e_{privacy} is bigger than e_{utility} in most of the cases.

N_d	N_c	Algorithm 1		Algorithm 2	
		e_{privacy}	LP	e_{privacy}	LP
scene dataset (n = 294)					
1	5	0.289	51.0%	0.289	51.0%
3	3	0.235	44.7%	0.233	44.5%
5	1	0.047	41.0%	0.043	41.8%
CAL500 dataset (n = 68)					
1	173	0.230	48.6%	0.005	58.9%
87	87	0.081	56.9%	0.008	55.9%
173	1	0.068	44.0%	0.029	44.4%
wq dataset (n = 16)					
1	13	0.308	48.9%	0.308	48.9%
7	7	0.132	39.2%	0.164	38.7%
13	1	0.092	31.9%	0.086	30.9%
oes97 dataset (n = 263)					
1	15	4.67E+6	39.4%	4.67E+6	39.4%
8	8	7.48E+6	43.6%	7.48E+6	43.6%
15	1	1.16E+7	18.2%	1.16E+7	18.2%

predicting confidential information via the SVM model with RBF kernel. The feature vectors are cleaned under the setting $\epsilon = 0.01$ for both Algorithm 1 and Algorithm 2. The experimental results are shown in Table 9.5. Observe that e_{privacy} is considerably higher then e_{utility} and around 50% of the test samples achieve LP.

Table 9.6 shows the results with non-linear SVM model under the proposed attack. Instead of building a linear model at the step 2 of the attack algorithm shown in Figure 9.5, a non-linear SVM prediction model is built using the cleaned training data. The overall LP is around 30%.

9.5.3 Comparison with a differential privacy mechanism

The goal of differential privacy is protecting the individual entry of a database while allowing accurate statistical analysis of the entire database. A standard way to achieve differential privacy is the Laplace Noise (Dwork and Roth, 2014). Carefully adding noise drawn from a Laplace

Table 9.5: Results with non-linear SVM model. LP is the percentage of test cases that achieves *linear privacy*.

N_d	N_c	Algorithm 1		Algorithm 2	
		e_{privacy}	LP	e_{privacy}	LP
scene dataset (n = 294)					
1	5	0.229	42.0%	0.229	42.0%
3	3	0.271	45.0%	0.267	45.0%
5	1	0.255	46.5%	0.249	46.3%
CAL500 dataset (n = 68)					
1	173	0.233	52.3%	0.059	52.2%
87	87	0.132	46.4%	0.071	48.7%
173	1	0.124	46.1%	0.238	49.4%
wq dataset (n = 16)					
1	13	1.048	41.3%	1.048	41.3%
7	7	1.159	43.2%	1.159	43.2%
13	1	2.305	52.4%	2.305	52.4%
oes97 dataset (n = 263)					
1	15	159.7	38.3%	159.7	38.3%
8	8	155.9	38.5%	155.9	38.5%
15	1	185.5	36.3%	185.5	36.3%

distribution to the features would achieve differential privacy but at the cost of utility. On the other hand if one adds noise to the feature vector while attempting to keep e_{utility} small, then e_{privacy} would also remain very small, thus privacy is sacrificed.

In the experiments, we adjust the Laplace noise parameter *mean* and *scale* to produce the same e_{utility} as Algorithm 1. The results are shown in Table 9.7. Observe that even with the proposed attack our approach achieves better *linear privacy* compared to the Laplace noise. Thus for the same value of utility our approach provides much better privacy.

9.6 Discussion

The problem considered here is protecting the privacy of predictors. It is different from studies concerned with the privacy of the training data, such as differential privacy. Recent studies about balancing utility and privacy propose similar models with different (or no) assumptions about the

Table 9.6: Results with non-linear SVM model under the proposed attack. LP is the percentage of test cases that achieves *linear privacy*.

N_d	N_c	Algorithm 1		Algorithm 2	
		e_{privacy}	LP	e_{privacy}	LP
scene dataset (n = 294)					
1	5	0.189	18.2%	0.189	18.2%
3	3	0.193	19.0%	0.193	19.0%
5	1	0.167	22.0%	0.167	22.4%
CAL500 dataset (n = 68)					
1	173	0.065	49.9%	0.050	49.3%
87	87	0.050	44.5%	0.025	44.3%
173	1	0.074	43.5%	0.017	43.4%
wq dataset (n = 16)					
1	13	0.332	33.3%	0.332	33.3%
7	7	0.257	33.6%	0.246	33.5%
13	1	0.254	33.2%	0.229	32.9%
oes97 dataset (n = 263)					
1	15	179.8	36.0%	179.8	36.0%
8	8	113.2	35.9%	113.2	35.9%
15	1	91.98	34.4%	91.98	34.4%

Table 9.7: Comparison with the Laplace Noise. LP is the percentage of test cases that achieves *linear privacy*. For dataset scene, N_d is 1, N_c is 5. For dataset oes97, N_d is 8, N_c is 8. Our approach provides much better privacy for the same value of utility.

