DISSERTATION

SPARSE MULTIVARIATE ANALYSES VIA ℓ_1-REGULARIZED OPTIMIZATION PROBLEMS SOLVED WITH BREGMAN ITERATIVE TECHNIQUES

Submitted by

Nicholas Rohrbacker

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2012

ABSTRACT

SPARSE MULTIVARIATE ANALYSES VIA ℓ_1-REGULARIZED OPTIMIZATION PROBLEMS SOLVED WITH BREGMAN ITERATIVE TECHNIQUES

In this dissertation we propose Split Bregman algorithms [38] for several multivariate analytic techniques for dimensionality reduction and feature selection including Sparse Principal Components Analysis, Bisparse Singular Value Decomposition (BSSVD) and Bisparse Singular Value Decomposition with an ℓ_1-constrained classifier BSSVD$_{l1}$. For each of these problems we construct and solve a new optimization problem using these Bregman iterative techniques. Each of the proposed optimization problems contain one or more ℓ_1-regularization terms to enforce sparsity in the solutions. The use of the ℓ_1-norm to enforce sparsity is a widely used technique, however, its lack of differentiability makes it more difficult to solve problems including these types of terms. Bregman iterations make these solutions possible without the addition of variables and algorithms such as the Split Bregman algorithm makes additional penalty terms and multiple ℓ_1 terms feasible, a trait that is not present in other state of the art algorithms such as the fixed point continuation algorithm [40]. It is also shown empirically to be faster than another iterative solver for total variaton image denoising, another ℓ_1-regularized problem, in [38]. We also link sparse Principal Components to cluster centers, denoise Hyperspectral Images using the BSSVD, identify and remove ambiguous observations from a classification problem using the algorithm and detect anomalistic subgraphs using Sparse Eigenvectors of the Modularity Matrix.

ACKNOWLEDGMENTS

Many thanks and acknowledgments need to be made. First, to my Advisor Michael Kirby for introducing me to the topics that would ultimately shape this thesis, supporting me in times of need, and always being able to give some much needed perspective. Christopher Peterson for his valuable guidance during my Master's work, and a compliment at the end of my Masters studies which I will always remember. To my Mom and Dad, Loella and Robert Rohrbacker, who were always proud of me, no matter what. My In Laws, Alan and Colleen Campbell, for their timely words of support. To my wonderful, loving and tolerant wife Dawn Rohrbacker who took on the extra burden during my absence and made all of this possible. And finally to Orion and Kepler Rohrbacker for always reminding me what truly matters.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Optimization problems featuring ℓ_1-constrained terms are becoming more familiar in the mathematical and statistical literature. The ℓ_1-norm has many well known properties including being robust to outliers and enforcing sparsity in optimization problems where it is present as part of the optimization problem. This has led to its use in many applications including variable selection [82], matrix rank minimization and completion [53, 19, 17, 15], multiple target tracking [44] and image analysis [38, 90] among others. While ℓ_1-optimization has been around for a while [29, 82, 2], there has been a recent explosion of interest in this area. This is due in part, to the success of compressed sensing [18, 98, 28].

Compressed sensing is the technique whereby a signal x of length n is recovered using $m \ll n$ linear functional measurements. This is achieved by solving an ℓ_1-minimization problem such as the basis pursuit problem

$$\underset{u}{\text{minimize}} \, \|u\|_1 + \frac{\lambda}{2} \|Au - b\|_2^2 \tag{1.1}$$

The ℓ_1-norm is also used in the relatively new field of large margin hyperplane classifiers, commonly known as Support Vector Machines (SVM) [12], [34]. In this setting, the ℓ_1-norm is used to enforce sparsity in a decision vector w where the decision function classifies data into two classes, C_+, C_- via

$$\operatorname{sgn}(x'w - \gamma) = \begin{cases} 1 & \text{then } x \in C_+, \\ -1 & \text{then } x \in C_-. \end{cases} \tag{1.2}$$

As can be seen by the decision function given in Equation (1.2), if w is sparse, then the variables that determine class separation are more easily determined.

The ℓ_1-norm has also been used in fields such as image denoising. In 2005, Stanley Osher used an ℓ_1 regularized optimization problem to minimize the total variation problem [64]. It was also used in [80] for hyperspectral image demixing.

Another well known use of ℓ_1 regularization is the Least Absolute Shrinkage and Selection Operator or LASSO for short, introduced by Tibshirani in 1996 [82]. In [82] the

ℓ_1-norm is used to enforce sparsity in the vector of coefficients in regression. The problem stated in [82] is given by the least-squares regression problem,

$$(\hat{\alpha}, \hat{\beta}) = \arg\min \left\{ \sum_{i=1}^{N} y_i - \alpha - \sum_{j=1}^{P} \beta_j x_{ij} \right\} \qquad \text{subject to } \ \|\beta\|_1 \leq t \tag{1.3}$$

where the goal is to solve for a vector of regression coefficients β subject to the constraint that the ℓ_1-norm of β is less than or equal to some real number t.

Beginning in 2003, sparsity within multivariate analytic techniques began with the introduction of the SCoTLASS algorithm [46]. This algorithm used the LASSO criteria to calculate sparse principal components. This was followed up by Zou et al. who also used as LASSO criteria to look for sparse principal components [102].

Other multivariate techniques also followed, including sparse Canonical Correlation Analysis (CCA) and sparse versions of the Singular Value Decomposition (SVD). Since the SVD is a necessary component for the solution of CCA at least two of the sparse SVD algorithms come from papers where sparse CCA is the goal [65, 93, 51, 3, 76].

Other areas of active research that may benefit from the introduction of ℓ_1 regularized problems are image segmentation and finding community structure in networks [74, 61, 60, 59]. In these problems, the task is to solve the combinatorial optimization problem [74, 62, 91]. However, the majority of these techniques involve a ”relaxation” of an eigenvector problem to allow continuous values in the results. Then, the values are assigned to either -1 or 1 based on their sign.

For example, in some cases, finding small subgraphs within a network is desirable. The formulation of this problem results in solving for eigenvectors of a matrix called the Modularity Matrix [57]. Solving for the eigenvectors of this symmetric matrix may be accomplished using Sparse PCA.

The remainder of this dissertation will be structured as follows. Chapter two will cover Bregman and Split Bregman Iterative schemes and their convergence. The ability to allow variable parameters, and a general dual formulation will also be covered. Chapter three will be devoted to a review of Sparse Principal Component Analysis and the implementation of Bregman iterations to that problem. Through a numerical example, the dimension reduction capability of the sparse PCA will be demonstrated. This capability will be further displayed using images of faces and allowing the Sparse PCA algorithm to reduce the data by choosing geometrically meaningful principal components. Chapter four will cover a sparse version of the Singular Value Decomposition (SVD) that we call the Bisparse SVD (BSSVD). The Defense Threat Reduction Agency (DTRA) is interested in algorithms that can identify plumes of possibly threatening chemicals and we will show an application using the BSSVD to this effect. In Chapter five an optimization problem for a Bisparse SVD with an ℓ_1-constrained classifier will be proposed and solved, we will refer to this algorithm as the BSSVD_{l1}. Applying this algorithm to both toy and real-world data has shown promising results in the area of feature selection. Feature selection in model building can be important in many ways, such as selecting the most important variables in a classification or predictive model, or by selecting the needed variables for dimensionality reduction. In this paper we focus on the ability of the algorithm to identify observations from a dataset meant for classification that are ambiguous, or inconsistent with the supplied labeling. We will examine the ambiguous data from a geometric viewpoint and also test to see if removing these ambiguous records during the training process aids in the overall classification rate. Chapter six will visit the modularity problem and will calculate sparse eigenvectors of the modularity matrix using the sparse PCA algorithm. Using this technique we are able to identify hidden subgraphs of interest. This problem mimics threat detection and anomaly detection in a network setting. Chapter 7 will be used to discuss areas of possible future work and conclusions.

2 Bregman and Split Bregman Iterations and Convergence

Bregman distances were introduced in 1967 by the mathematician L.M. Bregman [13]. Bregman was mainly concerned with finding a common point of convex sets, which corresponded to a solution to a convex optimization problem. Bregman introduced the distance

$$D_f(x,y) = f(x) - f(y) - (g(y), x - y)$$

where f is a strictly convex differentiable function, g is the gradient of f, and (x, y) denotes the dot product of the two vectors x and y. The Bregman distance is not a distance in the strict sense, since $D(x, y)$ does not necessarily equal $D(y, x)$[1].

Bregman iterations based on the Bregman distance were introduced by Stanley Osher in 2005 for image restoration based on total variation [64]. Osher also presented a solid theoretical foundation for using Bregman iterations to solve convex optimization problems. Since then, it has been applied to many different applications such as inverse scale space methods for image restoration [14], image super resolution via total variation regularization [55], likelihood estimation [58], wavelet based denoising [94], compressed sensing [98], hyperspectral image demixing [80], surface reconstruction [37] and a fused lasso and fused lasso support vector classifier [96]. In general, Bregman iterations based on Bregman distances were introduced as a method for solving convex optimization problems with ℓ_1-regularization terms. For example, in the realm of compressed sensing, Bregman iterations are used to solve the basis pursuit problem,

$$\underset{u}{\text{minimize}}\; \|u\|_1 + \frac{\lambda}{2}\|Au - b\|_2^2 \tag{2.1}$$

For the total variation based image restoration, the Split Bregman technique was used to solve

$$\underset{u}{\text{minimize}}\; \|\nabla_x u\|_1 + \|\nabla_y u\|_1 + \frac{\mu}{2}\|u - f\|_2^2$$

From these two examples, it is clear that the technique is able to be adapted to a range of different ℓ_1-regularized optimization problems.

[1]Consider the function $f(x) = x^4$. Here, $D(1, 2) = -11 \neq D(2, 1) = -17$.

2.1 Bregman and Split Bregman

In this section, we focus on the convergence of Bregman iterative schemes and extend this to the Split Bregman algorithm, which allows multiple ℓ_1 terms and easy addition of penalty terms [38]. These features make the Split Bregman algorithm easily adaptable to a wide range of ℓ_1-regularized problems.

We begin by focusing on the following problem statement:

$$\underset{u}{\text{minimize}}\,\{E(u) : H(u) = 0, u \in \mathbb{R}^n\} \tag{2.2}$$

where E and H are nonnegative convex functions. This equation is converted to an unconstrained problem by adding a penalty-term for the constraints.

$$\underset{u}{\text{minimize}}\ E(u) + \lambda H(u) \tag{2.3}$$

Based on the work by Bregman in his paper [13], if we define the Bregman distance associated with a function E and subgradient[2] of E at v, p_v as

$$D_E^{p_v} = E(u) - E(v) - (p_v, u - v),$$

then Goldstein and Osher [38] suggested iteratively solving

$$\begin{aligned} u_{k+1} &= \underset{u}{\text{minimize}}\ D_E^p(u, u_k) + \lambda H(u) \\ &= \underset{u}{\text{minimize}}\ E(u) - E(u_k) - (p_k, u - u_k) + \lambda H(u) \\ &= \underset{u}{\text{minimize}}\ E(u) - (p_k, u - u_k) + \lambda H(u) \end{aligned} \tag{2.4}$$

where the term $E(u^k)$ is dropped from the second line due to it being constant and not effecting the nonnegativity of the Bregman distance, nor the minimization of the original problem. The term p_k represents a subgradient of E at u_k. This means that in the case where E is a differentiable convex functional, that p_k is the gradient of E at u_k.

[2]See Appendix A for definition of subgradient

If we assume that $H(u)$ is differentiable and consider the subdifferential of $D_E^{p_k}(u, u_k + \lambda H(u))$ then

$$\partial E(u_{k+1}) - p_k + \nabla H(u) = 0$$

since by definition of the subdifferential, 0 will be in the subdifferential of this expression at u_{k+1}. Given that $p_{k+1} \in \partial E(u_{k+1})$ we can then write

$$p_{k+1} = p_k - \nabla H(u_{k+1}) \tag{2.5}$$

This is a variant of a formula, Equations (2.7) and (2.8), seen in the proof of theorem 3 in Bregman's original paper [13]

2.1.1 Theoretical Formulation

To show how to formulate the Split Bregman algorithm for our problem we will use three theorems and one lemma. The first theorem was stated in [38] and stated and proven in [98].

Theorem 1. *Assume E and H are both convex and that H is differentiable, then, H monotonically decreases in the iterates u_k.*

Proof. Define

$$Q_k(u) = \underset{u}{\text{minimize}}\ E(u) - E(u_{k-1}) - (p_{k-1}, u - u_{k-1}) + H(u).$$

By assumption, u_k minimizes Q_k, this, along with the nonnegativity of the Bregman distance implies that

$$\begin{aligned} H(u_k) &\le E(u_k) - E(u_{k-1}) - (p_{k-1}, u_k - u_{k-1}) + H(u_k) \\ &= Q_k(u_k) \le Q_k(u_{k-1}) = H(u_{k-1}) \end{aligned}$$

□

The next lemma provides an inequality that will help in the proof of the next theorem. Lemma 1 and Theorem 2 are a portion of a proposition and a theorem that appear in [64].

Lemma 1. *Assume that E and H are both convex and that H is also differentiable, then*

$$D^{p_k}(u,u_k) + D^{p_{k-1}}(u_k,u_{k-1}) + H(u_k) \leq H(u) + D^{p_{k-1}}(u,u_{k-1})$$

Proof.

$$\begin{aligned}
&D^{p_k}(u,u_k) - D^{p_{k-1}}(u,u_{k-1}) + D^{p_{k-1}}(u_k,u_{k-1})\\
=&E(u) - E(u_k) - (p_k, u-u_k) - E(u) + E(u_{k-1}) + (p_{k-1}, u-u_{k-1})\\
+&E(u_k) - E(u_{k-1}) - (p_{k-1}, u_k - u_{k-1})\\
=&-(p_k,u) + (p_k,u_k) + (p_{k-1},u) - (p_{k-1},u_{k-1}) - (p_{k-1},u_k) + (p_{k-1},u_{k-1})\\
=&(u_k - u, p_k - p_{k-1})\\
=&(u_k - u, -\nabla H(u_k))\\
=&(u - u_k, \nabla H(u_k))
\end{aligned}$$

Given that H is assumed to be convex, the final line implies that

$$(u-u_k, \nabla H(u_k)) \leq H(u) - H(u_k)$$

Substituting $D^{p_k}(u,u_k) - D^{p_{k-1}}(u,u_{k-1}) + D^{p_{k-1}}(u_k,u_{k-1})$ back into the left hand term and simple algebra give the result.

□

Theorem 2. *Assume that E and H are both convex and that H is also differentiable, then if $\tilde{u}$ is a minimizer of $H(u)$ such that $E(u) < \infty$ then*

$$H(u_k) \leq H(\tilde{u}) + \frac{E(\tilde{u})}{k}$$

Proof. Summing both sides of lemma 1 gives

$$\sum_{j=1}^{k} D^{p_j}(u,u_j) + D^{p_{j-1}}(u_j,u_{j-1}) + H(u_j) \leq \sum_{j=1}^{k} H(u) + D^{p_{j-1}}(u,u_{j-1}),$$

$$\sum_{j=1}^{k} D^{p_j}(u,u_j) - D^{p_{j-1}}(u,u_{j-1}) + \sum_{j=1}^{k} D^{p_{j-1}}(u_j,u_{j-1}) + H(u_j) - H(u) \leq 0$$

If we focus on the first term on the left side of this inequality it becomes clear that this is a telescoping sum which when expanded will become equivalent to $-D^{p_0}(u,u_0)+D^{p_k}(u,u_k)$, which when substituted back into the original inequality results in

$$D^{p_k}(u,u_k)+\sum_{j=1}^{k} D^{p_{j-1}}(u_j,u_{j-1})+H(u_j)-H(u)\leq D^{p_0}(u,u_0)$$

Since $D^{p_k}(u,u_k)\geq 0$, $D^{p_{k-1}}(u_k,u_{k-1})\geq 0$ and $H(u)$ is monotonic we can conclude that

$$k\left[H(u_k)-H(u)\right]\leq D^{p_0}(u,u_0)\leq E(u),$$

which then implies that

$$H(u_k)\leq H(u)+\frac{E(u)}{k}.$$

$\square$

Theorem 2 states that if E and H satisfy some simple assumptions, then the sequence $\{u_j\}$ chosen through the Bregman iterations will converge, i.e. $H(u_j)\to H(\tilde{u})$ as $j\to\infty$

Theorem 3 below gives the final convergence result, namely the fact that the Bregman iterative scheme proposed in Equation (2.4) will converge to an optimal solution given minor assumptions on E. This theorem is a slight variant of a theorem proven in [38], in that the form of $H(u)$ is unrestricted.

Theorem 3. *Let E be convex, and assume that $H(u)$ is convex and differentiable with minimum value 0. Also assume that some iterate u^* satisfies $H(u)=0$. Then, u^* is a solution to the problem,*

$$\text{minimize}\,\{E(u):H(u)=0,u\in\mathbb{R}^n\}\tag{2.6}$$

Proof. Let u^* be such that $H(u^*)=0$ and

$$u^*=\min_u E(u)+\lambda H(u)\tag{2.7}$$

If $\hat{u}$ is a solution to the original problem given by Equation(2.6) then $H(u^*) = H(\hat{u}) = 0$. Since u^* is a solution of Equation 2.7, this implies that

$$E(u^*) + \lambda H(u^*) \leq E(\hat{u}) + \lambda H(\hat{u})$$

Given the equality of the second term on both sides of the equation we can conclude that $E(u^*) \leq E(\hat{u})$. Plus, since $\hat{u}$ solves Equation (2.6) implies that this inequality can be changed to an equality and that u^* is also a solution to Equation (2.6). □

Thus, using Theorems 1, 2 and 3 and Lemma 1, we can conclude the following: Theorem 1 proves that H will monotonically decrease in the iterates u. Lemma 1 and Theorem 2 show that H will monotonically decrease towards the minimum of H provided that E is finite. Finally, Theorem 3 proves that the iterates will approach a solution to (2.6) since the second term in this problem satisfy the assumptions on H in the previous theorems and lemma.