Dataset	e_{utility}	Laplace Noise		Alg 1 with no attack		Alg 1 with attack	
		e_{privacy}	LP	e_{privacy}	LP	e_{privacy}	LP
scene	0.01	0.009	31.5%	0.796	81.8%	0.289	51.0%
	0.02	0.019	31.4%	0.772	81.8%	0.289	50.3%
	0.03	0.027	31.3%	0.770	81.8%	0.289	51.1%
oes97	0.01	0.015	24.6%	4.23E+7	63.0%	7.48E+6	43.6%
	0.10	0.162	24.6%	4.23E+7	63.0%	7.48E+6	43.6%
	1.00	1.698	24.6%	4.23E+7	63.0%	7.48E+6	43.5%

predictors. By contrast, we assume partial knowledge of these predictors, which may provide sharper results.

The proposed algorithms can be applied to models that start with a linear operator. We observe that projections on the null space of the predictors do not change the prediction value. These projections give out unnecessary information for the prediction. Therefore, cleaning the data by zeroing out projections on the null-space will not affect the prediction accuracy but will increase the privacy. The more distortion in feature vectors, the higher privacy and the lower utility. The trade-off between utility and privacy can be controlled by adjusting the ϵ parameter.

CHAPTER 10

CONCLUSIONS

The work presented in this dissertation addressed two important problems in data analytics: feature selection and data privacy. The algorithms developed in Chapter 4 address the classical column subset selection problems with improved accuracy when compared to the current state of the art. The algorithms run slower than some numerical linear algebra/randomized techniques, but are still practical for large datasets. It should be a preferred choice when an increase in running time can be tolerated.

The algorithms shown in Chapter 5 compute an optimal solution for column subset selection under unitarily invariant criteria that are much faster than exhaustive search. One version of the algorithms is optimal and the other gives exact bounds on optimality. While there are many studies that address speed and accuracy for the column subset selection problem, few come with a guarantee on optimality. There are currently no alternatives to the algorithms described in Chapter 5 when one is interested in obtaining the best possible selection in anything but the Frobenius norm.

Chapter 6 describes an approach that performs feature selection in two stages where the second stage is unsupervised. It was not previously known that it is possible to use unsupervised algorithms in the second stage in a general setting. We compared the results to a typical setting where the second stage is implemented with a supervised algorithm. Our method is almost as accurate as using supervised algorithms, while at the same time it may be significantly faster.

The greedy approaches discussed in Chapter 7 can effectively select features for multi-label classification. Many studies achieve feature selection by regularization and the solutions are obtained iteratively until convergence. By contrast, our method is numerically stable and efficient. The accuracy of our algorithms is comparable with current state of the art. The generalization of the error criteria to unitary invariant functions for a classic greedy approach is described in Chapter 8, where the new algorithm is used to solve the outlier-robust PCA problem. The solution is

novel and is very different from previous studies. Experimental results show that our algorithm outperforms a well-known outlier-robust PCA method that uses convex optimization.

Chapter 9 describes algorithms that protect the privacy of predictors that start with a linear operator. Our model is different from studies concerned with the privacy of the dataset, such as differential privacy. Recent studies about balancing utility and privacy propose similar models with different (or no) assumptions about the predictors. By contrast, we assume partial knowledge of these predictors, which may provide sharper results. In a nutshell, the projections on the null space of the predictors give out information that is unnecessary for the prediction. Therefore, cleaning the data by zeroing out projections on the null-space will not affect the prediction accuracy but will increase the privacy. The more distortion in feature vectors, the higher privacy and the lower utility. The trade-off between utility and privacy can be controlled by adjusting a user defined parameter.

REFERENCES

Altunay, S., Z. Telatar, and O. Erogul (2010). Epileptic eeg detection using the linear prediction error energy. *Expert Systems with Applications 37*(8), 5661–5665.

Arai, H., C. Maung, and H. Schweitzer (2015). Optimal column subset selection by A-Star search. In *Proceedings of the 29th National Conference on Artificial Intelligence (AAAI'15)*, pp. 1079–1085. AAAI Press.

Arai, H., C. Maung, K. Xu, and H. Schweitzer (2016). Unsupervised feature selection by heuristic search with provable bounds on suboptimality. In *Proceedings of the 30th National Conference on Artificial Intelligence (AAAI'16)*, pp. 666–672. AAAI Press.

Banerjee, M. and S. Chakravarty (2011). Privacy preserving feature selection for distributed data using virtual dimension. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, New York, NY, USA, pp. 2281–2284. ACM.

Bergeaud, F. and S. Mallat (1995). Matching pursuit of images. In *International Conference on Image Processing)*, Volume 1, pp. 53–56.

Bini, D. A. and L. Robol (2014). Solving secular and polynomial equations: A multiprecision algorithm. *Journal of Computational and Applied Mathematics 272*, 276–292.

Boutell, M. R., J. Luo, X. Shen, and C. M. Brown (2004). Learning multi-label scene classification. *Pattern recognition 37*(9), 1757–1771.

Boutsidis, C., P. Drineas, and M. Magdon-Ismail (2013). Near-optimal coresets for least-squares regression. *IEEE Transactions on Information Theory 59*(10), 6880 – 6892.

Boutsidis, C., M. W. Mahoney, and P. Drineas (2009). An improved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pp. 968–977. Society for Industrial and Applied Mathematics.

Businger, P. and G. H. Golub (1965). Linear least squares solutions by Householder transformations. *Numer. Math. 7*, 269–276.

Cabral, R., F. D. L. Torre, J. P. Costeira, and A. Bernardino (2013). Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In *2013 IEEE International Conference on Computer Vision*, pp. 2488–2495.

Candès, E. J., X. Li, Y. Ma, and J. Wright (2011). Robust principal component analysis? *Journal of the ACM 58*(3), 11:1–11:37.

Candès, E. J. and T. Tao (2010). The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory 56*(5), 2053–2080.

Carney, P. R., S. Myers, and J. D. Geyer (2011). Seizure prediction: methods. *Epilepsy & behavior 22*, S94–S101.

Çivril, A. (2014). Column subset selection problem is ug-hard. *Journal of Computer and System Sciences 80*(4), 849–859.

Çivril, A. and M. Magdon-Ismail (2009). On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science 410*(47-49), 4801–4811.

Çivril, A. and M. Magdon-Ismail (2012). Column subset selection via sparse approximation of SVD. *Theoretical Computer Science 421*, 1–14.

Chandrashekar, G. and F. Sahin (2014). A survey on feature selection methods. *Computers & Electrical Engineering 40*(1), 16–28.

Chang, X., F. Nie, Y. Yang, and H. Huang (2014). A convex formulation for semi-supervised multi-label feature selection. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1171–1177. AAAI Press.

Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation 19*, 1155–1178.

Chen, J. and X. Huo (2006). Theoretical results of sparse representations of multiple measurement vectors. *IEEE Transactions on Signal processing 54*(12), 4634–4643.

Chen, S., S. A. Billings, and W. Luo (1989). Orthogonal least squares methods and their application to non-linear system identification. *International Journal of control 50*(5), 1873–1896.

Chen, Y., H. Xu, C. Caramanis, and S. Sanghavi (2011). Robust matrix completion and corrupted columns. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 873–880.

Çivril, A. (2017). Sparse approximation is provably hard under coherent dictionaries. *Journal of Computer and System Sciences 84*, 32–43.

Coope, I. D. and P. F. Renaud (2009). Trace inequalities with applications to orthogonal regression and matrix nearness problems. *Journal of Inequalities in Pure and Applied Mathematics 10*(4).

Cortés, J., G. E. Dullerud, S. Han, J. Le Ny, S. Mitra, and G. J. Pappas (2016). Differential privacy in control and network systems. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pp. 4252–4272. IEEE.

Cotter, S. F., B. D. Rao, K. Engen, and K. Kreutz-Delgado (2005). Sparse solutions to linear inverse problems with multiple measurement vectors. *ASP 53*(7), 2477–2488.

Dahmen, A. R. B. A. C. W. and R. A. DeVore (2008). Approximation and learning by greedy algorithms. *Annals of Statistics 36*(1), 64–94.