2.1.2 Split Bregman

The goal of Split Bregman is to split the ℓ_2/differentiable portion from the ℓ_1 portions of the problem. To begin, consider the following variation of the original unconstrained problem for a differentiable function Φ

$$\underset{u,d}{\arg\min}\ \|d\|_1 + F(u) \text{ s.t. } d = \Phi(u),$$

which can be rewritten as:

$$\underset{u,d}{\arg\min}\ \|d\|_1 + F(u) + \frac{\lambda}{2}\|d - \Phi(u)\|_2^2. \tag{2.8}$$

This is a restatement of the original problem with $E(u,d) = \|d\|_1 + F(u)$ and $H(u,d) = \|d - \Phi(u)\|_2^2$. Thus we look to iteratively solve

$$(u_{k+1}, d_{k+1}) = \underset{u,d}{\arg\min}\ \|d\|_1 + F(u) - (p_k^u, u - u_k) - (p_k^d, d - d_k) + \frac{\lambda}{2}\|d - \Phi(u)\|_2^2 \tag{2.9}$$

Here is where the split takes place. Equation (2.9) is split into the portions of the problem that include differentiable components, and non-differentiable components i.e.,

1. $u_{k+1} = \arg\min_u F(u) - (p_k^u, u - u_k) + \frac{\lambda}{2} \|d_k - \Phi(u)\|_2^2$

2. $d_{k+1} = \arg\min_d \|d\|_1 - (p_k^d, d - d_k) + \frac{\lambda}{2} \|d - \Phi(u_{k+1})\|_2^2$

3. $p_{k+1}^u = p_k^u - \nabla H_u(u_{k+1})$

4. $p_{k+1}^d = p_k^d - \nabla H_d(d_{k+1})$

where ∇H_u indicates the gradient of H with respect to u and ∇H_d is defined similarly. For the first equation, since all components are differentiable, the structure of the problem can be used to identify the best solution method. For the second equation, since the components can all be calculated separately from one another, shrinkage operators are used to find the optimal solution [40, 82] i.e.,

$$(d_{k+1})_j = shrink(\Phi(u_{k+1})_j + (p_k^d)_j, \frac{1}{\lambda})$$

where

$$shrink(x, \gamma) = \frac{x}{|x|} \cdot max(|x| - \gamma, 0).$$

A second formulation of the problem given in [98] removes terms involving the subdifferential and can be easier to implement when the constraints associated with a problem are linear. For this, we need Theorem 4. First, note that $\frac{\partial}{\partial u}(\frac{1}{2} \|Au - b\|_2^2) = A^T(Au - b)$. Substituting this into ∇H gives

$$p_{k+1} = p_k - A^T(Au_{k+1} - b) \tag{2.10}$$

Theorem 4. *The following versions, V1 and V2 of the Bregman iterative scheme are equivalent*

V1:

```
$u_0 := 0, \quad p_0 := 0$
For $k = 0, 1, \ldots,$ do
    $u_{k+1} = \underset{u}{\text{minimize}}\ D^{p_k}(u, u_k) + \frac{1}{2}\|Au - b\|_2^2$    (2.11)
    $p_{k+1} = p_k - A^T(Au_{k+1} - b)$
```

V2:

```
$b_0 := 0, \quad u_0 := 0$
For $k = 0, 1, \ldots,$ do
    $b_{k+1} = b + (b_k - Au_{k+1})$
    $u_{k+1} = \underset{u}{\text{minimize}}\ E(u) + \frac{1}{2}\|Au - b_{k+1}\|_2^2$    (2.12)
```

Proof. At $k = 0$, (3.9) gives:

$$\begin{aligned}
&=\underset{u}{\text{minimize}}\ D^{p_0}(u, u_0) + \frac{1}{2}\|Au - b\|_2^2 \\
&=\underset{u}{\text{minimize}}\ E(u) - (0, u - 0) + \frac{1}{2}\|Au - b\|_2^2 \\
&=\underset{u}{\text{minimize}}\ E(u) + \frac{1}{2}\|Au - b\|_2^2
\end{aligned}$$

Note that in the second line, $E(u_0)$ has been dropped as it is constant and will not change the calculation of the minimum. At $k = 0$, (2.12) gives:

$$\begin{aligned}
&=\underset{u}{\text{minimize}}\ E(u) + \frac{1}{2}\|Au - b_1\|_2^2 \\
&=\underset{u}{\text{minimize}}\ E(u) + \frac{1}{2}\|Au - (b + (b_0 - Au_0))\|_2^2 \\
&=\underset{u}{\text{minimize}}\ E(u) + \frac{1}{2}\|Au - b\|_2^2
\end{aligned}$$

[40] showed that for all optimal solutions, $A^T(b - Au)$ is constant. This implies that $A^T(b - Au_1) = A^T(b - A\bar{u}_1)$ for all optimal solutions. Hence

$$p_1 = p_0 - A^T(Au_1 - b) = A^T(b - Au_1) = A^T(b - A\bar{u}_1) = A^T(b_1 - A\bar{u}_1)$$

Using induction on $p_k = A^T(b_k - A\bar{u}_k)$, and moving all terms that are constant in u into the constants C_1, C_2 and C_3,

$$\begin{aligned} D^{p_k}(u,u_k) + \frac{1}{2}\left\|Au-b\right\|_2^2 &= E(u) - (p_k,u) + \frac{1}{2}\left\|Au-b\right\|_2^2 + C_1 \\ &= E(u) - (b_k - A\bar{u}_k, Au) + \frac{1}{2}\left\|Au-b\right\|_2^2 + C_2 \\ &= E(u) - (b_k - A\bar{u}_k, Au) + \frac{1}{2}(Au-b, Au-b) + C_2 \end{aligned}$$

If we focus on the term $(b_k - A\bar{u}_k, Au) - \frac{1}{2}(Au-b, Au-b)$ we see that this term

$$\begin{aligned} &= (b_k, Au) - (A\bar{u}_k, Au) + \frac{1}{2}(Au,Au) - (Au,b) + \frac{1}{2}(b,b) \\ &= \frac{1}{2}\left[2(b_k,Au) - 2(A\bar{u}_k,Au) + (Au,Au) - 2(Au,b) + (b,b)\right] \\ &= \frac{1}{2}\left[2(b_k,Au) - 2(A\bar{u}_k,Au) + (Au,Au) - 2(Au,b) + (b,b)\right] + \\ &\quad \frac{1}{2}\left[(b_k,b_k) + (A\bar{u}_k,A\bar{u}_k) - (b_k,b_k) - (A\bar{u}_k,A\bar{u}_k)\right] \\ &= \frac{1}{2}(Au - (b + (b_k - A\bar{u}_k)), Au - (b + (b_k - A\bar{u}_k))) - \frac{1}{2}(b_k,b_k) - \frac{1}{2}(A\bar{u}_k,A\bar{u}_k) \end{aligned}$$

if $C3 := C_2 - \frac{1}{2}(b_k,b_k) - \frac{1}{2}(A\bar{u}_k,A\bar{u}_k)$ then continuing from above we get

$$\begin{aligned} &= E(u) - (b_k - A\bar{u}_k, Au) + \frac{1}{2}\left\|Au-b\right\|_2^2 + C_2 \\ &= E(u) + \frac{1}{2}\left\|Au - (b + (b_k - A\bar{u}_k))\right\|_2^2 + C_3 \\ &= E(u) + \frac{1}{2}\left\|Au - b_{k+1}\right\|_2^2 + C_3. \end{aligned}$$

Which we see is equivalent to Equation (2.12). □

Theorem 4 allows us to write the Bregman iterative problem

$$\begin{aligned} u_{k+1} &= \arg\min_u D^{p_k}(u,u_k) + \frac{\lambda}{2}\left\|Au-b\right\|_2^2 \\ p_{k+1} &= p_k - \lambda A^T(Au_{k+1} - b) \end{aligned}$$

in the following form:

$$\begin{aligned} u_{k+1} &= \arg\min_u E(u) + \frac{\lambda}{2}\left\|Au-b_k\right\|_2^2 \\ b_{k+1} &= b_k + b - Au_k. \end{aligned}$$

To put this in the Split Bregman framework, the ℓ_1 and ℓ_2 portions are again split apart, i.e.,

$$\begin{aligned}(u_{k+1}, d_{k+1}) &= \arg\min_{u,d} \ \|d\|_1 + H(u) + \frac{\lambda}{2}\|d - u - b_k\|_2^2 \\ b_{k+1} &= b_k + u_{k+1} - d_{k+1}.\end{aligned}$$

The solution method is similar to the original formulation to solve the first problem. It is split into the portions of the problem that include differentiable components, and non-differentiable components, i.e.,

1. $u_{k+1} = \arg\min_{u} H(u) + \frac{\lambda}{2}\|d_k - u - b_k\|_2^2$
2. $d_{k+1} = \arg\min_{d} \ \|d\|_1 + \frac{\lambda}{2}\|d - u_{k+1} - b_k\|_2^2$
3. $b_{k+1} = b_k + u_{k+1} - d_{k+1}$.

It should be noted that it has been shown in [32] that the Split Bregman method can be equivalent to the Alternating Direction Method of Multipliers (ADMM) found in [35]. However, it is also noted that this is in general only the case when the constraints on the optimization problem are linear. As evidenced by the convergence theory of the Split Bregman method it is able to allow for convex constraints.

2.2 Addition of Penalty Terms

The addition of penalty terms in the Split Bregman Framework is generally very simple and one of its major advantages over other algorithms. How to change the optimization problem depends on if the function is differentiable or not. For example, consider the basic problem given by,

$$\underset{u}{\text{minimize}} \ \|u\|_1 + \|Au - f\|_2^2 \tag{2.13}$$

Where $u \in \mathbb{R}^n, f \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times n}$. In the Split Bregman framework, using a simplification presented in [38], this is represented as

$$\underset{u,d}{\text{minimize}} \ \|d\|_1 + \frac{\lambda_1}{2}\|Au - f\|_2^2 + \frac{\lambda_2}{2}\|d - u - b\|_2^2 \tag{2.14}$$

Where b has taken the place of the Bregman update parameter which previously was given by p^k.

In order to add an ℓ_1 term $P_1(u)$ the problem would need to be changed as follows

$$\underset{u,d,d_P}{\text{minimize}}\ \|d\|_1+\|d_P\|_1+\frac{\lambda_1}{2}\|Au-f\|_2^2+\frac{\lambda_2}{2}\|d-u-b\|_2^2+\frac{\lambda_3}{2}\|d_P-P_1(u)-b_P\|_2^2 \tag{2.15}$$

To add a differentiable term, the term is just added to the main optimization problem, with no additional variables needed. Let $P_1(u)$ be a differentiable function of u, then we have the following Bregman interpretation of the problem.

$$\underset{u,d}{\text{minimize}}\ \|d\|_1+\frac{\lambda_1}{2}\|Au-f\|_2^2+\frac{\lambda_2}{2}\|d-u-b\|_2^2+\lambda_3 P_1(u) \tag{2.16}$$

The differentiable term, $P_1(u)$ will be added to the iteration of u_{i+1}, and will not be present when working with the ℓ_1 penalized terms.

2.3 Non-constant Penalty Parameters on $E(u)$

The proof of convergence for this technique given above does not allow variable parameters. However, the ability to change parameters attached to penalty terms may have beneficial effects in regards to sparsity levels and iteration step size. Following the outline given in [98] we note that if we let $E(u)_k := \mu_k E(u)$, then, a subgradient for $E(u)_k$ would be given by $\mu_k p_k$. Considering the following problem,

$$\underset{u}{\text{minimize}}\ \mu_k E(u)+\lambda H(u) \tag{2.17}$$

and its Bregman version

$$\underset{u}{\text{minimize}}\ \mu_k E(u)-(p_{k-1},u-u_{k-1})+\lambda H(u). \tag{2.18}$$

We note that p_{k-1} solved for in the iterative sense would actually be a subgradient of $E(u)_{k-1}$ and therefore the problem that needs to be solved is given by

$$\underset{u}{\text{minimize}}\ \mu_k E(u)-\frac{\mu_k}{\mu_{k-1}}(p_{k-1},u-u_{k-1})+\lambda H(u). \tag{2.19}$$

Further calculations give that

$$p_{k+1} = -\mu_{k+1} \sum_{j=1}^{k+1} \frac{\lambda \nabla H(u_j)}{\mu_j} \tag{2.20}$$

Using the proof of Theorem 4 above, and the third part of the proof of Theorem 3.1 in [38] we have that

$$\begin{aligned} p_{k+1} = \frac{\mu_k}{\mu_{k-1}} p_k - A^T(Au_{k+1} - b) &= \frac{\mu_k}{\mu_{k-1}} p_k - A^T(A\bar{u}_{k+1} - b) \\ &= \frac{\mu_k}{\mu_{k-1}} A^T(b_k - A\bar{u}_k) - A^T(A\bar{u}_{k+1} - b) \\ &= A^T(b + \frac{\mu_k}{\mu_{k-1}}(b_k - A\bar{u}_k) - A\bar{u}_{k+1}). \end{aligned} \tag{2.21}$$

When used in conjunction with equation 3.9 in [98], this gives that

$$b_{k+1} = b + \frac{\mu_k}{\mu_{k-1}}(b_k - Au_k) \tag{2.22}$$

Thus, if the sequence of μ_k is bounded, we can see that while the number of necessary iterations may increase, the overall convergence of the optimization problem will not be compromised.

2.4 The Dual Formulation

The derivations found in this section come mainly from [32] and [99]. Consider a convex functional $J(u)$ and the following constrained optimization problem:

$$\begin{aligned} &\underset{u \in \mathbb{R}^m}{\text{minimize}}\ J(u) \\ &\text{subject to } Ku = f \end{aligned} \tag{2.23}$$

Where $K \in \mathbb{R}^{s \times m}$ and $f \in \mathbb{R}^s$. If we assume that $J(u)$ can be split into two different functions, $J(u) = E(u) + H(u)$, then we can form

$$\begin{aligned} &\underset{u,z \in \mathbb{R}^m}{\text{minimize}}\ E(z) + H(u) \\ &\text{subject to } Bz + Au = f. \end{aligned} \tag{2.24}$$

Where B is defined to be,

$$B = \begin{pmatrix} I_{m,m} \\ 0_{s,m} \end{pmatrix}$$

A is defined as,

$$A = \begin{pmatrix} I_{m,m} \\ K \end{pmatrix}$$

and b is

$$b = \begin{pmatrix} 0_{m,1} \\ f \end{pmatrix}.$$

Equation 2.24 is equivalent to

$$\begin{aligned} &\underset{u,z\in\mathbb{R}^m}{\text{minimize}}\ E(z) + H(u) \\ &\text{subject to } z = u \text{ and } Ku = f \end{aligned} \tag{2.25}$$

To find the dual formulation, we begin by forming the Lagrangian.

$$L(z, u, \lambda) = E(z) + H(u) + (\lambda, b - Au - Bz). \tag{2.26}$$

If we want to find the dual associated with L, then we look at

$$\max_{\lambda\in\mathbb{R}^{s+m}} q(\lambda) \tag{2.27}$$

where $q(\lambda) = \min_{z,u\in\mathbb{R}^m} L(z, u, \lambda)$. The dual functional is generally written in terms of the Legendre-Fenchel transform [32, 99, 70].

$$\begin{aligned} q(\lambda) &= \min_{z,u} E(z) + H(u) + (\lambda, b - Au - Bz) \\ &= \min_{z} E(z) - (\lambda, Bz) + \Big(\min_{u} H(u) - (\lambda, Au)\Big) + (\lambda, b) \\ &= -\max_{z}(B^T\lambda, z) - E(z) - \Big(\max_{u}(A^T\lambda, u) - H(u)\Big) + (\lambda, b) \\ &= -E^*(B^T\lambda) - H^*(A^T\lambda) + (\lambda, b) \end{aligned} \tag{2.28}$$

Where E^* and H^* are the Legendre-Fenchel transforms of E and H defined by

$$E^*(B^T\lambda) = \max_{z}(B^T\lambda, z) - E(z) \tag{2.29}$$

$$H^*(A^T\lambda) = \max_{u}(A^T\lambda, u) - H(u) \tag{2.30}$$

2.5 Summary

In this section we covered the Split Bregman method and its applicability to ℓ_1-regularized optimization problems. We showed convergence under mild assumptions on the terms in the objective function, gave examples of how to add additional penalty terms, briefly explored the notion of variable parameters and calculated a general form for the dual formulation.

Much of the work above was expository in nature, confirming that this method would be satisfactory for the problems of interest. However, there were novel contributions that furthered the work that has been done in this area. These include the explicit instruction for how to add additional penalty terms to the objective function, providing a fully contained proof of convergence, and expanding the details within these proofs. All of these are important as they provide a solid foundation for using this framework to its fullest extent and in many cases are constructive in the sense of providing guidelines forn when the method is appropriate and how to tailor it to the problem at hand.

3 Sparse Principal Components Analysis

Principal Components Analysis is a multivariate statistical technique which replaces observed variables with derived variables [45]. These derived variables are linear combinations of the original variables called principal components. The principal components for a given data set is the orthonormal set of vectors, that explains the most variance within the data set when compared to all other orthonormal bases [45]. PCA has many uses including dimensionality reduction, classification, clustering, facial recognition, and independent variable creation for regression modeling [95, 47, 78, 48]. For many of these applications, it is desirable that the principal components be easily interpreted, and this is sometimes the case. Interpretation is generally easier when the number of non-zero loadings is small. If a principal component has non-zero loadings on a large number of the variables interpretation is more difficult. However, since the principal components are linear combinations of all of the observed variables, it is not guaranteed that the loadings will be sufficiently sparse in order to make interpretation easy. The ℓ_1-norm has been introduced in the last several years as a way to force as many zero entries in the PCA coefficients as possible [46, 102].

Sparse Principal Component Analysis dates back to the 1980's when Hausman restricted the set of loading coefficients in PCA to the set $\{1, 0, -1\}$ by using a branch and bound algorithm [41]. In 2000, Vines proposed a method that restricts the coefficients to be integers. His algorithm was an iterative algorithm based on the Jacobi method [88]. The algorithm SCoTLASS proposed by Jolliffe in 2003 was one of the first algorithms to attempt to force zero coefficients in the principal components [46]. It does this by enforcing a lasso constraint on the normal formulation of the principal component optimization problem [82]. One of the most influential papers written on sparse PCA came in 2006 from Zou et al. In their paper, PCA is reformulated as a regression problem and Ridge and Lasso penalization terms are added to enforce sparsity [102]. The field of sparse principal components has grown in recent years, e.g. [73, 26, 43, 52, 36, 27].

Table 3.1: Variables Included in Jeffers' Pitprops Data.

Variable	Description
x_1	Top diameter in inches
x_2	Length in inches
x_3	Moisture content, percent of dry weight
x_4	Specific gravity at time of test
x_5	Oven-dry specific gravity
x_6	Number of annual rings at top
x_7	Number of annual rings at bottom
x_8	Maximum bow in inches
x_9	Distance of point of maximum bow from top in inches
x_{10}	Number of knot whorls
x_{11}	Length of clear prop from top in inches
x_{12}	Average number of knots per whorl
x_{13}	Average diameter of the knots in inches

A classic example of the potential difficulty in interpreting principal components (PCs) is illustrated by the pitprops data introduced by [42] that consists of 180 observations of 13 variables. The following two tables taken from [46] help illustrate this concept. Table 3.1 contains descriptions of each of the variables and Table 3.2 contains the loadings for the first six principal components. As can be seen by the values in Table 3.2 the interpretation of the components is made difficult by the large number of loadings that could be interpreted as significant or non-significant present in each PC. Granted, in each PC there are larger values and smaller values by magnitude, but there are also always values between these that would make association of any one of the PC's back to a small subset of the original variables difficult.

There have been many different formulations used to solve the sparse PCA problem.

Table 3.2: PCA Loadings for Pitprops Data

Variables	PC1	PC2	PC3	PC4	PC5	PC6
x_1	0.404	0.212	-0.219	-0.027	-0.141	-0.086
x_2	0.406	0.180	-0.245	-0.025	-0.188	-0.111
x_3	0.125	0.546	0.114	0.015	0.433	0.120
x_4	0.173	0.468	0.328	0.010	0.361	-0.090
x_5	0.057	-0.138	0.493	0.254	-0.122	-0.560
x_6	0.284	-0.002	0.476	-0.153	-0.269	0.032
x_7	0.400	-0.185	0.261	-0.125	-0.176	0.030
x_8	0.294	-0.198	-0.222	0.294	0.203	0.103
x_9	0.357	0.010	-0.202	0.132	-0.117	0.103
x_{10}	0.379	-0.252	-0.120	-0.201	0.173	-0.019
x_{11}	-0.008	0.187	0.021	0.805	-0.302	0.178
x_{12}	-0.115	0.348	0.066	-0.303	-0.537	0.371
x_{13}	-0.112	0.304	-0.352	-0.098	-0.209	-0.671
Simplicity Factor (Varimax)	0.059	0.103	0.082	0.397	0.086	0.266
Variance (%)	32.4	18.2	14.4	8.9	7.0	6.3
Cumulative Variance (%)	32.4	50.7	65.0	74.0	80.9	87.2

Here, we give an overview of the problems associated with four of the most well-known sparse PCA algorithms. To begin, we look at the problem solved for SCoTLASS in [46]. In that problem, the authors look for vectors a_k that maximize the variance with respect to the correlation matrix R with a scaling and an orthogonality constraint,

$$\text{maximize}_{a_k} \quad a_k^T R a_k \tag{3.1}$$
$$a_k^T a_k = 1 \text{ and } a_k^T a_h = 0 \text{ for } k \neq h$$

with the added constraint that $\|a_k\|_1 \leq t$ for some positive real-value t. In [26] the authors use a relaxation of a semidefinite programming problem to directly restrict the cardinality of the solutions. If we define $\mathbf{Tr}(A)$ to be the trace of the matrix A and $|X|$ to be the sum of the absolute values of the entries of X, then their optimization problem is based on finding a rank one matrix X which optimizes the following,

$$\begin{aligned} \underset{X}{\text{maximize}} \quad & \mathbf{Tr}(AX) - \rho \mathbf{1}^T |X| \mathbf{1} \\ \text{subject to} \quad & \mathbf{Tr}(X) = 1, \\ & X \succeq 0, \end{aligned} \tag{3.2}$$

where the constraint $\mathbf{Tr}(X) = 1$ results in the norm of solution being equal to one and $X \succeq 0$ means that the solution matrix X is positive semidefinite. Furthermore, the greedy algorithm they construct can find all of the possible solutions, meaning all solutions with cardinality 1 to n. Zou et al. in [102] recast PCA as a regression problem in order to utilize the Elastic Net penalty.

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{\alpha,\beta} \sum_{i=1}^{n} \left\|X_i - \alpha\beta^T X_i\right\|_2^2 + \lambda \sum_{j=1}^{k} \beta_j^2 + \sum_{j=1}^{k} \lambda_{1,j} |\beta_j| \tag{3.3}$$
$$\text{subject to } \alpha^T \alpha = I_k$$

In the above equation, X is an $n \times p$ matrix, α and β are both $p \times k$ matrices. For the ridge penalty term, the parameter λ stays fixed for all entries of β. However, for the LASSO penalty, the $\lambda_{1,j}$ can change for each entry of β. In cases where the number of variables is much larger than the number of observations, the problem is solved using soft thresholding rather than regression methods. Finally, in [73] they solve the following optimization problem

$$\text{minimize } \left\| X - \tilde{u}\tilde{v}^T \right\|_F^2 + P_\lambda(\tilde{v}) \tag{3.4}$$

Which is equivalent to minimizing the Frobenius error of a rank one approximation while enforcing a sparsity penalty on $\tilde{v}$ using the penalty term $p_\lambda(\tilde{v})$. In the algorithm used to solve this problem, $\tilde{u}$ continually changes as the iterations progress, and is also scaled at each step to have unit norm. Thus, from an interpretation standpoint, the basis vectors on which the sparse loading vector, $\tilde{v}$ is based, are constantly changing throughout the algorithm. They also offer three different penalties to use for $P_\lambda(\tilde{v})$, soft-thresholding, hard thresholding and the SCAD (smoothly clipped absolute deviation) [33].