Dantzig, G. B. (1957). Discrete-variable extremum problems. *Operations Research 5*, 266–277.

Das, A. and D. Kempe (2008). Algorithms for subset selection in linear regression. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 45–54.

Das, A. and D. Kempe (2011). Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proceedings of the 28th International Conference on Machine Learning, ICML'11*, pp. 1057–1064.

Dasgupta, A., P. Drineas, B. Harb, V. Josifovski, and M. W. Mahoney (2007). Feature selection methods for text classification. In P. Berkhin, R. Caruana, and X. Wu (Eds.), *KDD*, pp. 230–239. ACM.

Davis, G., S. Mallat, and Z. Zhang (1994). Adaptive time-frequency decompositions with matching pursuit. *Optical Engineering 33*(7).

Demuth, H. B., M. H. Beale, O. De Jess, and M. T. Hagan (2014). *Neural network design*. Martin Hagan.

Deshpande, A. and L. Rademacher (2010). Efficient volume sampling for row/column subset selection. In *Foundations of Computer Science (FOCS)*, pp. 329–338. IEEE.

Deshpande, A., L. Rademacher, S. Vempala, and G. Wang (2006). Matrix approximation and projective clustering via volume sampling. *Theory of Computing 2*(12), 225–247.

Diplaris, S., G. Tsoumakas, P. A. Mitkas, and I. Vlahavas (2005). Protein classification with multiple algorithms. In *Panhellenic Conference on Informatics*, pp. 448–456. Springer.

Dong, J., B. Cheng, X. Chen, T.-S. Chua, S. Yan, and X. Zhou (2013). Robust image annotation via simultaneous feature and sample outlier pursuit. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 9*(4), 24.

Drineas, P., J. Lewis, and P. Paschou (2010). Inferring geographic coordinates of origin for europeans using small panels of ancestry informative markers. *PLoS ONE 5*(8), e11892. doi:10.1371/journal.pone.0011892.

Duda, R. O., P. E. Hart, and D. G. Stork (2001). *Pattern Classification* (Second ed.). John Wiley & Sons.

Dwork, C. and A. Roth (2014). The algorithmic foundations of differential privacy. *Foundions and Trends in Theoretical Computer Science 9*(3 & 4), 211–407.

Dy, J. G., C. E. Brodley, A. Kak, L. S. Broderick, and A. M. Aisen (2003, March). Unsupervised feature selection applied to content-based retrieval of lung images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*(3), 373–378.

Džeroski, S., D. Demšar, and J. Grbović (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence 13*(1), 7–17.

Enev, M., J. Jung, L. Bo, X. Ren, and T. Kohno (2012). Sensorsift: balancing sensor data privacy and utility in automated face understanding. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pp. 149–158. ACM.

Fan, J. and J. Lv (2008). Sure independence screening for ultra-high dimensional feature space. *Journal of Royal Statistical Society B 70*, 849–911.

Fan, J. and J. Lv (2010). A selective overview of variable selection in high dimensional feature space. *Statistica Sinica 20*, 101–148.

Frieze, A. M., R. Kannan, and S. Vempala (2004). Fast Monte-Carlo algorithms for finding low-rank approximations. *Journal of the ACM 51*(6), 1025–1041.

Gharavi-Alkhansari, M. and T. S. Huang (1998). A fast orthogonal matching pursuit algorithm. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, Volume 3, pp. 1389–1392 vol.3.

Goldfarb, D. and A. U. Idnani (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming 27*(1), 1–33.

Golub, G. H. (1973). Some modified matrix eigenvalue problems. *SIAM Review 15*(2), 318–334.

Golub, G. H. and C. F. Van-Loan (2013). *Matrix Computations* (Fourth ed.). The Johns Hopkins University Press.

Goodrich, M. T. and R. Tamassia (2002). *Algorithm Design: Foundations, Analysis, and Internet Examples*. John Wiley & Sons.

Gu, M. and S. C. Eisenstat (1996). Efficient algorithms for computing a strong rank-revealing qr factorization. *SIAM Journal on Scientific Computing 17*(4), 848–869.

Guruswami, V. and A. K. Sinop (2012). Optimal column-based low-rank matrix reconstruction. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pp. 1207–1214. SIAM.

Guyon, I. and A. Elisseeff (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research 3*, 1157–1182.

Guyon, I., S. Gunn, S. B. Hur, and G. Dror (2004). Result analysis of the nips2003 feature selection challenge. In *Advances in Neural Information Processing Systems 17*, pp. 545–553.

Guyon, I., J. Weston, S. Barnhill, and V. Vapnik (2002). Gene selection for cancer classification using support vector machines. *Machine Learning 46*(1-3), 389–422.

Halko, N., P. G. Martinsson, and J. A. Tropp (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review 53*(2), 217–288.

Hamm, J. (2015). Preserving privacy of continuous high-dimensional data with minimax filters. In *Artificial Intelligence and Statistics*, pp. 324–332.

Hardy, G., J. E. Littlewood, and G. Polya (1952). *Inequalities* (Second ed.). Cambridge University Press.

Hardy, Q. (2016). Looking beyond the internet of things. *New York Times 2*.

Hart, P. E., N. Nilsson, and B. Raphael (1968). A formal basis for the heurisitc determination of minimal cost paths. *IEEE Transactions on Systems Science and Cybernetics 4*(2), 100–107.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning* (second ed.). Springer.

Jafer, Y., S. Matwin, and M. Sokolova (2015). A framework for a privacy-aware feature selection evaluation measure. In *13th Annual Conference on Privacy, Security and Trust (PST)*, pp. 62–69.

Javed, K., H. Babri, and M. Saeed (2012). Feature selection based on class-dependent densities for high-dimensional binary data. *IEEE Transactions on Knowledge and Data Engineering 24*(3), 465–477.

Jian, L., J. Li, K. Shu, and H. Liu (2016). Multi-label informed feature selection. In *25th International Joint Conference on Artificial Intelligence*, pp. 1627–1633.

Jimenez-Rodriguez, L. O., E. Arzuaga-Cruz, and M. Velez-Reyes (2007). Unsupervised linear feature-extraction methods and their effects in the classification of high-dimensional data. *IEEE Transactions on Geoscience and Remote Sensing 45*(2), 469–483.

Johannesen, J. K., J. Bi, R. Jiang, J. G. Kenney, and C.-M. A. Chen (2016). Machine learning identification of eeg features predicting working memory performance in schizophrenia and healthy adults. *Neuropsychiatric electrophysiology 2*(1), 3.

Jolliffe, I. T. (1986). *Principal Component Analysis*. Springer-Verlag.

Katakis, I., G. Tsoumakas, and I. Vlahavas (2008). Multilabel text classification for automated tag suggestion. *ECML PKDD discovery challenge 75*.

Kerschbaum, F. and J. Strüker (2012). J.: From a barrier to a bridge: data-privacy in deregulated smart grids. In *Thirty Third International Journal Conference on Information Systems, Orlando USA*.

Koufogiannis, F. and G. J. Pappas (2016). Location-dependent privacy. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pp. 7586–7591. IEEE.

Kumar, S., M. Mohri, and A. Talwalkar (2012). Sampling methods for the nystrom method. *Journal of Machine Learning Research 13*, 981–1006.

Li, Z., Y. Yang, J. Liu, X. Zhou, and H. Lu (2012). Unsupervised feature selection using nonnegative spectral analysis. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press.

Liao, B., Y. Jiang, W. Liang, W. Zhu, L. Cai, and Z. Cao (2014, Nov). Gene selection using locality sensitive laplacian score. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 11*(6), 1146–1156.

Lichman, M. (2013). UCI machine learning repository.

Likhachev, M., G. J.Gordon, and S. Thrun (2003). ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems 16 (NIPS'03)*, pp. 767–774.

Lu, Z. and Y. Zhang (2015). Schatten-p quasi-norm regularized matrix optimization via iterative reweighted singular value minimization. *arXiv preprint arXiv:1401.0869*.

Mahoney, M. W. and P. Drineas (2009). CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences 106*(3), 697–702.

Mallat, S. (1999). *A wavelet Tour of Signal Processing*. Academic Press.

Mallat, S. and Z. Zhang (1993). Matching pursuits with time-frequency dictionaries. *Trans. Sig. Proc. 41*(12), 3397–3415.

Mao, K. Z. (2004). Orthogonal forward selection and backward elimination algorithms for feature subset selection. *IEEE Transactions on Systems, Man, and Cybernetics 34*(1), 629–634.

Marjanovic, G. and V. Solo (2012). On l_q optimization and matrix completion. *IEEE Transactions on signal processing 60*(11), 5714–5724.