3.1 Bregman Sparse PCA

To formulate our sparse PCA problem we begin by constructing the optimization problem to be solved. For the approximation to the *kth* Sparse PCA loading vector (LV) v_k we will solve the problem

$$\underset{v}{\text{minimize}} \ \ \|v\|_1 + \frac{\lambda_1}{2} \left\| X - \sigma^k u^k v^T \right\|_F^2 + \frac{\mu}{2} \|v\|_2^2 \tag{3.5}$$

where σ^k and u^k are the *kth* left singular value and singular vector (basis vector) from the SVD of X respectively. The way our optimization problem is structured utilizes the Frobenius error as in [73], but also adds in the ridge penalty as in [102]. Also, as will be seen when the algorithm is detailed, u^k stays fixed during the iterations and the sparse principal components (PC) are calculated using the sparse loading vectors as $\tilde{U} = X\tilde{V}S^{-1}$ where the SVD of X is given by $X = USV^T$ and $\tilde{V}$ is the set of sparse loading vectors calculated during the algorithm.

In order for this problem to satisfy the theorems given above, we need to show convexity[1]

[1]See Appendix A for the definition of convexity

of all the terms and differentiability of the last two terms. For the first term, it is known that the ℓ_1 norm is not differentiable anywhere where one of the components is equal to zero. However, it is easily shown to be convex.

If f is defined to be the ℓ_1-norm, then we aim to show that

$$\begin{aligned} f(tx+(1-t)y) &\leq tf(x)+(1-t)f(y) \qquad (3.6)\\ \|tx+(1-t)y)\|_1 &\leq t\|x\|_1+(1-t)\|y\|_1 \end{aligned}$$

Which, since we are dealing with a norm, is true by the triangle inequality. For the Frobenius norm portion, a similar approach will show convexity, however, convexity and differentiability can be shown by expanding the norm itself. The Frobenius norm for an $m \times n$ matrix X is defined as

$$\|X\|_F = \sum_{i,j} x_{i,j}^2 \qquad (3.7)$$

This can be viewed as a function from $\mathbb{R}^{mn} \to \mathbb{R}$. Expansion of the Frobenius norm term in Equation (3.5) shows that the optimization problem is twice differentiable and that the Hessian is diagonal with positive entries and hence postive semidefinite. Hence that term is also convex. A similar approach will show that the final ℓ_2 norm term is also differentiable and convex.

In the Split Bregman framework, using the simplification presented, this will be solved as

$$\underset{v,d}{\text{minimize}} \ \|d\|_1 + \frac{\lambda_1}{2}\left\|X-\sigma_k u_k v^T\right\|_F^2 + \frac{\mu}{2}\|v\|_2^2 + \frac{\lambda_2}{2}\|d-v\|_2^2 \qquad (3.8)$$

This will give the following iterations:

$$\begin{aligned} v_{k+1} &= \arg\min_v \ \frac{\lambda_1}{2}\left\|X-\sigma^k u^k v^T\right\|_{Fro}^2 + \frac{\mu}{2}\|v\|_2^2 + \frac{\lambda_2}{2}\|d_k - v - b_k\|_2^2 \qquad (3.9)\\ d_{k+1} &= \arg\min_d \ \|d\|_1 + \frac{\lambda_2}{2}\|d-v_k-b_k\|_2^2 \qquad (3.10)\\ b_{k+1} &= b_k + v_{k+1} - d_{k+1} \qquad (3.11) \end{aligned}$$

To solve for v_{k+1} we differentiate with respect to v i.e.

$$\begin{aligned}\frac{\partial v_{k+1}}{\partial v} &= \frac{\partial}{\partial v}\left(\frac{\lambda_1}{2}\left\|X-\sigma^k u^k v^T\right\|_F^2+\frac{\mu}{2}\|v\|_2^2+\frac{\lambda_2}{2}\|d_k-v-b_k\|_2^2\right) \qquad (3.12)\\ &= \frac{\partial}{\partial v}\left(\frac{\lambda_1}{2}\sum_{i,j}\left(X_{i,j}-(\sigma^k u^k v^T)_{i,j}\right)^2+\frac{\mu}{2}\|v\|_2^2+\frac{\lambda_2}{2}\|d_k-v-b_k\|_2^2\right)\\ &= \frac{\partial}{\partial v}\left(\frac{\lambda_1}{2}\sum_{i,j}X_{i,j}^2-2X_{i,j}(\sigma^k u^k v^T)_{i,j}+(\sigma^k u^k v^T)_{i,j}^2+\frac{\mu}{2}\|v\|_2^2+\frac{\lambda_2}{2}\|d_k-v-b_k\|_2^2\right)\\ &= \frac{\partial}{\partial v}\left(\frac{\lambda_1}{2}(\|X\|_F^2-\sum_{i,j}2X_{i,j}(\sigma^k u^k v^T)_{i,j}+\left\|\sigma^k u^k v^T\right\|_F^2)+\frac{\mu}{2}\|v\|_2^2+\frac{\lambda_2}{2}\|d_k-v-b_k\|_2^2\right)\\ &= -\lambda_1\sigma^k X^T u^k+\lambda_1\sigma^{k^2}(u^k)^T u^k v+\mu v-\lambda_2(d_k-b_k-v)\end{aligned}$$

Setting this equal to zero gives

$$v_{k+1}=\frac{\lambda_1\sigma^k X^T u^k+\lambda_2(d_k-b_k)}{\lambda_1\sigma^{k^2}+\lambda_2+\mu} \qquad (3.13)$$

For d_{k+1} we find the subdifferential with respect to d due to the ℓ_1 terms. Thus we look for

$$\frac{\partial d_{k+1}}{\partial d}=\frac{\partial}{\partial d}\left(\|d\|_1+\frac{\lambda_2}{2}\|d-v_{k+1}-b_k\|_2^2\right) \qquad (3.14)$$

$$=\Gamma+\lambda_2 d-\lambda_2 v_{k+1}-\lambda_2 b_k \qquad (3.15)$$

where Γ is defined as

$$\Gamma_i=\begin{cases}1 & \text{if } d_i>0,\\ [-1,1] & \text{if } d_i=0,\\ -1 & \text{if } d_i\le 0.\end{cases} \qquad (3.16)$$

According to [40, 70], d is a solution to Equation (3.15) if and only if the subdifferential of Equation (3.15) is 0 when evaluated at d. Equation (3.15) can be solved componentwise, and will require three cases. Note that using simple algebra on the components we get that

$$d_i=(v_{k+1})_i+(b_k)_i-\frac{1}{\lambda_2}\Gamma_i. \qquad (3.17)$$

If $d_i>0$ then $\Gamma_i=1$ which gives that

$$v_{k+1_i}+b_{k_i}-\frac{1}{\lambda_2}\Gamma_i>0 \qquad (3.18)$$

which implies that $v_{k+1_i} + b_{k_i} > \frac{1}{\lambda_2}$. We get an analogous result when $d_1 < 0$. In that case $\Gamma_i = -1$ and therefore we get that $v_{k+1_i} + b_{k_i} < \frac{1}{\lambda_2}$. If $d_i = 0$, we have that $\Gamma_i \in [-1, 1]$. Thus, using the previous two results we get that $\frac{-1}{\lambda_2} \leq v_{k+1_i} + b_{k_i} \leq \frac{1}{\lambda_2}$. If we combine all three results, we get the shrinkage operator mentioned in section two, namely that

$$d_i = sign(v_{k+1_i} + b_{k_i}) * max(|v_{k+1_i} + b_{k_i}| - \frac{1}{\lambda_2}, 0) \tag{3.19}$$

The iteration for b_{k+1} is just given by

$$b_{k+1} = b_k + v_{k+1} - d_{k+1} \tag{3.20}$$

When solving for subsequent loading vectors v_k, we enforce orthogonality by looking at the complementary projection of X onto $v_1, \ldots, v_{k-1}$ where v_i is the ith sparse loading vector. For example, after solving for the first loading vector, the data matrix used to solve for the second loading vector would be $\hat{X}$ defined by

$$\hat{X} = X^T \left(I - v_1 v_1^T \right).$$

Since the optimization problem is based on Frobenius error, and by definition the first PC and loading vector will explain a maximum amount of that error, any deviation from that combination will result in some of that initial variance remaining. Hence, when solving for the second vector, some of the original variance that would have been explained by the first traditional PCA pair remains and will at times cause some non-orthogonality in the sparse loading vector solutions. The extent to which this is the case depends on many factors, including parameter selection and the data being analyzed. If the calculated sparse loading vectors are sufficiently sparse, for example, each picking out one variable, then there can be orthogonality at the expense of variance explained.

The algorithm for computing the first N sparse loading vectors is given below. For each iteration of the inner while loop, the operations count for the calculation of v_{k+1} is dominated by the matrix-vector multiplication, which if the data matrix X is $m \times n$

will mean that each loop will be on the order of $O(mn)$. Computation of both d_{k+1} and b_{k+1} will be $O(n)$, meaning that the inner loop will be dominated by the matrix-vector multiplication with order $O(mn)$.

To calculate the first N loading vectors for a data matrix X of size $m \times n$:

Bregman Sparse PCA

```
1: Select parameters $\lambda_1$, $\lambda_2$ and $\mu$
2: Set $V = zeros(N, n)$
3: for i = 1:N do
4:     Initialize $v_0, d_0, b_0$ and set $k = 0$
5:     while $\|v_k - v_{k-1}\| \geq \delta_v$ do
6:         $v_{k+1} = \underset{v}{\text{minimize}} \frac{\lambda_1}{2} \left\|X - \sigma^i u^i v^T\right\|_F^2 + \frac{\mu}{2} \|v\|_2^2 + \frac{\lambda_2}{2} \|d_k - v - b_k\|_2^2$
7:         $d_{k+1} = \underset{d}{\text{minimize}} \|d\|_1 + \frac{\lambda_2}{2} \|d - v_k - b_k\|_2^2$
8:         $b_{k+1} = b_k + v_{k+1} - d_{k+1}$
9:         $k = k + 1$
10:    end while
11:    $v = \frac{v}{norm(v,2)}$
12:    $V(:, i) = v$
13:    $X = X^T(I - vv^T)$.
14: end for
```

3.2 Numerical Example

As a numerical example, a 10×10 matrix with uniform random values from the interval $[0, .5]$ was constructed. Two vectors v_1 and v_2 that had the first five entries equal to one, and the last five entries equal to one respectively were inserted into the second and seventh column of the matrix. The result is seen in figure 3.1. A sparse PCA algorithm will hopefully identify columns 2 and 7 as the variables of interest, in spite of the noise. As is seen in Figure 3.2, the Bregman sparse PCA algorithm does in fact identify these two variables. Note specifically, that all values other than those of the second and seventh position are equal to zero. When compared to the PC found by the traditional SVD approach, it is clear that the sparse PCA has set a large number of inconsequential variables to zero. In the next section, the benefit of enforcing sparsity on the left singular vectors as well will be shown. This comes from the fact that the sparse PCs are still working with dense basis vectors. Specifically, with such a high level of noise, the first singular vector is devoted to capturing this mean level of noise, see figure 3.3.

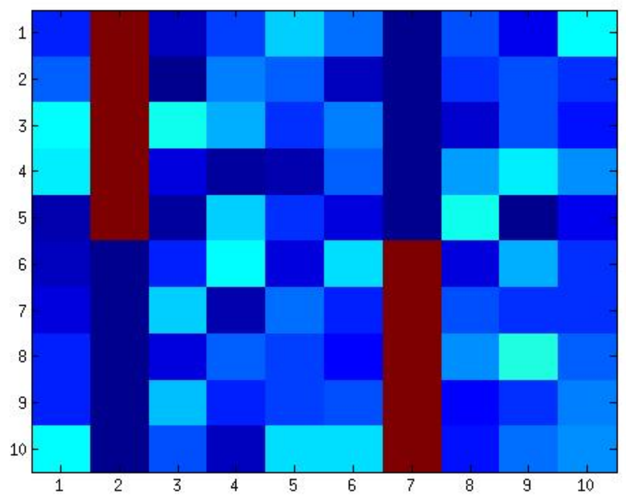

Figure 3.1: The original data matrix X

3.3 Sparse PCA Application: Face Data

In this section, we focus on the ability of the sparse PCA algorithm to select features/variables of interest in a real-world setting. For this example, we will use images of faces and analyze the selection of faces made by the Sparse PCA algorithm.

Data Description and Experiments

The data used consists of 98 different face images of dimension 1440×1080. These images are turned into vectors of size 1555200×1 and used to form a data matrix of size $1555200 \times$

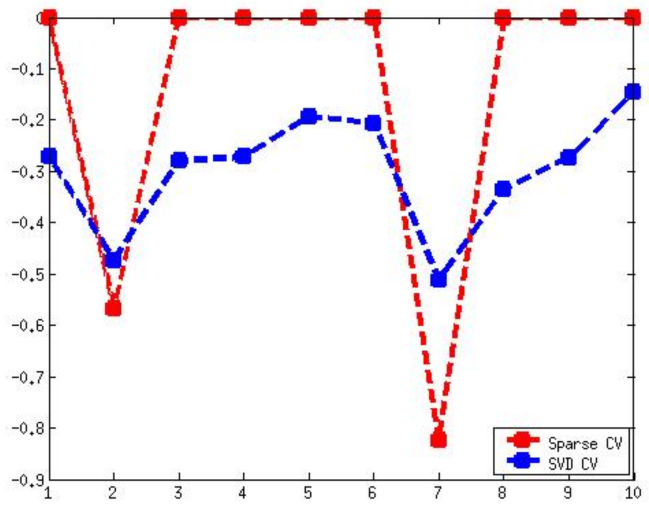

Figure 3.2: The first sparse loading vector and the first PCA loading vector.

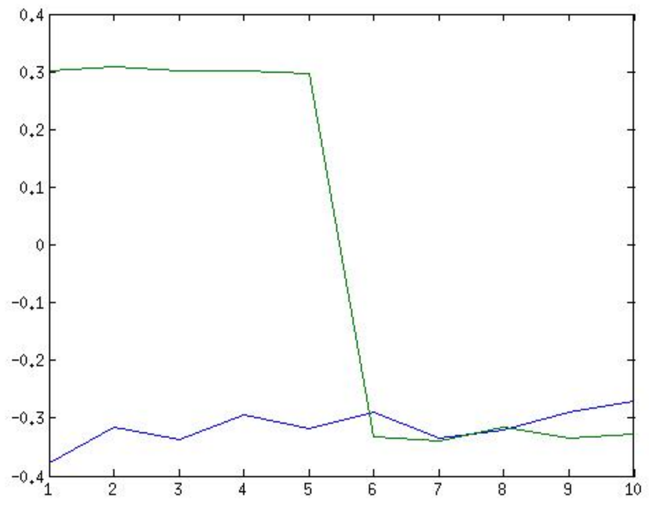

Figure 3.3: The basis vectors as determined by the SVD of X. Note that the first basis vector (in blue) is devoted to capturing the high level of noise in the data.

98. For these examples, in order to center the data, it is mean subtracted prior to any analysis.

The first experiment is a sparse PCA of the data matrix. The goal is to find the sparsest loading vectors (LV)/right singular vectors as possible. The first five loading vectors are found via the sparse PCA algorithm. In order to obtain the sparsest vectors, it was necessary to change the parameters for each subsequent vector. Explicitly, λ_1 changed for each sparse loading vector that was found. The ideal parameter was chosen to maximize sparsity, i.e., the algorithm was repeated until it either found one face or could not be forced to be more sparse. The associated eigenfaces for each of the sparse loading vectors was found via the following calculation, where X is the original, mean subtracted data matrix, $\tilde{V}$ is the matrix with the sparse loading vectors as its columns, and $\tilde{U}$ is the matrix with the calculated sparse eigenfaces as its columns.

$$X\tilde{V} = \tilde{U} \tag{3.21}$$

These eigenfaces will be compared to those calculated using the traditional PCA, which are the left singular vectors found during the calculation of the SVD of the data matrix X.

The first figure shows the first five loading vectors from the traditional PCA and the sparse PCA algorithms. Note that there is only one spike in each sparse loading vector, indicating only one non-zero entry and thus, a unique face chosen as an eigenface when the above formula is implemented. The following five figures show the difference between the traditional PCA based eigenfaces and the sparse PCA based eigenfaces.