Marshall, A. W., I. Olkin, and B. C. Arnold (2011). *Inequalities: Theory of Majorization and Its Applications* (Second ed.). Springer.

Mathar, R. and R. Meyer (1993). Preorderings, monotone functions, and best rank r approximations with applications to classical MDS. *Journal of Statistical Planning and Inference 37*, 291–305.

Maung, C. and H. Schweitzer (2013). Pass-efficient unsupervised feature selection. In *Advances in Neural Information Processing Systems (NIPS)*, Volume 26, pp. 1628–1636.

Maung, C. and H. Schweitzer (2015, March). Improved greedy algorithms for sparse approximation of a matrix in terms of another matrix. *IEEE Transactions on Knowledge and Data Engineering 27*(3), 769–780.

McDaniel, P. and S. McLaughlin (2009). Security and privacy challenges in the smart grid. *IEEE Security & Privacy 7*(3).

Melman, A. (1998, January). Analysis of third-order methods for secular equations. *Mathematics of Computation 67*(221), 271–286.

Miller, A. (2002). *Subset Selection in Regression* (Second ed.). Chapman & Hall/CRC.

Muthukrishnan, S. (2011). Theory of data stream computing: Where to go. In *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '11, New York, NY, USA, pp. 317–319. ACM.

Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM Journal of Computing 25*(2), 227–234.

Neff, R. and A. Zakhor (1997). Very low bit-rate video coding based on matching pursuits. *IEEE Transactions on Circuits and Systems for Video Technology 7*(1), 158–171.

Nie, F., H. Huang, X. Cai, and C. Ding (2010). Efficient and robust feature selection via joint ℓ2,1-norms minimization. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, NIPS'10, USA, pp. 1813–1821. Curran Associates Inc.

Nie, F., H. Huang, and C. Ding (2012). Low-rank matrix recovery via efficient schatten p-norm minimization. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 655–661.

Pati, Y. C., R. Rezaiifar, and P. S. Krishnaprasad (1993). Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, Volume 1, pp. 40–44.

Pattuk, E., M. Kantarcioglu, H. Ulusoy, and B. Malin (2015). Privacy-aware dynamic feature selection. In *IEEE 31st International Conference on Data Engineering (ICDE)*, pp. 78–88.

Paul, S. and P. Drineas (2016). Feature selection for ridge regression with provable guarantees. *Neural computation*.

Paul, S., M. Magdon-Ismail, and P. Drineas (2015). Column selection via adaptive sampling. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28*, pp. 406–414. Curran Associates, Inc.

Pearl, J. (1984). *Heuristics : intelligent search strategies for computer*. Reading, Massachusetts: Addison-Wesley.

Rebollo-Neira, L. and D. Lowe (2002). Optimized orthogonal matching pursuit approach. *IEEE Signal Processing Letters 9*, 137–140.

Russell, S. and P. Norvig (2010). *Artificial Intelligence - A Modern Approach*. Pearson Education.

Sarwate, A. and K. Chaudhuri (2013). Signal processing and machine learning with differential privacy: theory, algorithms, and challenges. *IEEE Signal Processing Magazine 30*(5), 86–94.

Shitov, Y. (2017). Column subset selection is NP-complete. *arXiv preprint arXiv:1701.02764*.

Soussen, C., R. Gribonval, J. Idier, and C. Herzet (2013). Joint k-step analysis of orthogonal matching pursuit and orthogonal least squares. *IEEE Transactions on Information Theory 59*(5), 3158–3174.

Spyromitros-Xioufis, E., G. Tsoumakas, W. Groves, and I. Vlahavas (2016). Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning 104*(1), 55–98.

Suykens, J. A. and J. Vandewalle (1999). Least squares support vector machine classifiers. *Neural processing letters 9*(3), 293–300.

Tang, Y., Y. Zhang, and Z. Huang (2007). Development of two-stage svm-rfe gene selection strategy for microarray expression data analysis. *IEEE/ACM Transactions onComputational Biology and Bioinformatics 4*(3), 365–381.

Tao, T. (2012). *Topics in Random Matrix Theory*, Volume 132 of *Graduate studies in mathematics*. American Mathematical Society.

Tramèr, F., F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart (2016). Stealing machine learning models via prediction apis. *CoRR abs/1609.02943*.

Tropp, J. A. (2004). Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory 50*(10), 2231–2242.