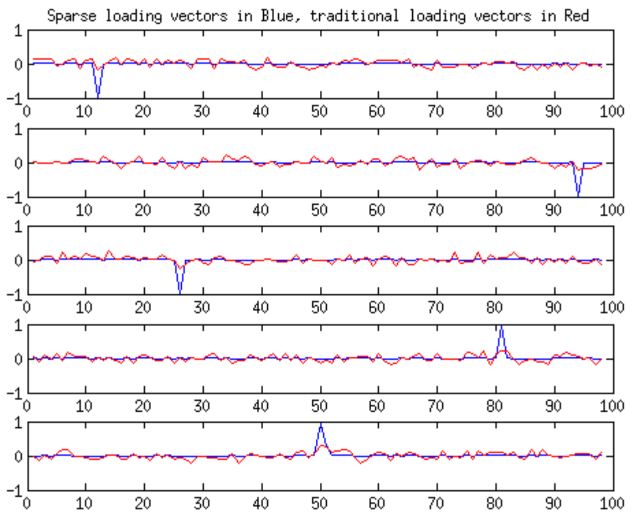

Figure 3.4: The first five loading vectors

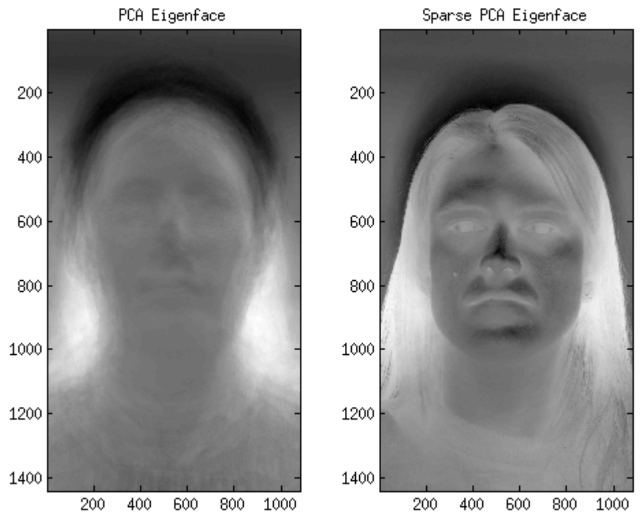

Figure 3.5: The first sparse PCA and traditional PCA based eigenface

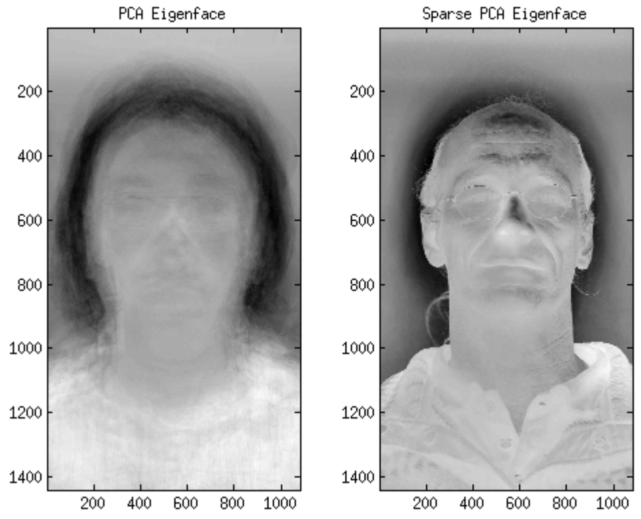

Figure 3.6: The second sparse PCA and traditional PCA based eigenface

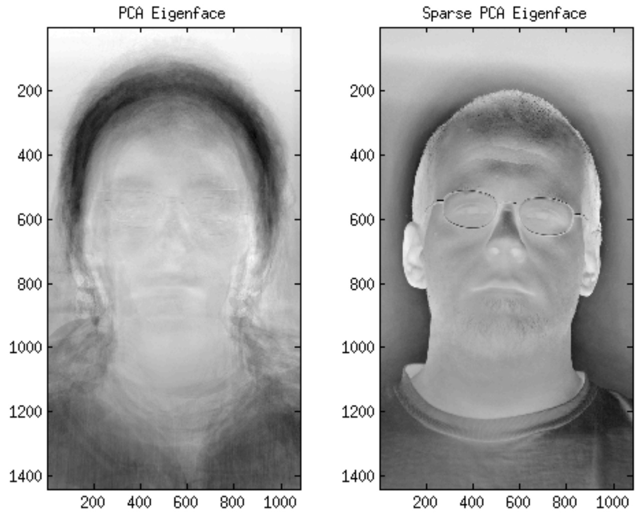

Figure 3.7: The third sparse PCA and traditional PCA based eigenface

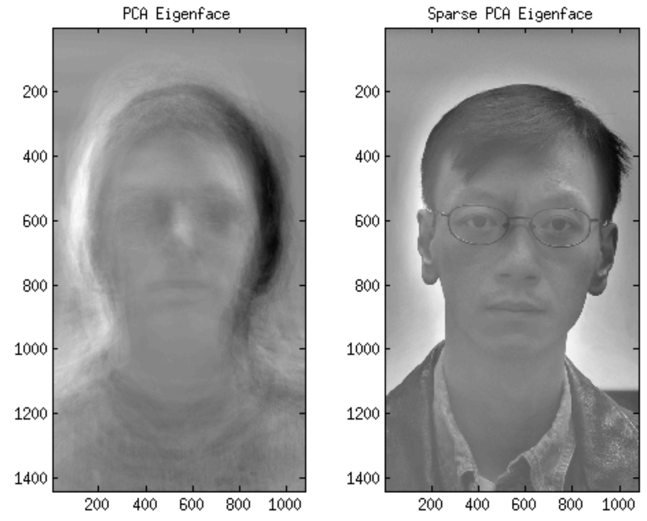

Figure 3.8: The fourth sparse PCA and traditional PCA based eigenface

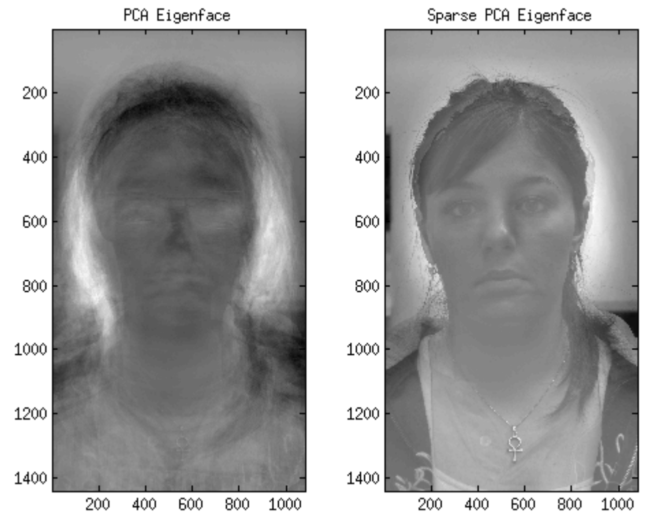

Figure 3.9: The fifth sparse PCA and traditional PCA based eigenface

From Figures 3.5- 3.9, it is clear that the sparse PCA based loading vectors are choosing unique faces, rather than combinations of faces. However, it is unclear what this means in a geometric sense. The images lie in a dimension much too high to visualize, thus, the algorithm Laplacian Eigenmaps or (Lmaps) was used to map the images to a lower dimensional manifold [8]. Laplacian Eigenmaps was chosen in place of a distance preserving mapping such as ISOMAP [81] for its emphasis on local distances and the natural link to spectral clustering that exists via its use of the Laplacian Matrix. The results are in Figure 3.10. As can be seen, the sparse eigenfaces chosen by the sparse PCA algorithm each correspond to one of the three clusters seen in the figure. While it is tempting to append the three eigenfaces chosen by the traditional PCA approach to the data and then do the low dimensional embedding, this is ill-advised. Since these eigenfaces are aggregates of actual faces, they lie perhaps not on the face manifold itself and thus an embedding including these eigenfaces would be artificially skewed.

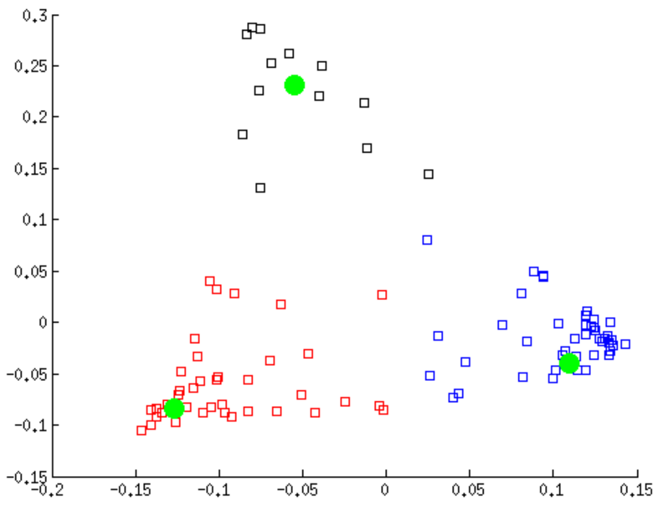

Figure 3.10: The graph associated with the 2D Laplacian Eigenmaps embedding. Note the three clusters and the green dots representing the embedding of the first three sparse Eigenfaces.

3.4 Simultaneous Rank K Approximations

In an attempt to speed up the calculations, simultaneous rank K approximations were attempted. Meaning that instead of solving for one sparse loading vector at a time, K vectors are solved for simultaneously. This changes the given algorithm for sparse PCA in many ways. For one, instead of inputting a vector and getting a vector out, we input a matrix with K appropriately sized vectors as its columns. Also, the method of complimentary projection to enforce orthogonality is no longer valid. Instead, a QR factorization was put in place during each iteration to enforce orthogonality between the columns.

If we let our data matrix X be $m \times n$ with an SVD of $X = USV^T$, then the algorithm used to calculate a rank K approximation can be detailed as follows.

Bregman Sparse PCA Simultaneous Rank K

```
1: Select parameters $\lambda_1$, $\lambda_2$ and $\mu$
2: Set $v_0 = zeros(n, K)$
3: Initialize $d_0, b_0$
4: **while** $\|v_k - v_{k-1}\|_F \geq \delta_v$ **do**
5:     $v_{k+1} = (\lambda_1 * S(1:K, 1:K) * X^T * U(:, 1:K) + \lambda_2 * (d_k - b_k))./(ss * ones(size(v_k)) + \lambda_2 * ones(size(v_k)))$
6:     $v_{k+1} = QR(v_{k+1})$
7:     $d_{k+1} = sign(b_k + v_{k+1}) * max(abs(b_k + v_{k+1}) - (eta/lam2), 0)$
8:     $b_{k+1} = b_k + v_{k+1} - d_{k+1}$
9:     $k = k + 1$
10: **end while**
```

Using the face data, a simultaneous rank 5 solution was achieved in 50 iterations and approximately one minute. The same data was analyzed using subsequent solutions (the original Sparse PCA) in approximately 175 seconds. The solutions were found in 13 - 23 iterations of the algorithm. However, for the simultaneous rank K approach, strict convergence criteria was not satisfied and the vectors returned were not as sparse as those returned using subsequent solutions. The solutions returned after 50 iterations were consistent, meaning it returned the same solution when the experiment was run 10 times. Also, the eigenfaces returned did not correspond to cluster centers as clearly as those returned when using subsequent solutions. This could be due to the lack of control presented by using the QR algorithm to enforce orthogonality. While the QR algorithm can be removed, since its main use is to enforce orthogonality, it does result in significant overlap of the calculated loading vectors up to and including duplication. Since the speed gains are attractive, future work in this area should focus on convergence results and control while enforcing orthogonality.

3.5 Summary

In this chapter the Split Bregman algorithm was applied to the Sparse Principal Component Analysis problem. We formulated an optimization problem, provided numerical examples as to the benefits of a sparse approach to PCA, applied the technique to images of faces and explored the geometric meaning of the Eigenfaces chosen by the algorithm.

Novel contributions in this section include the optimization problem chosen to produce the Sparse PCA which combines the ℓ_1-norm penalty as well as a ridge regression penalty and holds the PCs (left singular vectors) static during calculation of the sparse loading vectors. We also applied the Split Bregman framework to this problem, which gives a proof of convergence since the objective functions meet the assumptions of the proofs given in Chapter Two and also gives the option for further refinement of the objective function through the addition of penalty terms as needed. Finally, the technique was applied to images of faces and the connection to cluster centers was explored via the Laplacian Eigenmap embedding of the data.

4 The Bisparse SVD

Given the link between the SVD and PCA, it is only natural to extend the notion of sparsity to both sets of singular vectors and not just the loading coefficients. This gives the effect of feature selection not only in the variable space, but also in the observation space. Enforcing sparsity in both the variable and observation spaces is useful as it will naturally identify correlations between variables and observations and set to zero variables and observations that do not have a significant impact on the overall variance explained by a pair of sparsity constrained singular vectors. However, the literature here is much more limited.

In a 2008 PhD thesis, Elena Parkhomenko calculates a sparse SVD in the process of calculating a sparse canonical correlation analysis (SCCA) [65]. While never giving an explicit optimization problem, her algorithm combines soft-thresholding with a power method to converge to sparse left and right singular vectors. The vectors are calculated using an alternating approach, meaning v is fixed while u is calculated and then the updated u is used to solve for the next iterate of v. The algorithm is given below for a correlation matrix K.

Sparse SVD for Sparse CCA Algorithm

```
1:  Select sparseness parameters $\lambda_u$ and $\lambda_v$
2:  Initialize $U_0$ and $V_0$ and set $i = 0$
3:  **while** $\|u^i - u^{i-1}\| \geq \delta_u$ and $\|v^i - v^{i-1}\| \geq \delta_v$ **do**
4:      $u^{i+1} = Kv^i$
5:      Normalize: $u^{i+1} = \frac{u^{i+1}}{\|u^{i+1}\|}$
6:      Apply soft-thresholding: $u^{i+1} = sign(u^{i+1})(|u^{i+1}| - \frac{1}{2}\lambda_u)_+$
7:      Normalize: $u^{i+1} = \frac{u^{i+1}}{\|u^{i+1}\|}$
8:      $v^{i+1} = Ku^{i+1}$
9:      Normalize: $v^{i+1} = \frac{v^{i+1}}{\|v^{i+1}\|}$
10:     Apply soft-thresholding: $v^{i+1} = sign(v^{i+1})(|u^{i+1}| - \frac{1}{2}\lambda_v)_+$
```

```
11:     Normalize: $v^{i+1} = \frac{v^{i+1}}{\|v^{i+1}\|}$
12:     $i = i + 1$
13: end while
```

In this algorithm, the soft-thresholding operator, denoted by $(\cdot)_+$ is identical to the thresholding operation defined in Chapter 2 of this paper, namely, the thresholding used in the Split Bregman algorithm. Convergence is unclear in the case when sparsity constraints are added and her method is generally used for rank one approximations. There is no clear method with which to add additional terms including additional smoothness and sparsity constraints.

Witten et al. also used a penalized matrix decomposition (PMD) that is in fact a sparse version of the SVD in their 2009 paper for the purpose of sparse canonical correlation analysis and sparse principal components [93]. The rank one optimization problem they solve is given by

$$\begin{aligned} &\underset{u,v}{\text{maximize}}\ u^T X v \\ &\text{subject to}\ \ \|u\|_2^2 \leq 1, \|v\|_2^2 \leq 1, P_1(u) \leq c_1, P_2(v) \leq c_2 \end{aligned} \tag{4.1}$$

The penalty functions P_1 and P_2 that are suggested are the well known LASSO penalty [82] and the fused lasso penalty, which enforces smoothness in the produced singular vectors [83]. The algorithm they use to solve this problem again is primarily used for a rank one approximation and as with the algorithm of [65] it alternates between u and v.

PMD for Sparse SVD and CCA

```
1: Initialize $v$ to have $\ell_2$-norm 1.
2: Iterate until convergence
3:     $u = \text{argmax}_u u^T X v$ subject to $P_1(u) \leq c_1$ and $\|u\|_2^2 = 1$
4:     $v = \text{argmax}_v u^T X v$ subject to $P_2(v) \leq c_2$ and $\|v\|_2^2 = 1$
```

```
5: $d = u^T X v$
```

Steps 3 and 4 are carried out using soft thresholding. It is unclear whether or not the algorithm they use to solve this optimization problem is flexible enough to accept other penalty terms. Finally, their algorithm is not guaranteed to find a global optimum. The solutions are generally interpretable, but convergence to an optimal solution is not guaranteed.

The first directly named sparse singular value decomposition (SSVD) came from Lee et al. in 2010 [51]. They again use an iterative thresholding approach to solve a rank one problem, employing deflation to solve for subsequent approximations. For their sparsity penalization they use the adaptive lasso, which uses different weightings to threshold each entry of a singular vector [100]. In contrast to the approach taken by Witten et al. [93], the following algorithm minimizes Frobenius norm error with a penalty term for sparsity rather than maximizing variance. The optimization problem is given below, followed by the algorithm. In the optimization problem, note that they are minimizing a rank one approximation to $\mathbf{X}$ via the constant s, and the vectors $\mathbf{u}, \mathbf{v}$, while enforcing sparsity using an adaptive LASSO penalty on both $\mathbf{u}$ and $\mathbf{v}$. The two λ parameters stay fixed while the two sets of w parameters are allowed to change for each entry.

$$\underset{u}{\text{minimize}} \;\; \left\|X - suv^T\right\|_F^2 + s\lambda_u \sum_{i=1}^{n} w_{1,i} u_i + s\lambda_u \sum_{j=1}^{d} w_{2,j} |v_j| \tag{4.2}$$

Sparse SVD Algorithm

```
1: Step 1: Apply the standard SVD to $X$. Let $\{s_{old}, u_{old}, v_{old}\}$ denote the first SVD triplet.
2: Step 2: Update:
3:         $v_{new} = X^T u_{old}$
4:         Perform component-wise soft thresholding on $v_{new}$ using the
           parameters $\lambda_v$ and $w_2$. Normalize the new $v_{new}$ to have unit norm.
```

```
5:        $u_{new} = Xv_{new}$
6:        Perform component-wise soft thresholding on $u_{new}$ using the
          parameters $\lambda_u$ and $w_1$. Normalize the new $u_{new}$ to have unit norm.
7:        set $u_{old} = u_{new}$ and repeat the step 2 until convergence.
8:  Step 3: Set $u = u_{new}$, $v = v_{new}$, $s = u^T Xv$ at convergence.
```

The algorithm does not appear to adapt easily to different penalty functions. In 2011, Sill et al. extended the algorithm of [51] by incorporating the stability selection criteria of [56], [76].

Finally, in 2011 Allen et al. introduced a Generalized Least Squares Matrix Decomposition [3]. The method equates to sphering the data relative to the covariance structure found in both the rows and the columns, and taking the SVD of the resulting data. Their method of optimization allows for non-iid noise assumptions in the matrix decomposition. Namely, they assume that the column and row noise need not be iid Gaussian, but that they are separable. The method does need to approximate this noise structure and they supply guidance and logic for doing so. In their paper, they not only search for sparsity, but they also give ways to enforce smoothness in the singular vectors. The method will allow penalty terms that are convex and homogeneous of order one, or that are convex and satisfy $P(cx) = cP(x)$. The algorithm again uses soft-thresholding using the given penalty functions.

4.1 Bisparse SVD

Like the majority of the prior references, to formulate the Bisparse SVD (BSSVD) we will use the ℓ_1-norm to induce sparsity in our solutions. We begin with a rank one decomposition for the data matrix X. We propose to solve for u and v that

$$\underset{u,v}{\text{minimize}} \ \|u\|_1 + \|v\|_1 + \frac{\lambda_1}{2}\left\|X - uv^T\right\|_F^2 + \frac{\mu}{2}\|u\|_2^2 + \frac{\nu}{2}\|v\|_2^2 \tag{4.3}$$

where $X \in \mathbb{R}^{m\times n}$ matrix which has been mean subtracted and the columns (variables) have been normalized to have unit variance, and F indicates the Frobenius norm. The two vectors u and v are of size $m \times 1$ and $n \times 1$ respectively. Therefore, in the above equation, we have the classic Frobenius norm minimization that would lead to the SVD, however, we have added two penalty parameters, namely $\|u\|_1$ and $\|v\|_1$ to enforce sparsity on our solution. Also note that the norm of the solution is also kept from trivially iterating to the zero vector by the Frobenius penalty term. If both u and v are zero, and λ_1 is chosen sufficiently large, then this is a less optimal solution than nonzero vectors would give.

In the Split Bregman framework, the optimization problem above would be given by

$$\underset{u,v,d^u,d^v}{\text{minimize}} \; \|d^u\|_1 + \|d^v\|_1 + \frac{\lambda_1}{2}\left\|X - uv^T\right\|_F^2 + \frac{\mu}{2}\|u\|_2^2 + \frac{\nu}{2}\|v\|_2^2 \tag{4.4}$$

$$+\frac{\lambda_2}{2}\|d^u - u\|_2^2 + \frac{\lambda_3}{2}\|d^v - v\|_2^2 \tag{4.5}$$

For a fixed u, note that the above optimization problem, Equation (4.5), is equivalent to the sparse PCA formulation. Since the algorithm alternately optimizes u and then v while holding the other parameters fixed, the iterations for u and v can be modeled after those found for the sparse PCA algorithm. Therefore, Equation (4.5) would have iterations defined by

$$\begin{aligned}
u_{k+1} &= \frac{\lambda_1 X v_k + \lambda_2(d_k^u - b_k^u)}{\lambda_1 v_k^T v_k + \lambda_2 + \mu} \\
v_{k+1} &= \frac{\lambda_1 X^T u_{k+1} + \lambda_3(d_k^v - b_k^v)}{\lambda_1 u_{k+1}^T u_{k+1} + \lambda_3 + \nu} \\
d_{k+1}^u &= \text{shrink}(u_{k+1} + b_k^u, \lambda_2) \\
d_{k+1}^v &= \text{shrink}(v_{k+1} + b_k^v, \lambda_3) \\
b_{k+1}^u &= b_k^u + u_{k+1} - d_{k+1}^u \\
b_{k+1}^v &= b_k^v + v_{k+1} - d_{k+1}^v
\end{aligned} \tag{4.6}$$

The majority of the operations for this algorithm are found in the matrix multiplications, which will be $2 * O(m \cdot n)$. This is performed in each loop of this algorithm, until convergence, and is performed for each set of vectors to be found. Thus, if h is the number of pairs of singular vectors to be found, and j_k is the number of iterations it takes to get to convergence for the kth pair of sparse singular vectors, then the operations would be on the order of $2 * O(h \cdot j_k \cdot m \cdot n)$. As a comparison, the largest term in an SVD decomposition for a complete set of n vectors, assuming $n \leq m$ would be of order $O(m \cdot n^2)$.

When solving for subsequent pairs of singular vectors, we enforce orthogonality by looking at the complementary projection of X onto both $u_1, \ldots, u_{k-1}$ and $v_1, \ldots, v_{k-1}$ where u_i and v_i are the ith sparse left and right singular vectors respectively. For example, after solving for the first pair, the data matrix used to solve for the second pair of vectors would be $\hat{X}$ from below, computed as

$$\begin{aligned} \tilde{X} &= \left(I - u_1 u_1^T\right) X \\ \hat{X} &= \tilde{X}^T \left(I - v_1 v_1^T\right). \end{aligned} \tag{4.7}$$

To illustrate the effectiveness of the algorithm, we use the same matrix structure as with sparse PCA, see Figure 3.1. The goal here is two-fold. First, we would like to show the benefit of enforcing sparsity on the basis vectors as well as the loading vectors (left and right singular vectors). Second, when compared to the normal SVD, we will show the benefits of our approach.

Tom demonstrate this, we propose three experiments with increasing levels of noise, and in the final experiment higher dimensions as well. In the first example we begin with a smaller level of noise than in the PCA example, the noise is within the interval of $[0, .1]$. This will allow us to increase the noise in the next experiment and show the increasing difference between the sparse singular vectors and the traditional singular vectors. The two vectors v_1 and v_2 are defined identically as in the PCA example in section 3.2. The original data is shown in Figure 4.1. The rank 2 SVD approximation is shown in Figure 4.2. Note that the noise is beginning to be captured in the approximation. As expected, the BSSVD version of the rank 2 approximation does not contain any noise and is shown in Figure 4.3

To understand why the BSSVD takes the noise out of the approximation, it helps to compare the left singular vectors. Figure 4.4 shows that the first singular vector is again used to capture variance associated with the noise in the data. When compared to the first two BSSVD left singular vectors in Figure 4.5 the benefits of the sparsity constraints is clear. Both vectors are used to capture signal and not noise. Note that the two vectors are complementary in the signal they capture, illustrating the orthogonality via complementary projections.

Another benefit of the sparse left singular vectors is made clear by comparing the right singular vectors. Figure 4.6 shows the right singular vectors from the SVD. It is clear that the second and seventh variables are the important variables to keep. However, the BSSVD vectors in Figure 4.7 are much cleaner due to the sparsity in the solutions.

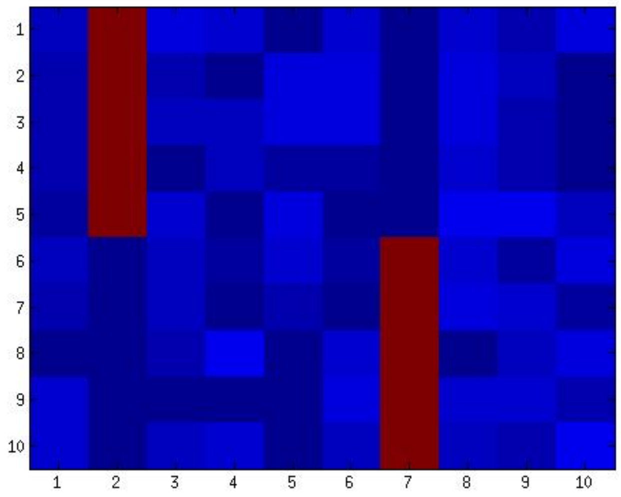

Figure 4.1: The original data matrix X with noise in the interval $[0, .1]$

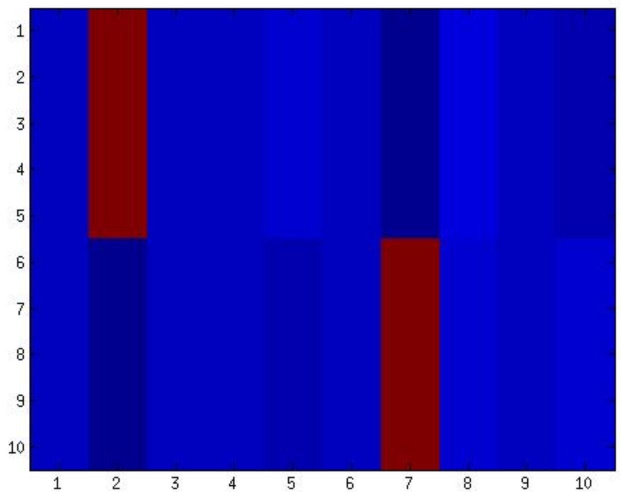

Figure 4.2: The rank 2 SVD approximation to X with noise in the interval $[0, .1]$

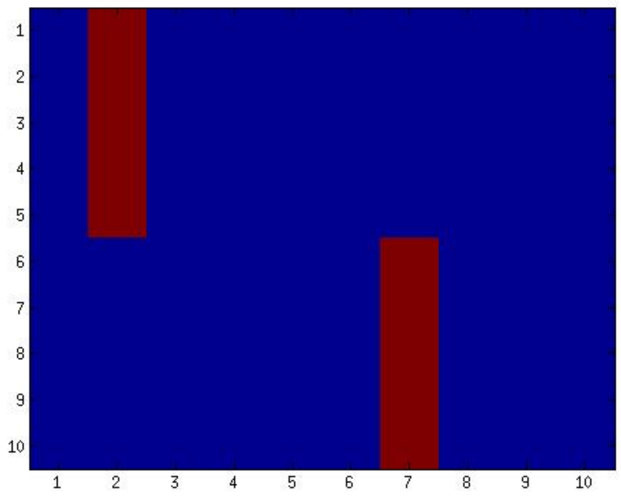

Figure 4.3: The rank 2 BSSVD approximation to X with noise in the interval $[0, .1]$

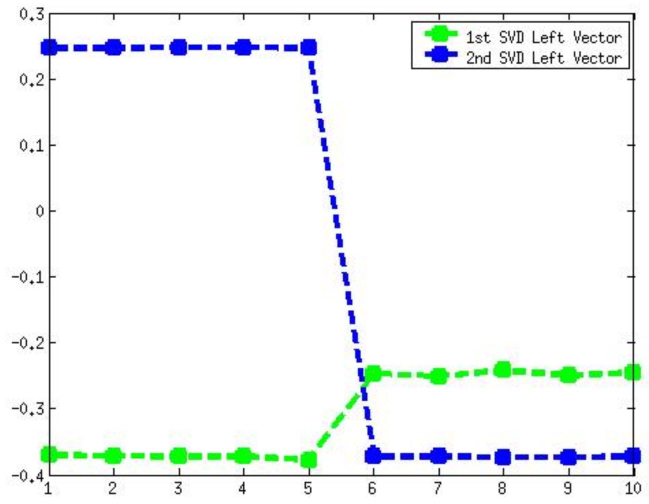

Figure 4.4: The first two left singular vectors from the rank 2 SVD approximation

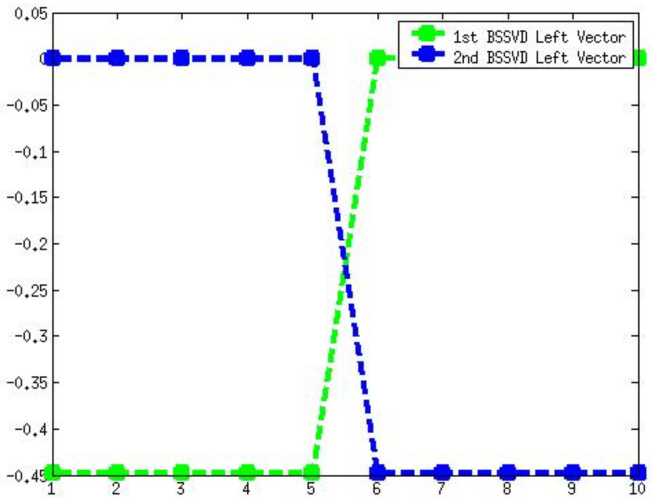

Figure 4.5: The first two left singular vectors from the rank 2 BSSVD approximation

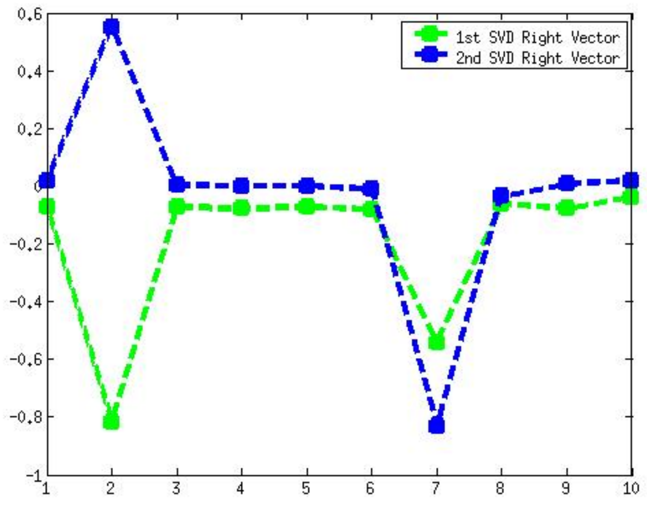

Figure 4.6: The first two right singular vectors from the rank 2 SVD approximation

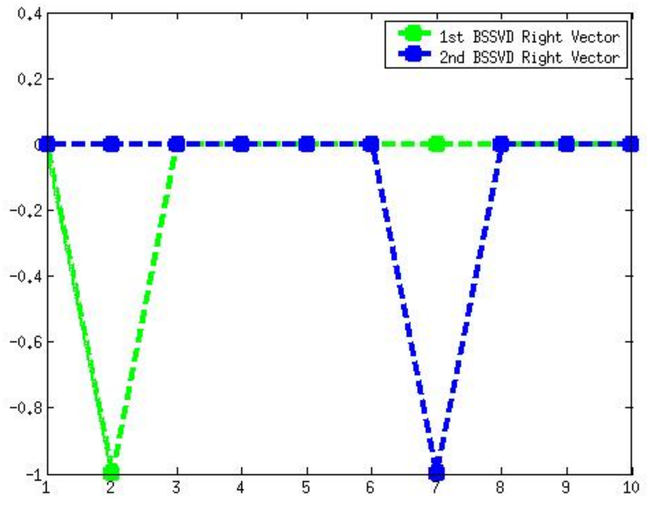

Figure 4.7: The first two right singular vectors from the rank 2 BSSVD approximation

In Figures 4.8 - 4.14 are the same sets of results, however the matrix X has uniform noise from the interval $[0, .3]$ instead of $[0, 0.1]$.

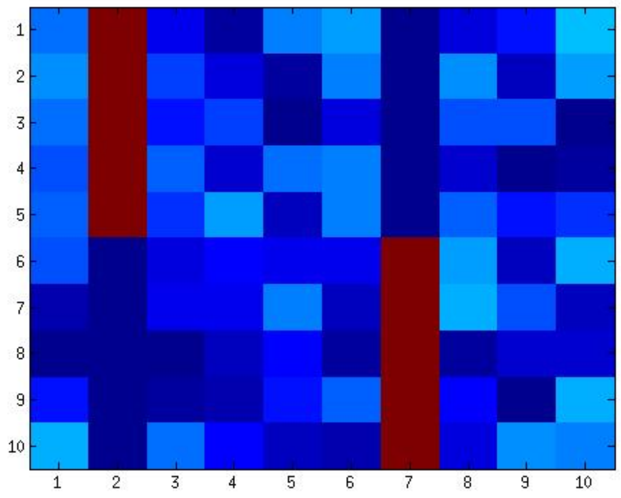

Figure 4.8: The original data matrix X with noise in the interval $[0, .3]$

The next set of figures deals with a much larger data set. This data is 1000×1000 and the noise comes from the interval $[0, 1.3]$. The two vectors are defined similarly, except that the first 500 entries of v_1 are set to 1, and the last 500 entries of v_2 are set to one. They are set to be the $200th$ and $700th$ columns of the data matrix X. This construction of the matrix means that the surrounding noise actually can be greater than the entries in the two inserted vectors. In this scenario, both the BSSVD and the classical SVD identified the noise with the first basis vector. Note, for this experiment only, the data has been mean subtracted prior to running both algorithms. Perhaps not surprisingly, the BSSVD identifies the two columns without noise. However, in this instance, we only use the rank one approximation, as the needed information is captured using only the first left and right singular vectors from the BSSVD.

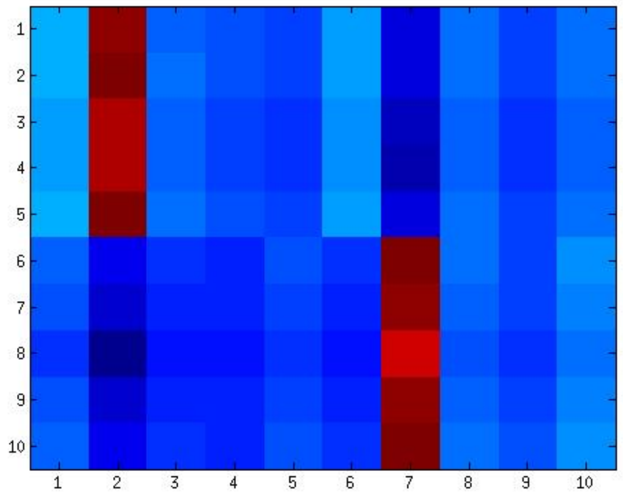

Figure 4.9: The rank 2 SVD approximation to X with noise in the interval $[0, .3]$

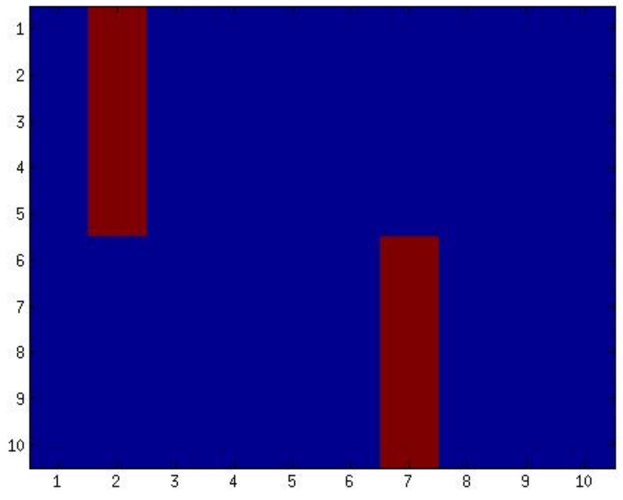

Figure 4.10: The rank 2 BSSVD approximation to X with noise in the interval $[0, .3]$

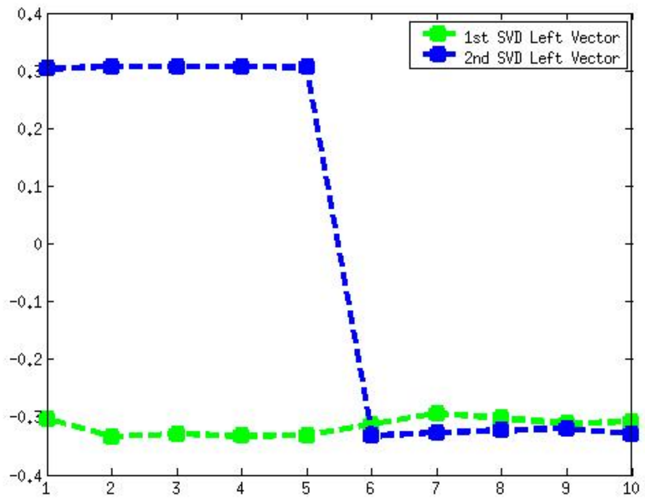

Figure 4.11: The first two left singular vectors from the rank 2 SVD approximation

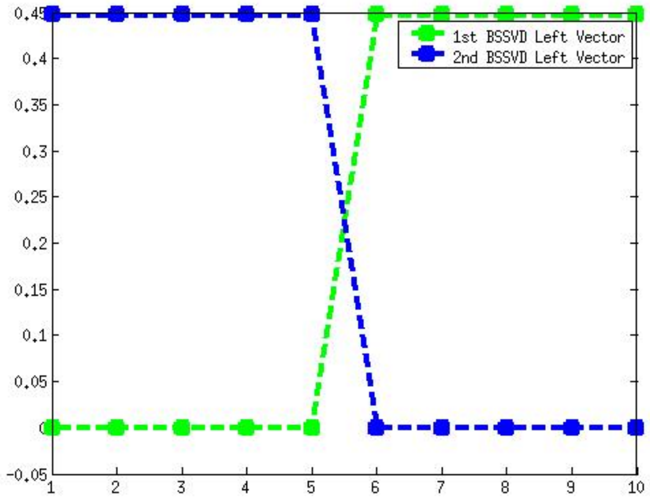

Figure 4.12: The first two left singular vectors from the rank 2 BSSVD approximation

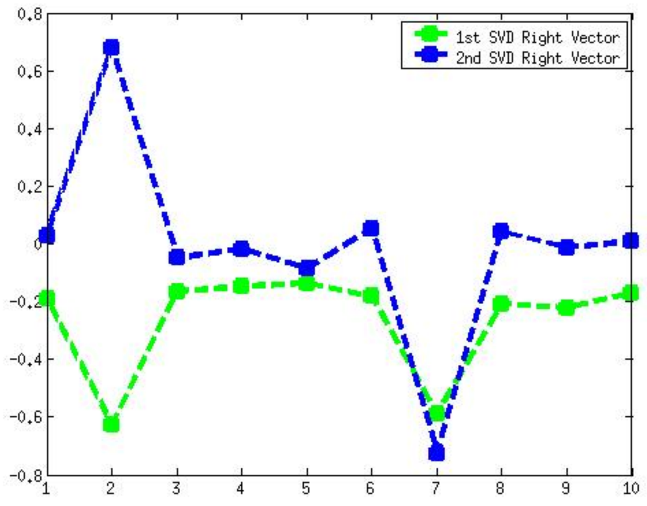

Figure 4.13: The first two right singular vectors from the rank 2 SVD approximation

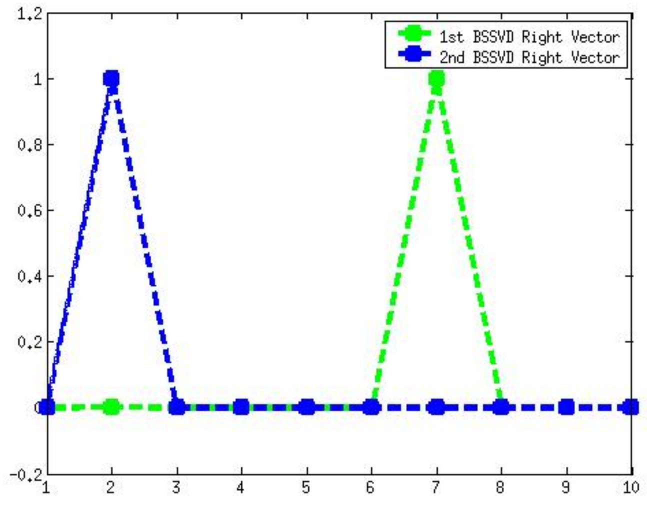

Figure 4.14: The first two right singular vectors from the rank 2 BSSVD approximation

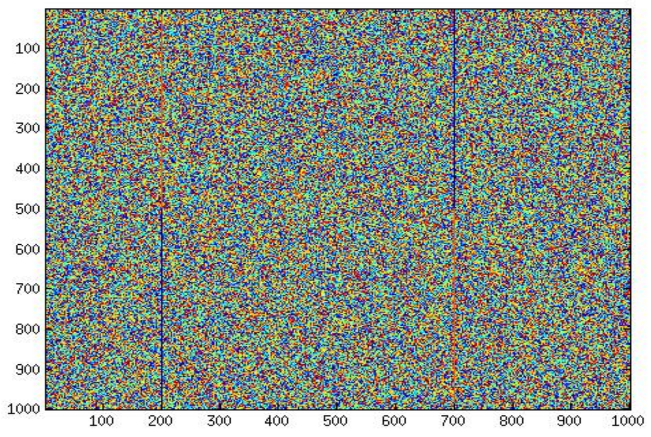

Figure 4.15: The original data matrix X with noise in the interval $[0, 1.3]$

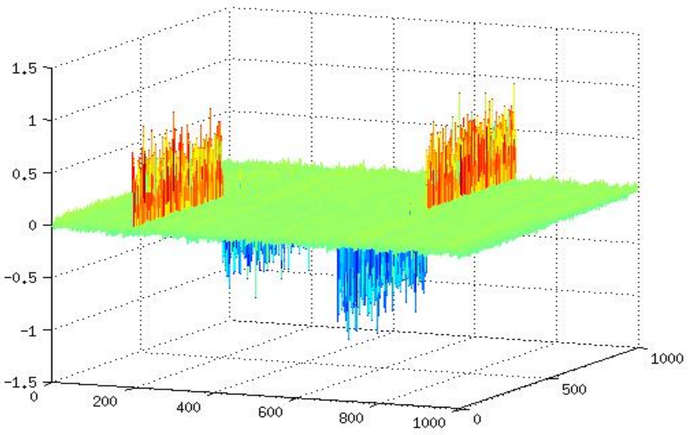

Figure 4.16: The rank 1 SVD approximation to X with noise in the interval $[0, 1.3]$

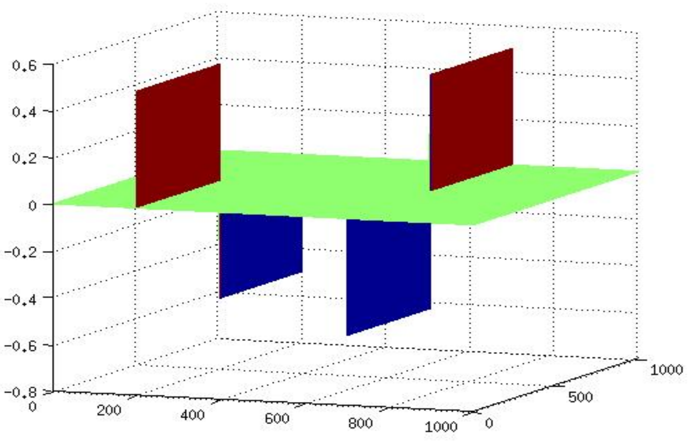

Figure 4.17: The rank 1 BSSVD approximation to X with noise in the interval $[0, 1.3]$

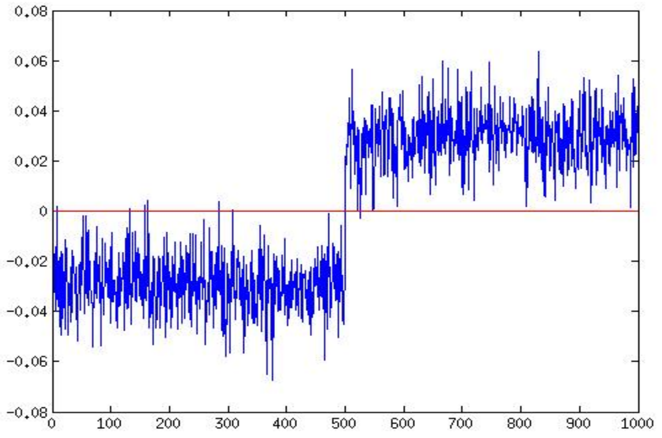

Figure 4.18: The first left singular vector from the rank 1 SVD approximation.