Tropp, J. A., A. C. Gilbert, and M. J. Strauss (2006). Algorithms for simultaneous sparse approximation. part I: Greedy pursuit. *Signal Processing 86*(3), 572–588.

Tsoumakas, G., E. Spyromitros-Xioufis, J. Vilcek, and I. P. Vlahavas (2011). MULAN: A Java Library for Multi-Label Learning. *Journal of Machine Learning Research 12*, 2411–2414.

Wang, C., M. Gong, M. Zhang, and Y. Chan (2015). Unsupervised hyperspectral image band selection via column subset selection. *IEEE Geoscience and Remote Sensing Letters 12*(7), 1411–1415.

Wei, H. and S. A. Billings (2007). Feature subset selection and ranking for data dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*(1), 162–166.

Whitehill, J. and J. Movellan (2012). Discriminately decreasing discriminability with learned image filters. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2488–2495. IEEE.

Xu, H., C. Caramanis, and S. Sanghavi (2010). Robust PCA via outlier pursuit. In *Advances in Neural Information Processing Systems*, pp. 2496–2504.

Xu, K., T. Cao, S. Shah, C. Maung, and H. Schweitzer (2017). Cleaning the null space: A privacy mechanism for predictors. In *Thirty-First AAAI Conference on Artificial Intelligence*, pp. 2789–2795.

Zhang, H., Z. Lin, C. Zhang, and E. Y. Chang (2015). Exact recoverability of robust pca via outlier pursuit with tight recovery bounds. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 3143–3149.

Zhang, M. and Z. Zhou (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering 26*(8).

Zhu, F., B. Fan, X. Zhu, Y. Wang, S. Xiang, and C. Pan (2015). 10,000+ times accelerated robust subset selection. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 3217–3223.

BIOGRAPHICAL SKETCH

Ke Xu was born in Wuchang, China. She went to Beihang University in 2009 and majored in computer science. After completing her college degree, she joined the Computer Science Department at The University of Texas at Dallas as a PhD student in 2013. She has been working with Dr. Haim Schweitzer since 2014. She interned at Microsoft in Beijing in the spring of 2013 and interned at Airbnb in San Francisco in the summer of 2017.

CURRICULUM VITAE

Ke Xu

800 W Campbell Rd, Richardson, TX 75080 ⋄ ke.xu5@utdallas.edu

EDUCATION

The University of Texas at Dallas — Expected Dec 2017
Ph.D. in Computer Science — *Richardson, TX*

Beihang University — Sep. 2009 - Jul. 2013
B.S. in Computer Science — *Beijing, China*

SKILLS

- Programming Languages: Python, Java, R, C++
- Framework: Airflow, Django, OpenCV, Weka

EXPERIENCES

Airbnb — May 2017 - Aug. 2017
Software Engineer Intern — *San Francisco, CA*

- Migrated the real-time prediction framework to a new scoring service.
- Accomplished automated model refresh for fake listing detection using Airflow.
- Implemented a feature contribution analysis tool to analyze which key model features contribute towards a high-risk score.

Microsoft — Apr. 2013 - May 2013
Intern — *Beijing, China*

- Built database for the project of Partners in Learning using SQL Server.
- Contributed to data organization and content correction for the project website.

PUBLICATIONS

- (Under Review) S.Shah, **K.Xu**, B.He, C.Maung and H.Schweitzer. Optimal Column Subset Selection and Related Problems under Unitarily Invariant Criteria. Association for Advancement of Artificial Intelligence (AAAI), 2018.
- **K.Xu**, H.Arai, C.Maung and H.Schweitzer. Two-Stage Feature Selection with Unsupervised Second Stage. International Conference on Tools with Artificial Intelligence (ICTAI), 2017.
- **K.Xu**, T.Cao, S.Shah, C.Maung and H.Schweitzer. Cleaning the Null Space: A Privacy Mechanism for Predictors. AAAI 2017.
- **K.Xu**, S.Shah, T.Cao, C.Maung and H.Schweitzer. Enhancing the Privacy of Predictors. AAAI 2017.
- H.Arai, **K.Xu**, C.Maung and H.Schweitzer. Weighted A* Algorithms for Unsupervised Feature Selection with Provable Bounds on Suboptimality. AAAI 2016.
- H.Arai, C.Maung, **K.Xu** and H.Schweitzer. Unsupervised Feature Selection by Heuristic Search with Provable Bounds on Suboptimality. AAAI 2016.