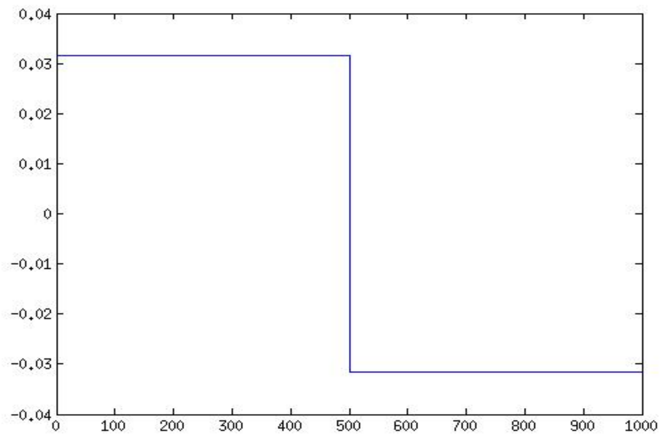

Figure 4.19: The first left singular vector from the rank 1 BSSVD approximation

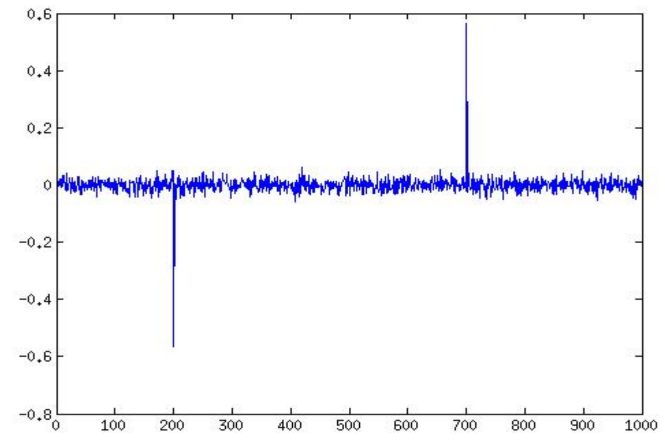

Figure 4.20: The first right singular vector from the rank 1 SVD approximation

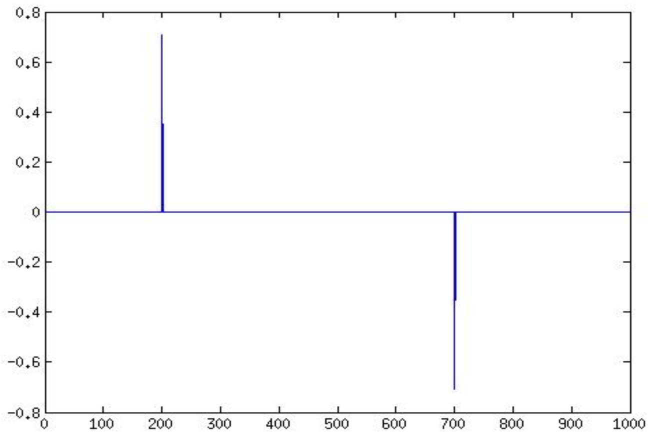
Figure 4.21: The first right singular vector from the rank 1 BSSVD approximation

4.2 Fabry-Perot Hyperspectral Imagery

In order to illustrate the applicability of the BSSVD on real world data we will use Hyperspectral Imagery data obtained via a Fabry-Perot Interferometer[1]. Hyperspectral imagery means that the image is taken over many, nearly contiguous wavelengths. Meaning that the image is a data cube including the two-dimensional classical notion of the image, along with a third dimension indicative of the multiple wavelengths at which the image was taken. The image below taken from [7] shows both the two spatial dimensions and the wavelength dimension.

Hyperspectral imagery has been used for many applications including the mapping of nonnative plants [86], mineral mapping [69], mapping of invasive plants [50], planetary exploration [16, 22] and military applications such as target detection [54]. Generally, applications utilizing hyperspectral imagery are taking advantage of the fact that each material in an image will have a distinct signature along the wavelength dimension. Meaning that each pixel, if it were a pure material, would match to a distinct signal vector indicating the material present. However, in practice most hyperspectral imagers do not capture only

[1]This data was made available through the NSF and Defense Threat Reduction collaboration on Algorithms for Threat Detection

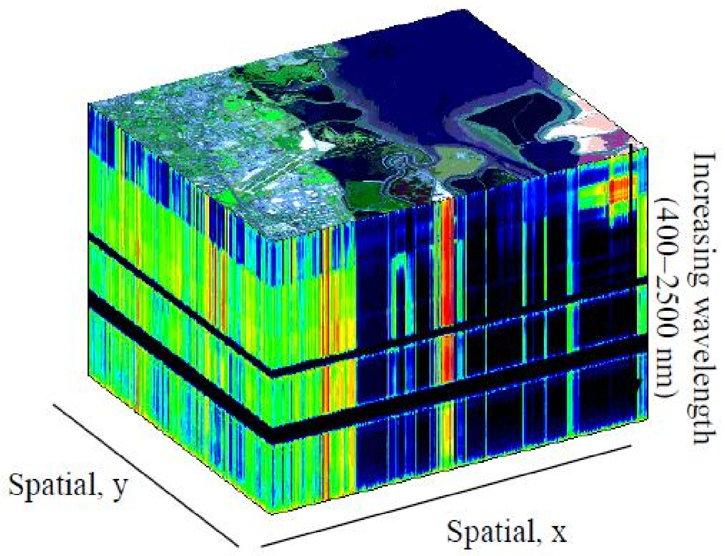

Figure 4.22: Example of hyperspectral image, with two spatial dimensions and one wavelength/color dimension

pure materials within one pixel thus, demixing/unmixing techniques or transformations of the coordinates are needed [20, 10, 92, 80, 4, 5, 6]

The data analyzed for this example are images of size 256×256 taken at 20 different wavelengths. There is also a time component to the data, as the purpose is to track an artificial plume created by detonating certain chemicals into the air and taking hyperspectral images of the plume as it disperses. Thus, there are 561 hyperspectral images of size $256 \times 256 \times 20$ to analyze.

The images are first trimmed in the first dimension to 51, in order to capture the region of most interest in the data. Then, the wavelength is limited to one wavelength that is known to capture the signature of the chemical of interest released in this dataset. In this instance it is Triethyl-phosphate. The data is then reshaped into a dataset where each column is an image, thus the resulting dataset is $51 * 256 \times 561 = 13,056 \times 561$. Given that the first 100 frames are known to not contain a plume, they are used to construct a basis for the background of the image and the data is then projected on the compliment of this data, effectively removing a large portion of the background. However, a large amount of noise remains and the BSSVD is used to remove as much of this noise as is possible. In the figures that follow, we compare the original image, the complementary projected image and the rank 10 BSSVD filtered image for several snapshots in time. Specifically, the 50, 130, 150, 170, 190, 210, 220, 230 and the 250th frames. No frames after 250 were chosen as the plume has generally run its course through the image. Note the colorbar inserted in the first figure. Since it is the 50th frame and part of the basis used for the background removal,the projected data image is entirely zero. However the BSSVD image does not appear to be zero, the colorbar indicates however, that the entries are near zero. Of note is that the plume is not visible in the original image, is obvious in the projected image, and is less noisy in the BSSVD image. In some cases, the BSSVD plume may appear smaller, this is not unexpected, particularly close to detonation since there will be noise in the form of dust and smoke that the BSSVD may filter out.

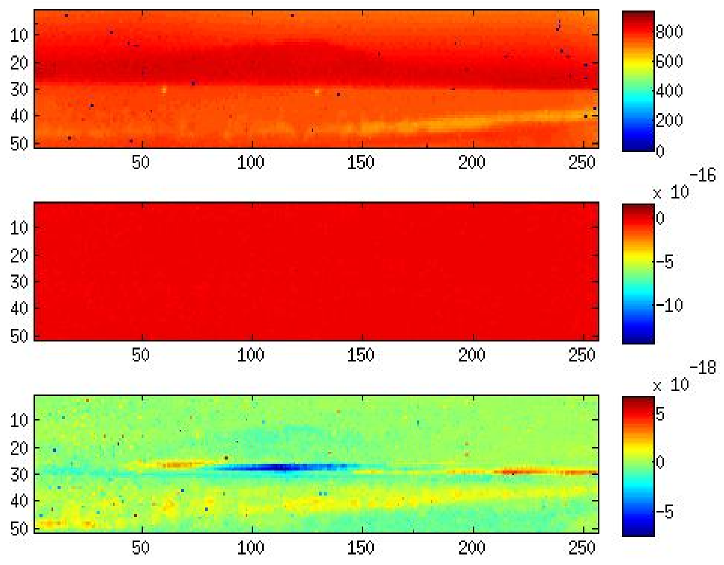

Figure 4.23: The 50th frame of Fabry Perot data. No plume present yet. Original image at top, background removed image in the middle, rank 10 BSSVD approximation at the bottom.

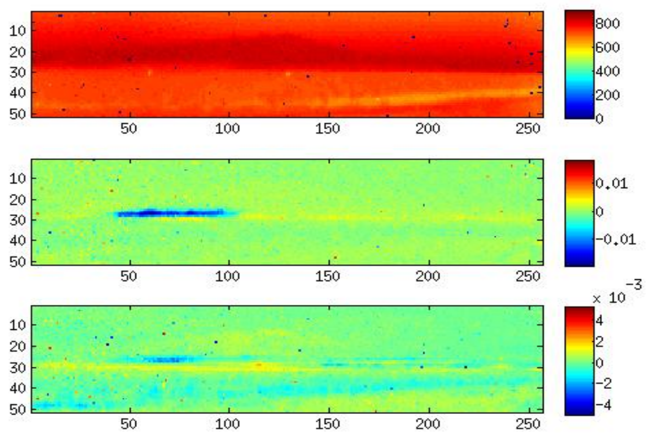

Figure 4.24: The 130th frame of Fabry Perot data. Original image at top, background removed image in the middle, rank 10 BSSVD approximation at the bottom.

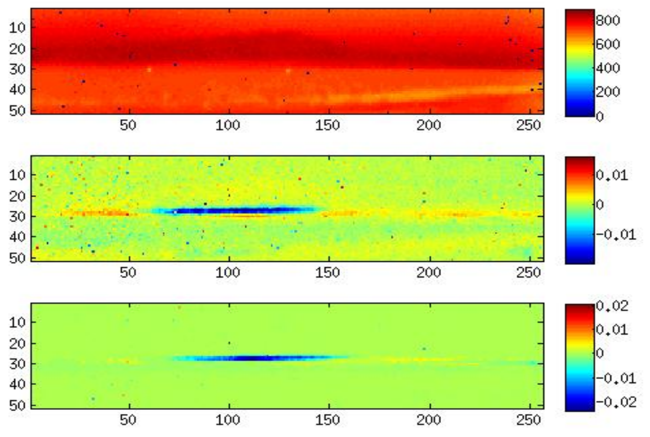

Figure 4.25: The 150th frame of Fabry Perot data. Original image at top, background removed image in the middle, rank 10 BSSVD approximation at the bottom.

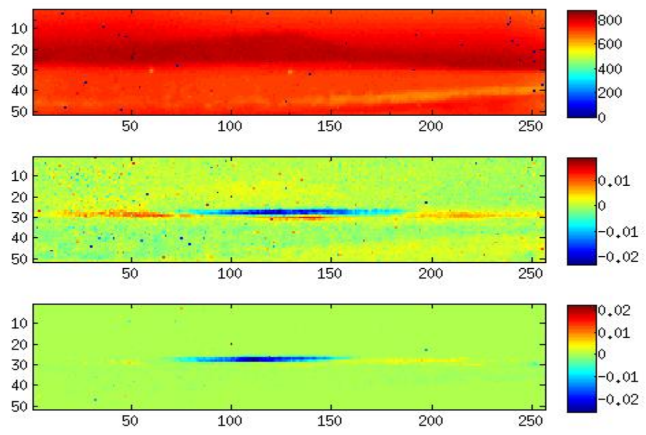

Figure 4.26: The 170th frame of Fabry Perot data. Original image at top, background removed image in the middle, rank 10 BSSVD approximation at the bottom.

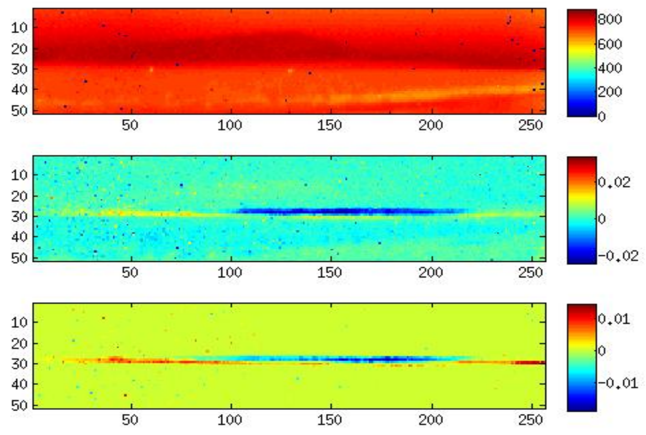

Figure 4.27: The 190th frame of Fabry Perot data. Original image at top, background removed image in the middle, rank 10 BSSVD approximation at the bottom.

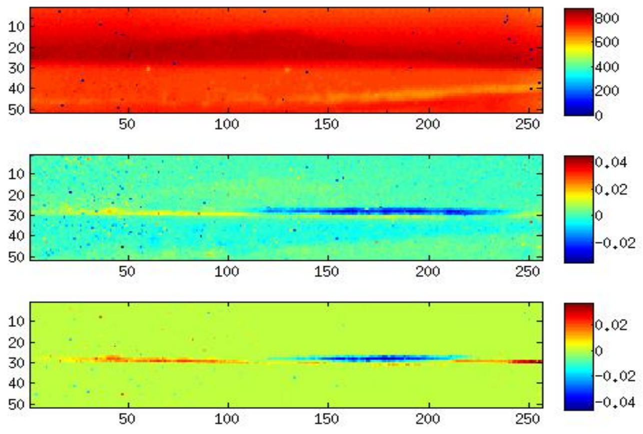

Figure 4.28: The 210th frame of Fabry Perot data. Original image at top, background removed image in the middle, rank 10 BSSVD approximation at the bottom.

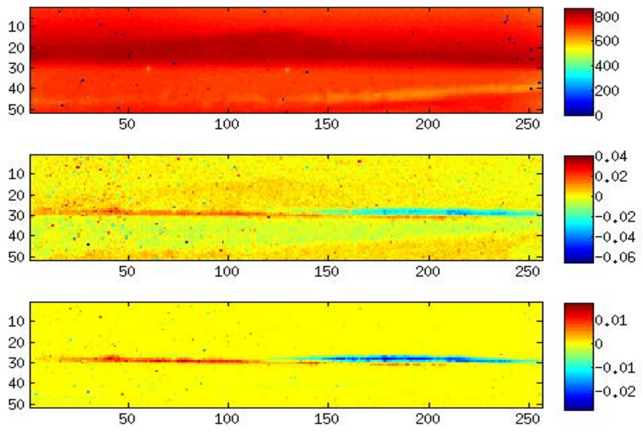

Figure 4.29: The 230th frame of Fabry Perot data. Original image at top, background removed image in the middle, rank 10 BSSVD approximation at the bottom.

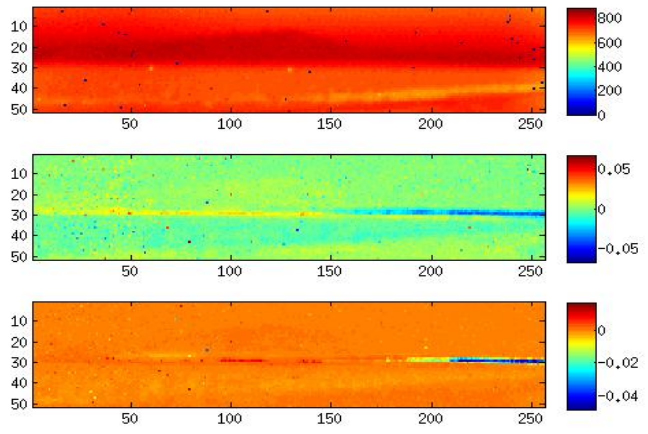

Figure 4.30: The 250th frame of Fabry Perot data. Original image at top, background removed image in the middle, rank 10 BSSVD approximation at the bottom.

To understand why the BSSVD decomposed images are less noisy than the projected images it helps to compare the BSSVD Left singular Vectors and Right Singular Vectors to those obtained via the SVD of the projected data. Below are the first four left and right singular vectors for the BSSVD and the SVD. Note the strong plume found as the fourth left singular vector by the BSSVD.

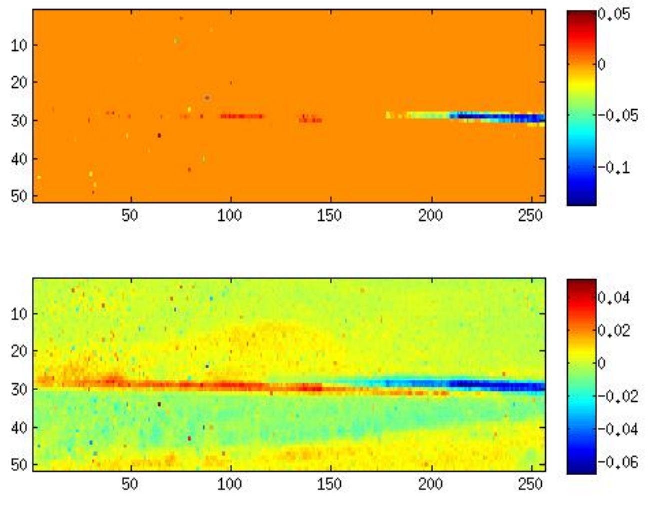

Figure 4.31: The first left vector in image form. BSSVD on top, SVD on bottom

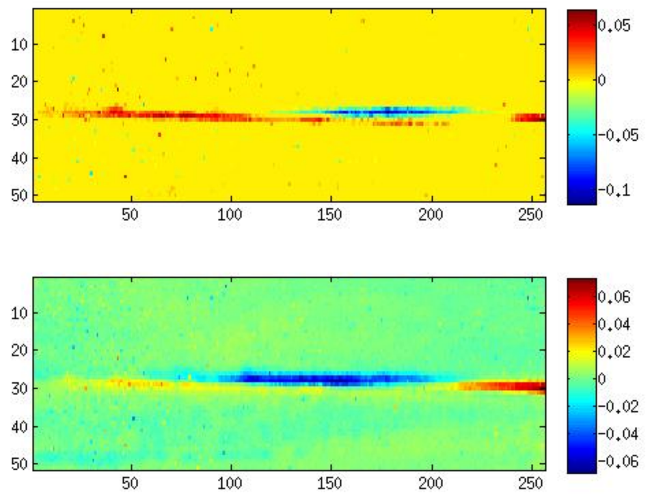

Figure 4.32: The second left vector in image form. BSSVD on top, SVD on bottom

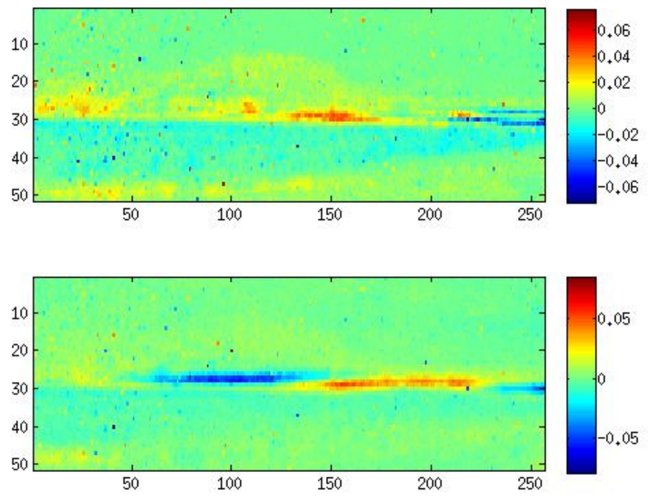

Figure 4.33: The third left vector in image form. BSSVD on top, SVD on bottom

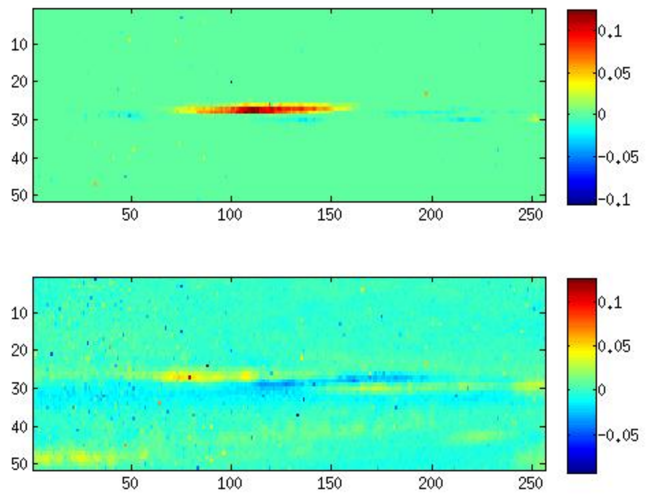
Figure 4.34: The fourth left vector in image form. BSSVD on top, SVD on bottom

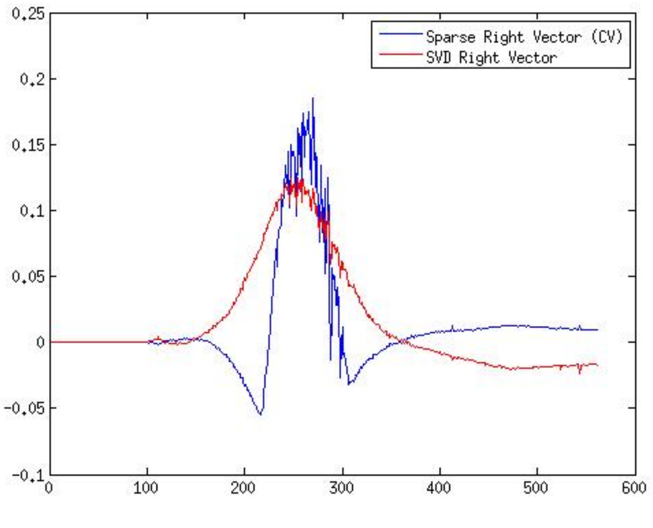

Figure 4.35: The first right vector.

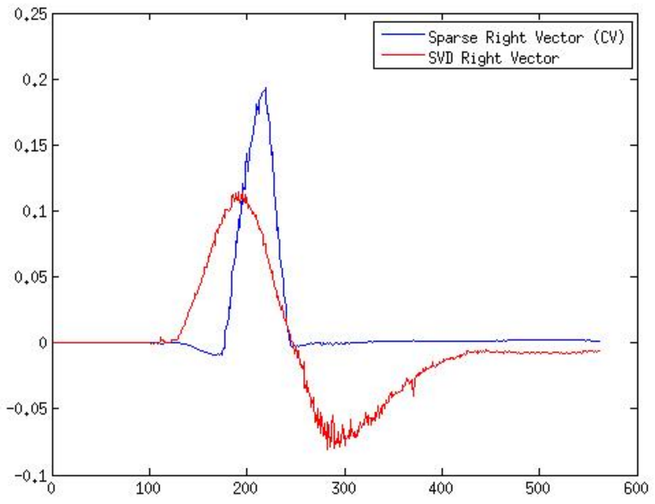

Figure 4.36: The second right vector.

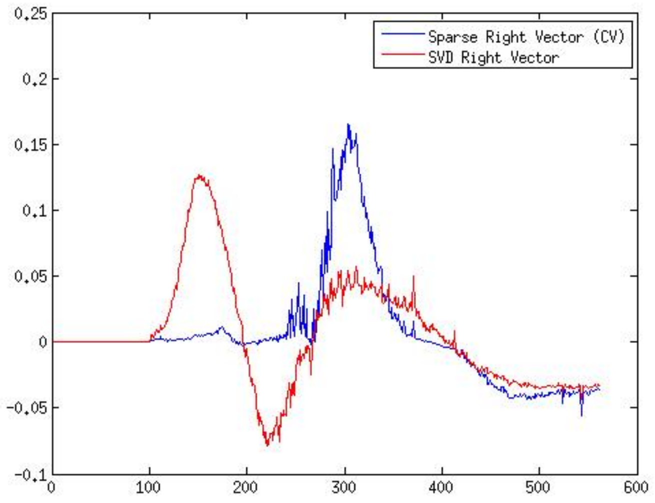

Figure 4.37: The third right vector.

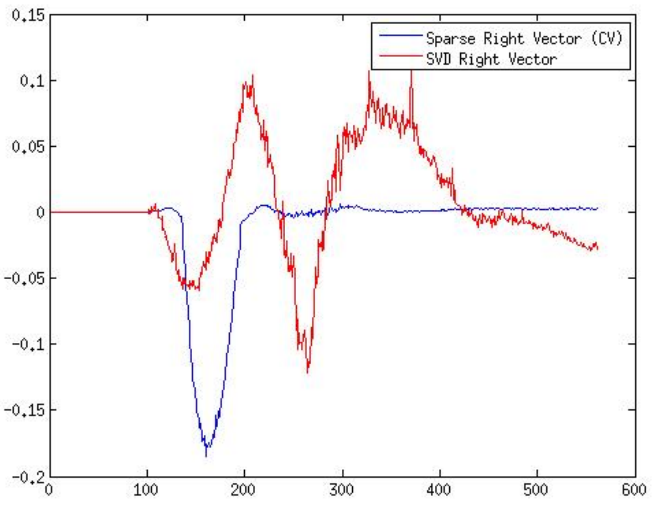

Figure 4.38: The fourth right vector.

4.3 BSSVD with Dynamic Parameters

The ability to change the parameter attached to the ℓ_1 penalty term was explored for the BSSVD algorithm. Specifically, it was used to attempt to only keep the largest magnitude component in each iteration of the algorithm while setting the rest to zero. This was done by explicitly changing the penalty associated with the ℓ_1-norm to be such that the entries in d_{k+1} would be set to zero unless they were of equal or greater magnitude to the largest possible entry. Meaning, that in the shrinkage operator, the thresholding level was manually set to the largest possible value so that all but one entry (or multiple entries with magnitude equal to that computed) of d_{k+1} would be set to zero. While in theory, this seemed like it would work and allow us to obtain only the most important/largest component in each vector, the results were highly mixed. There are some nuances during the updating of the Bregman parameter and the subsequent optimization of several other variables that for now it is unclear how they are reacting to this type of dynamic parameter selection. This is an area of future research, including the possibility of a more continuation like approach, where the algorithm is allowed to converge, or nearly converge and then the parameter is changed in order to focus on a single component.

4.4 Summary

In this chapter the Split Bregman algorithm was applied to the Bisparse Singular Value Decomposition. We formulated an optimization problem, provided numerical examples as to the benefits of a sparse approach to the SVD, extending the idea beyond Sparse PCA and showing why that extension is necessary. We also applied the technique to Hyperspectral Imagery in an attempt to find further information in a sparse component of a hyperspectral image.

Novel contributions in this section include the application of the Split Bregman framework to this problem. This gives a proof of convergence since the objective functions meet the assumptions of the proofs given in Chapter Two and also gives the option for further refinement of the objective function through the addition of penalty terms as needed. Finally, the technique was applied to frames of Hyperspectral Images in order to denoise and identify plumes of interest.

5 ℓ_1-Constrained BSSVD Classifier

In this section we explore the problem of incorporating an ℓ_1-constrained classifier into the algorithm for the BSSVD. The goal is to show that the inclusion of the classifier further aids in the beneficial properties of the BSSVD including denoising, variable selection and robustness to outliers.

5.1 A Sparse ℓ_1-Constrained Classifier

The first requirement is to formulate the optimization problem for the ℓ_1 constrained classifier. In this section we will explore two different classifiers, an elastic net support vector machine with a squared loss function and a naive elastic net classifier [89, 97, 101, 87, 25, 11]. If we let $\hat{y}, \hat{b}, \hat{1}$ be vectors of the training labels, a constant vector of the bias correction and a vector of ones respectively, and we let Y be defined to be a diagonal matrix with $\hat{y}$ on the diagonal, then the elastic net SVM is given by,

$$\underset{w,\hat{b}}{\text{minimize}} \;\; \|w\|_1 + \frac{1}{2}\|w\|_2^2 + \frac{\lambda_1}{2}\left\|Y \cdot (Xw + \hat{b}) - \hat{1}\right\|_2^2 \tag{5.1}$$

where X is a matrix containing the data. For the naive elastic net classifier, the optimization problem is given by,

$$\underset{v}{\text{minimize}} \;\; \|v\|_1 + \frac{1}{2}\|v\|_2^2 + \frac{\lambda_1}{2}\|Av - f\|_2^2 \tag{5.2}$$

In Equation (5.2) A is a data matrix and f is a training vector. When tested on sample data, as will be shown below, these optimization problems have performed well, both in terms of separating the data and enforcing sparsity. The majority of the time to run these algorithms is spent finding a pseudoinverse needed for each of the iterations which is of size $n \times n$ where n is the number of variables. This computation takes $O(n^3)$ operations.

Equation (5.1) is written in the Split Bregman framework as

$$\underset{d,w,\hat{b}}{\text{minimize}} \; \|d\|_1 + \frac{1}{2}\|w\|_2^2 + \frac{\lambda_1}{2}\left\|Y\cdot(Xw+\hat{b})-\hat{1}\right\|_2^2 + \frac{\lambda_2}{2}\|d-w-\beta\|_2^2 \tag{5.3}$$

where the Bregman parameter is written as β instead of the usual b to avoid confusion with the vector of biases $\hat{b}$. To find the iterations needed to solve this optimization problem, we first separate into differentiable and non-differentiable portions. Thus, we first look at differentiating terms involving w. Keeping in mind that Y is symmetric, in this case we have a function f given by,

$$\begin{aligned} f(w) =& \frac{1}{2}w^Tw + \frac{\lambda_1}{2}\left(Y\cdot(Xw+\hat{b})-\hat{1}\right)^T\left(Y\cdot(Xw+\hat{b})-\hat{1}\right) \\ &+ (d-w-\beta)^T(d-w-\beta) \\ =& \frac{1}{2}w^Tw + \frac{\lambda_1}{2}w^TX^TYYXw + \frac{\lambda_1}{2}\cdot 2\cdot w^TX^TYY\hat{b} \\ &- \frac{\lambda_1}{2}\cdot 2\cdot w^TYX^T\hat{1} + \frac{\lambda_1}{2}\hat{b}^TYY\hat{b} - \frac{\lambda_1}{2}\cdot 2\cdot \hat{b}^TY\hat{1} \\ &+ \frac{\lambda_1}{2}\hat{1}^T\hat{1} + \frac{\lambda_2}{2}d^Td + \frac{\lambda_2}{2}w^Tw + \frac{\lambda_2}{2}\beta^T\beta \\ &- \frac{\lambda_2}{2}\cdot 2\cdot d^Tw - \frac{\lambda_2}{2}\cdot 2\cdot d^T\beta + \frac{\lambda_2}{2}\cdot 2\cdot \beta^Tw \end{aligned} \tag{5.4}$$

If we note that the expression YY equals the identity and find $\frac{\partial f}{\partial w}$ and set it equal to zero, we end up with the linear system,

$$\left[(1+\lambda_2)I + \lambda_1X^TX\right]w = \lambda_1\left(X^TY\hat{1} - X^T\hat{b}\right) + \lambda_2(d-\beta).$$

If we define

$$A = (1+\lambda_2)I + \lambda_1X^TX$$

then we have that the iteration for w_{k+1} based on $\hat{b}_k, d_k$ and β_k is given by

$$w_{k+1} = A^\dagger\cdot\left[\lambda_1\left(X^TY\hat{1} - X^T\hat{b}_k\right) + \lambda_2(d_k-\beta_k)\right].$$

where $A^\dagger$ stands for the pseudoinverse of A.

To find the iterations for $\hat{b}$ we first look in the above equations for terms ultimately including $\hat{b}$. To this end we find,

$$\begin{aligned} h(b) &= \frac{\lambda_1}{2} \cdot 2 \cdot w^T X^T \hat{b} + \frac{\lambda_1}{2} \hat{b}^T \hat{b} - \frac{\lambda_1}{2} \cdot 2 \cdot \hat{b}^T Y \hat{1} \\ \frac{\partial h}{\partial \hat{b}} &= \lambda_1 \cdot 2 \cdot Xw + \lambda_1 \hat{b} - \lambda_1 Y \hat{1} = 0 \\ \hat{b} &= Y\hat{1} - Xw \\ b \cdot \hat{1} &= Y\hat{1} - w^T X^T \\ b \cdot \hat{1}^T \hat{1} &= \hat{1}^T Y \hat{1} - \hat{1}^T X w \\ mb &= \hat{1}^T Y \hat{1} - \hat{1}^T X w \\ b &= \frac{\sum_i y_i - X_i w}{m}. \end{aligned} \tag{5.5}$$

where X_i represents the ith row of X. Accordingly, the iteration for b_{k+1} is defined to be

$$b_{k+1} = \frac{\sum_i y_i - X_i w_{k+1}}{m}. \tag{5.6}$$

The variable d is found identically as in the Sparse PCA and BSSVD algorithms, being defined as

$$(d_{k+1})_i = sign((w_{k+1})_i + (\beta_k)_i) * max(|w_{k+1})_i + (\beta_k)_i| - \frac{1}{\lambda_2}, 0). \tag{5.7}$$

The Bregman parameter is updated in the usual manner i.e.,

$$\beta_{k+1} = \beta_k + w_{k+1} - d_{k+1}. \tag{5.8}$$

The final algorithm is as follows.

Bregman Elastic Net SVM

```
1: Select parameters $\lambda_1$, $\lambda_2$
2: Calculate $A^{-1}$.
3: Initialize $w_0 = 0, d_0 = 0, \hat{b}_0 = 0, \beta_0 = 0$ and set $k = 0$
4: while $\|w_k - w_{k-1}\| \geq \delta_w$ do
```

```
 5:      $w_{k+1} = A^{-1} \cdot \left[\lambda_1 \left(X^T Y \hat{1} - X^T \hat{b}_k\right) + \lambda_2 \left(d_k - \beta_k\right)\right]$
 6:      $b_{k+1} = \frac{\sum_i y_i - w_{k+1}^T X_i^T}{m}$
 7:      $d_{k+1} = sign(w_{k+1} + \beta_k) * max(|w_{k+1} + \beta_k| - \frac{1}{\lambda_2}, 0)$
 8:      $\beta_{k+1} = \beta_k + w_{k+1} - d_{k+1}$
 9:      $k = k + 1$
10: end while
```

For the elastic net classifier, the derivation is more straightforward. Recall that the elastic net classifier is defined as

$$\underset{v}{\text{minimize}} \;\; \|v\|_1 + \frac{1}{2}\|v\|_2^2 + \frac{\lambda_1}{2}\|Av - f\|_2^2 \tag{5.9}$$

In the Split Bregman framework this optimization problem becomes

$$\underset{d,v}{\text{minimize}} \;\; \|d\|_1 + \frac{1}{2}\|v\|_2^2 + \frac{\lambda_1}{2}\|Av - f\|_2^2 + \frac{\lambda_2}{2}\|d - v - b\|_2^2 \tag{5.10}$$

To find the iterations for v we again focus on the terms involving v, i.e.

$$\begin{aligned} f(v) =& \frac{1}{2}v^T v + \frac{\lambda_1}{2}(Av - f)^T (Av - f) + \frac{\lambda_2}{2}(d - v - b)^T (d - v - b) \qquad (5.11) \\ v_{terms} =& \frac{1}{2}v^T v + \frac{\lambda_1}{2}\left(v^T A^T A v - 2f^T A v + f^T f\right) + \frac{\lambda_2}{2}\left(d^T d - 2d^T v + v^T v + 2b^T v + b^T b - 2d^T b\right) \\ \text{So } \frac{\partial f}{\partial v} =& v + \lambda_1 A^T A v - \lambda_1 A^T f - \lambda_2 d + \lambda_2 b + \lambda_2 v = 0 \\ & \left[(1 + \lambda_2) I + \lambda_1 A^T A\right] v = \lambda_1 A^T f + \lambda_2 (d - b) \end{aligned}$$

Thus, if B is defined as $(1 + \lambda_2)I + \lambda_1 A^T A$ the algorithm for the elastic net will be as follows.

Bregman Elastic Net

```
1: Select parameters $\lambda_1$, $\lambda_2$
2: Calculate $B^{-1}$.
```

```
3: Initialize $v_0, d_0, b_0$ and set $k = 0$
4: while $\|v_k - v_{k-1}\| \geq \delta_v$ do
5:     $v_{k+1} = B^{-1b} \cdot \lambda_1 A^T f + \lambda_2(d - b)$
6:     $d_{k+1} = sign(v_{k+1} + b_k) * max(|v_{k+1} + b_k| - \frac{1}{\lambda_2}, 0)$
7:     $b_{k+1} = b_k + v_{k+1} - d_{k+1}$
8:     $k = k + 1$
9: end while
```

In order to test the performance of these classifiers, they were first tested on separable data. The test data used was constructed as follows, there were 500 observations for ten experiments, the first ten variables for 250 of these observations was uniformly distributed data on the interval [0,1], the first ten variables for the remaining 250 observations are on the interval [0,-1]. The remaining variables were uniformly distributed on the interval [0,1]. Thus, the first ten variables were the decision variables, while the remaining variables were noise. The ten experiments were separated by the number of variables included, increasing from 100 - 1000 variables in increments of one hundred. Table 5.1 gives the results for both classifiers. Note the constant nature of the elastic net classifier's iterations to convergence. It is also faster in most cases than the SVM, particularly as the number of variables increases. Both classifiers consistently use the ten decision variables for the classification, setting the rest to zero. Also, the times to convergence include the calculation of the needed pseudoinverse which is of size $P \times P$.

Table 5.1: Separable Data Classifier Results

Metrics	Elastic Net SVM	Elastic Net Classifier
	N = 500, P = 100	
Number Incorrectly Classified	0	0
Number of Non-Zero Components	10	10
Iterations to Convergence	30	57
Seconds to Convergence	0.26	0.11
	N = 500, P = 200	
Number Incorrectly Classified	0	0
Number of Non-Zero Components	10	10
Iterations to Convergence	35	54
Seconds to Convergence	0.44	0.16
	N = 500, P = 300	
Number Incorrectly Classified	0	0
Number of Non-Zero Components	10	10
Iterations to Convergence	46	59
Seconds to Convergence	0.68	0.32
	N = 500, P = 400	
Number Incorrectly Classified	0	0
Number of Non-Zero Components	10	10
Iterations to Convergence	52	58
Seconds to Convergence	1.02	0.51
	N = 500, P = 500	
Number Incorrectly Classified	0	0
Number of Non-Zero Components	10	10
Iterations to Convergence	56	56
Seconds to Convergence	1.11	0.63
	N = 500, P = 600	
Number Incorrectly Classified	0	0
Number of Non-Zero Components	10	10
Iterations to Convergence	59	54
Seconds to Convergence	1.99	0.77
	N = 500, P = 700	
Number Incorrectly Classified	0	0
Number of Non-Zero Components	10	10
Iterations to Convergence	72	60
Seconds to Convergence	2.60	1.04
	N = 500, P = 800	
Number Incorrectly Classified	0	0
Number of Non-Zero Components	10	10
Iterations to Convergence	74	58
Seconds to Convergence	2.965	1.37
	N = 500, P = 900	
Number Incorrectly Classified	0	0
Number of Non-Zero Components	10	10
Iterations to Convergence	71	51
Seconds to Convergence	3.35	1.65
	N = 500, P = 1000	
Number Incorrectly Classified	0	0
Number of Non-Zero Components	10	10
Iterations to Convergence	83	55
Seconds to Convergence	4.44	1.90

In the next set of experiments, the sizes were kept the same, and the data remained largely unchanged. The two major differences were that the data were shifted away from the origin, and were made to overlap slightly in order to create a linearly non-separable data set. Pseudo-matlab code to construct the data is found in the Appendix. Results are found in table 5.2. Again, the elastic net classifier is generally faster than the elastic net SVM, however the performance of the two classifiers was very similar with this data. The elastic net SVM held a slight edge in the number of non-zero components.

To run this classifier on data that has been decomposed via the BSSVD, we simply replace X above with the decomposed version of X from the BSSVD, denoted by $\tilde{X}$

$$\underset{w,\hat{b}}{\text{minimize}} \;\; \|w\|_1 + \frac{1}{2}\|w\|_2^2 + \frac{\lambda_1}{2}\left\| diag(\hat{y}) \cdot (\tilde{X}w + \hat{b}) - \hat{1} \right\|_2^2 \tag{5.12}$$

Since both classifiers performed similarly, the next examples will be run using the SVM. The naive elastic net could easily be substituted, however, the purpose of the examples to follow is not to compare classifiers, but rather to illustrate the technique of using the BSSVD to denoise the data prior to and concurrently with the fitting of the classifier.

The data used with Equation (5.12) consists of 100 observations and 10 variables. The first fifty observations have labels of 1, and the next fifty have labels of -1. For the first five variables, the first fifty observations have a value of one, the second fifty have random gaussian values with a mean of zero and a variance of 0.4. For variables six through ten, the first fifty observations have the random values, and the second fifty observations have a value of -1. The data is represented as a mesh plot in Figure 5.1. Directly below that, in Figure 5.2 is the rank two approximation to the data based on the SVD. Finally, the third figure, Figure 5.3 shows the rank two BSSVD approximation to the data. It is clear that the BSSVD data not only has much less noise in the random value areas, but even in the constant value areas of the data as well.

After the BSSVD decomposition, the data was fitted with a classifier using Equation (5.12). Figure 5.4 is a plot of both the original data based decision vector as well as the BSSVD based decision vector. The values for the BSSVD based decision vector for

Table 5.2: Non-Separable Data Classifier Results

Metrics	Elastic Net SVM	Elastic Net Classifier
	N = 500, P = 100	
Number Incorrectly Classified	3	5
Number of Non-Zero Components	74	83
Iterations to Convergence	25	40
Seconds to Convergence	0.18	0.08
	N = 500, P = 200	
Number Incorrectly Classified	2	3
Number of Non-Zero Components	82	89
Iterations to Convergence	42	57
Seconds to Convergence	0.46	0.16
	N = 500, P = 300	
Number Incorrectly Classified	3	5
Number of Non-Zero Components	89	92
Iterations to Convergence	52	64
Seconds to Convergence	0.68	0.32
	N = 500, P = 400	
Number Incorrectly Classified	4	4
Number of Non-Zero Components	98	107
Iterations to Convergence	59	70
Seconds to Convergence	1.11	0.46
	N = 500, P = 500	
Number Incorrectly Classified	12	13
Number of Non-Zero Components	99	108
Iterations to Convergence	61	68
Seconds to Convergence	1.19	0.71
	N = 500, P = 600	
Number Incorrectly Classified	24	22
Number of Non-Zero Components	108	113
Iterations to Convergence	60	59
Seconds to Convergence	1.82	0.88
	N = 500, P = 700	
Number Incorrectly Classified	25	25
Number of Non-Zero Components	133	133
Iterations to Convergence	56	50
Seconds to Convergence	2.04	1.21
	N = 500, P = 800	
Number Incorrectly Classified	23	23
Number of Non-Zero Components	156	156
Iterations to Convergence	63	53
Seconds to Convergence	2.69	1.68
	N = 500, P = 900	
Number Incorrectly Classified	19	17
Number of Non-Zero Components	120	119
Iterations to Convergence	68	56
Seconds to Convergence	3.55	1.53
	N = 500, P = 1000	
Number Incorrectly Classified	8	8
Number of Non-Zero Components	96	102
Iterations to Convergence	80	64
Seconds to Convergence	4.03	2.35

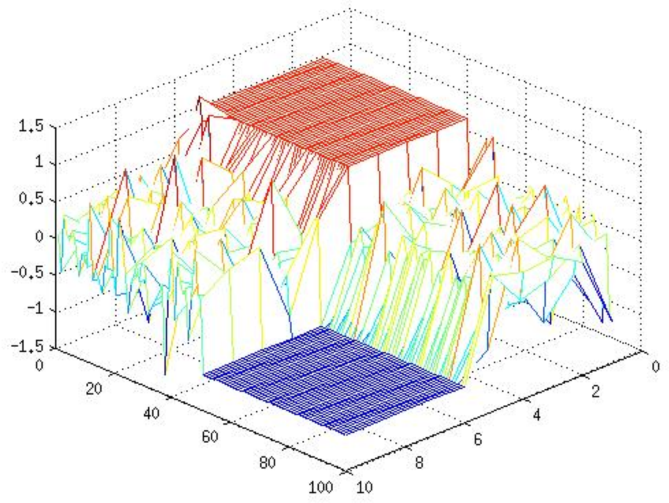

Figure 5.1: Mesh plot of the data used to test Equation (5.12)

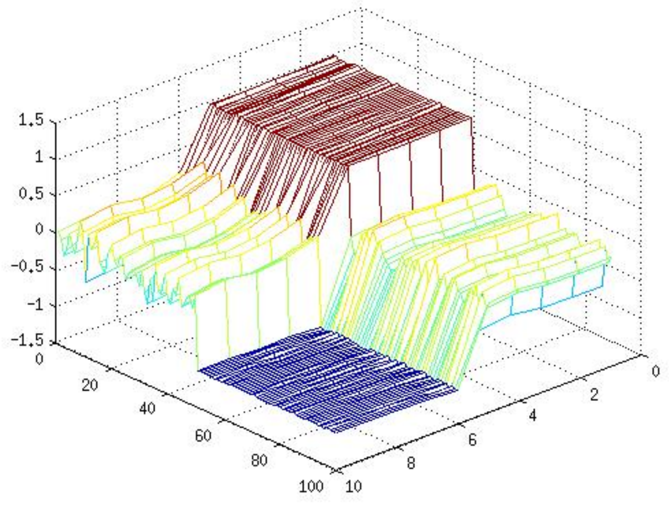

Figure 5.2: Mesh plot of the rank two SVD approximation to the data used to test Equation (5.12)

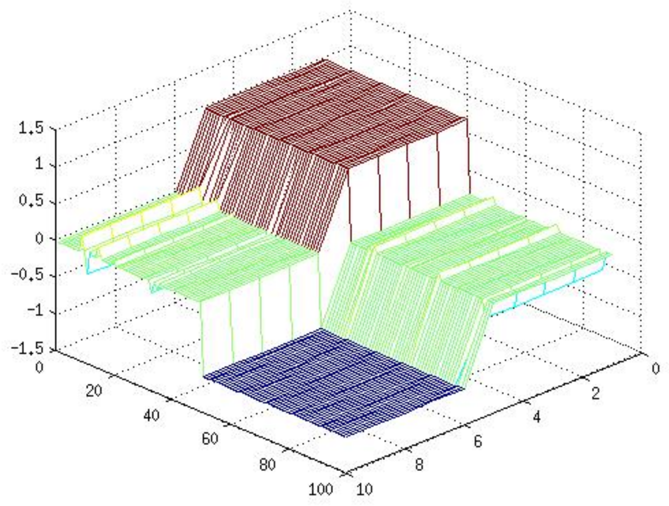

Figure 5.3: Mesh plot of the rank two BSSVD approximation to the data used to test Equation (5.12)

variables one through five are actually a small constant, negative number, while the original data based decision vector has only one nonzero value for the same set of variables. Thus, the original data based decision vector is actually sparser than the BSSVD based vector. However, after applying the decision vector back to the original data, the model trained on the original data makes three errors. In these cases, having only one small non-zero component (in the third component) for the first five variables is not sufficient to overcome applying the large nonzero values found in the eighth and ninth components to the noise found in the original data. This scenario does not occur with every random data set that is created, but is a good example of the benefits of denoising the data prior to building the model.

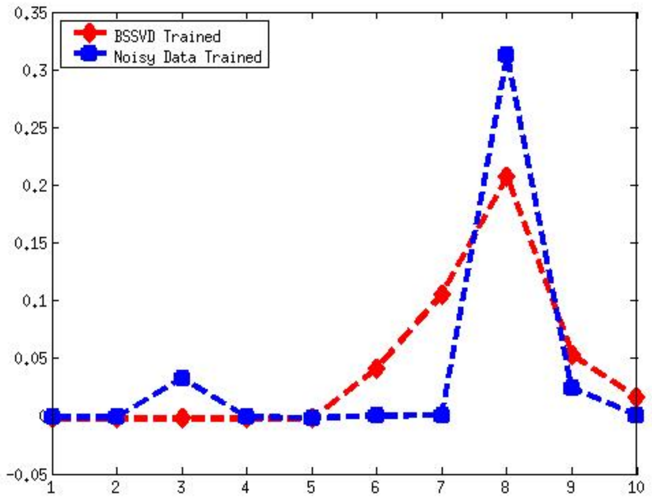

Figure 5.4: Plots of the decision vectors for the model trained on the original, noisy data, and the denoised BSSVD decomposed data.

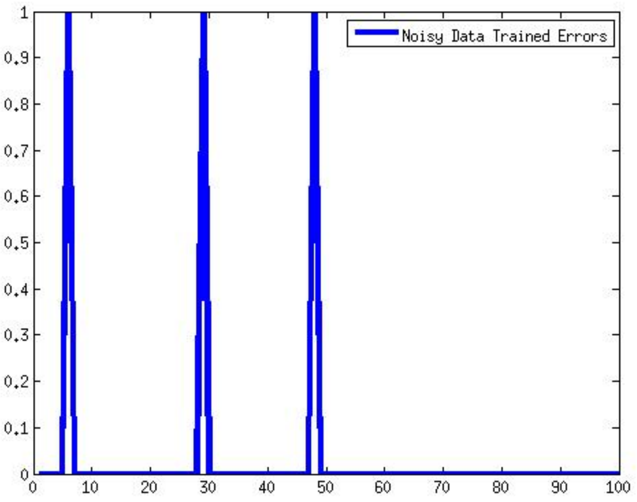

Figure 5.5: Plot of the errors from the model based on the original data. Note that they all occur within the first 50 observations where the small decision weight was unable to overcome larger weights applied to noisy data.

5.2 BSSVD$_{l1}$

For the optimization problem where we are simultaneously running the BSSVD and the ℓ_1-constrained classifier (BSSVD$_{l1}$), the proposed rank one approximation is given by

$$\begin{aligned} \underset{u,v,w}{\text{minimize}} \ & \|u\|_1 + \|v\|_1 + \|w\|_1 + \frac{1}{2}\|u\|_2^2 + \frac{1}{2}\|v\|_2^2 + \frac{1}{2}\|w\|_2^2 \\ & + \frac{\lambda_1}{2}\left\|X - uv^T\right\|_F^2 + \frac{\lambda_2}{2}\left\|Y(uv^Tw + \hat{b}) - \hat{1}\right\|_2^2 \end{aligned} \tag{5.13}$$

In the Split Bregman framework, we solve the following problem.

$$\begin{aligned} \underset{d^u,d^v,d^w,u,v,w}{\text{minimize}} \ & \|d^u\|_1 + \|d^v\|_1 + \|d^w\|_1 + \frac{1}{2}\|u\|_2^2 + \frac{1}{2}\|v\|_2^2 + \frac{1}{2}\|w\|_2^2 \\ & + \frac{\lambda_1}{2}\left\|X - uv^T\right\|_F^2 + \frac{\lambda_2}{2}\left\|Y(uv^Tw + \hat{b}) - \hat{1}\right\|_2^2 \\ & + \frac{\lambda_3}{2}\|d^u - u - b^u\|_2^2 + \frac{\lambda_4}{2}\|d^v - v - b^v\|_2^2 + \frac{\lambda_5}{2}\|d^w - w - b^w\|_2^2 \end{aligned} \tag{5.14}$$

If we focus on the term

$$\left\|Y(uv^Tw + \hat{b}) - \hat{1}\right\|_2^2 \tag{5.15}$$

we see that this expands to

$$\begin{aligned} =& w^Tvu^TYYuv^Tw + 2w^Tvu^TY^TY\hat{b} \\ & - 2w^Tvu^TY^T\hat{1} + \hat{b}^TY^TY\hat{b} - 2\hat{b}^TY + \hat{1}^T\hat{1} \\ =& w^Tvu^Tuv^Tw + 2w^Tvu^T\hat{b} - 2w^Tvu^TY\hat{1} \\ & + \hat{b}^T\hat{b} - 2\hat{b}^TY\hat{1} + \hat{1}^T\hat{1} \end{aligned} \tag{5.16}$$

Note that the simplification is possible due to the facts that YY equals the identity and that it is also symmetric. To find the optimal values for u, v, w and $\hat{b}$ we fix two of the variables and differentiate with respect to the third. First, we focus on terms including u. Then, the optimal value for u_{k+1} based on the values of v_k, w_k and $\hat{b}_k$ is given by minimizing the following:

$$\begin{aligned} u_{k+1} =& \arg\min_u \frac{1}{2}\|u\|_2^2 + \frac{\lambda_1}{2}\left\|X - uv_k^T\right\|_F^2 \\ & + \frac{\lambda_2}{2}\left\|Y(uv_k^Tw_k + \hat{b}_k) - \hat{1}\right\|_2^2 + \frac{\lambda_3}{2}\|d_k^u - u - b_k^u\|_2^2. \end{aligned} \tag{5.17}$$

Differentiating with respect to u gives,

$$\begin{aligned} \frac{\partial u_{k+1}}{\partial u} &= u + \lambda_1 \left(v_k^T v_k u - X v_k\right) \\ &+ \lambda_2 \left(w_k^T v_k v_k^T w_k u + \hat{b}_k v_k^T w_k - w_k^T v_k diag(\hat{y})\hat{1}\right) \\ &+ \lambda_3 \left(u - d_k^u + b_k^u\right). \end{aligned} \tag{5.18}$$

Taking $\frac{\partial u_{k+1}}{\partial u} = 0$ we obtain

$$\begin{aligned} \alpha &= 1 + \lambda_3 + \lambda_1 v_k^T v_k + \lambda_2 w_k^T v_k v_k v_k^T w_k \\ \alpha u &= \lambda_1 X v_k + \lambda_2 (w_k^T v_k \hat{1}^T Y - \hat{b}_k v_k^T w_k) + \lambda_3 (d_k^u - b_k^u) \end{aligned} \tag{5.19}$$

Note that the left side results in a scalar multiple of u, which implies that

$$u_{k+1} = \frac{1}{\alpha}\left(\lambda_1 X v_k + \lambda_2 (w_k^T v_k \hat{1}^T diag(\hat{y}) - \hat{b}_k v_k^T w_k) + \lambda_3 (d_k^u - b_k^u)\right). \tag{5.20}$$

The optimal value for v_{k+1} based on the values of u_{k+1}, w_k and $\hat{b}_k$ is given by minimizing the following.

$$\begin{aligned} v_{k+1} =& \arg\min_v \frac{1}{2}\|v\|_2^2 + \frac{\lambda_1}{2}\left\|X - u_{K+1} v^T\right\|_F^2 \\ &+ \frac{\lambda_2}{2}\left\|Y(u_{k+1} v^T w_k + \hat{b}_k) - \hat{1}\right\|_2^2 + \frac{\lambda_4}{2}\|d_k^v - v - b_k^v\|_2^2 \end{aligned} \tag{5.21}$$

Differentiating with respect to v gives,

$$\begin{aligned} \frac{\partial v_{k+1}}{\partial v} &= v + \lambda_1 \left(u_{k+1}^T u_{k+1} v - X^T u_{k+1}\right) \\ &+ \lambda_2 \left(w_k^T u_{k+1}^T u_{k+1} w_k^T v + w_k u_{k+1}^T \hat{b}_k - w_k u_{k+1}^T diag(\hat{y})\hat{1}\right) \\ &+ \lambda_4 \left(v - d_k^v + b_k^v\right) \end{aligned} \tag{5.22}$$

which, when set equal to zero results in the equation below,

$$\begin{aligned} A &= (1 + \lambda_4 + \lambda_1 u_{k+1}^T u_{k+1}) I + \lambda_2 w_k u_{k+1}^T u_{k+1} w_k^T \\ A v &= \lambda_1 X^T u_{k+1} + \lambda_2 (w_k u_{k+1}^T diag(\hat{y})\hat{1} - w_k u_{k+1}^T b_k) + \lambda_4 (d_k^v - b_k^v). \end{aligned} \tag{5.23}$$

Note that the left side this time results in a matrix multiple of v ,which implies that

$$v_{k+1} = A^\dagger \left(\lambda_1 X^T u_{k+1} + \lambda_2 (w_k u_{k+1}^T diag(\hat{y})\hat{1} - w_k u_{k+1}^T b_k) + \lambda_4 (d_k^v - b_k^v)\right) \tag{5.24}$$

where $A^\dagger$ again stands for the pseudoinverse of A. Solving this linear system can be done using many different solvers, however, for a small problem such as the example that is displayed at the end of this section, the pseudoinverse works well.

The optimal value for w_{k+1} based on the values of u_{k+1}, v_{k+1} and $\hat{b}_k$ is given by minimizing the following.

$$w_{k+1} = \arg\min_{w} \frac{1}{2}\|w\|_2^2 + \frac{\lambda_2}{2}\left\|Y(u_{k+1}v_{k+1}^T w + \hat{b}_k) - \hat{1}\right\|_2^2 + \frac{\lambda_5}{2}\|d_k^w - w - b_k^w\|_2^2 \tag{5.25}$$

Differentiating with respect to w gives,

$$\begin{aligned}\frac{\partial w_{k+1}}{\partial w} = & \\ & w + \lambda_2\left(v_{k+1}u_{k+1}^T u_{k+1}v_{k+1}^T w + v_{k+1}u_{k+1}^T\hat{b}_k - v_{k+1}u_{k+1}^T Y\hat{1}\right) \\ & + \lambda_5\left(w - d_k^w + b_k^w\right)\end{aligned} \tag{5.26}$$

Which, when set equal to zero results in the equation:

$$\begin{aligned} B &= (1+\lambda_5)I + \lambda_2 v_{k+1}u_{k+1}^T u_{k+1}v_{k+1}^T \\ Bw &= \lambda_2(v_{k+1}u_{k+1}^T Y\hat{1} - v_{k+1}u_{k+1}^T\hat{b}_k) + \lambda_5(d_k^w - b_k^w).\end{aligned} \tag{5.27}$$

Note that the left side again results in a matrix multiple of w, which implies that

$$w_{k+1} = B^\dagger\left(\lambda_2(v_{k+1}u_{k+1}^T Y\hat{1} - v_{k+1}u_{k+1}^T\hat{b}_k) + \lambda_5(d_k^w - b_k^w)\right) \tag{5.28}$$

Finally, for $\hat{b}_{k+1}$ we have that

$$\hat{b}_{k+1} = \frac{\sum_i y_i - (u_{k+1}v_{k+1}^T)_i w_{k+1}}{m}\cdot\hat{1} \tag{5.29}$$

where $(u_{k+1}v_{k+1}^T)_i$ represents the *ith* row of $u_{k+1}v_{k+1}^T$.

For the rank one ℓ_1-BSSVD classifier, we implement the following algorithm. For the parameter selection of variables $\lambda_1 - \lambda_5$, the decision is based on different criteria. To choose λ_2, the left vector of the BSSVD$_{l1}$ is visually inspected to find a split of the data which may result in a reasonable number of observations being labeled as ambiguous. From there, λ_2 is held fixed. The three parameters $\lambda_3 - \lambda_5$ are increased until convergence and sparsity reach reasonable levels. Finally, λ_1 is varied to balance variance explained versus sparsity in the three variables u, v, w.

ℓ_1-BSSVD Classifier

```
1: Select parameters $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$
2: Calculate the SVD of $X = \hat{U}\hat{S}\hat{V}^T$
3: Calculate $A^\dagger, B^\dagger$.
4: Initialize $u_0, v_0, w_0, \hat{b}_0, d_0^u, d_0^v, d_0^w$ and set $k = 0$
5: while $\left[d(u_k, u_{k-1}), d(v_k, v_{k-1}), d(w_k, w_{k-1}), d(\hat{b}_k, \hat{b}_{k-1})\right] \geq \epsilon_{u,v,w,\hat{b}}$ do
6:     $u_{k+1} = \frac{1}{\alpha}\left(\lambda_1 X v_k + \lambda_2(w_k^T v_k \hat{1}^T Y - \hat{b}_k v_k^T w_k) + \lambda_3(d_k^u - b_k^u)\right)$
7:     $v_{k+1} = A^{-1}\left(\lambda_1 X^T u_{k+1} + \lambda_2(w_k u_{k+1}^T Y\hat{1} - w_k u_{k+1}^T \hat{b}_k) + \lambda_4(d_k^v - b_k^v)\right)$
8:     $w_{k+1} = B^{-1}\left(\lambda_2(v_{k+1} u_{k+1}^T Y\hat{1} - v_{k+1} u_{k+1}^T \hat{b}_k) + \lambda_5(d_k^w - b_k^w)\right)$
9:     $\hat{b}_{k+1} = \frac{\sum_i y_i - (u_{k+1} v_{k+1}^T)_i w_{k+1}}{m} \cdot \hat{1}$
10:    $d_{k+1}^u = sign(u_{k+1} + b_k^u) * max(|u_{k+1} + b_k^u| - \frac{1}{\lambda_3}, 0)$
11:    $d_{k+1}^v = sign(v_{k+1} + b_k^v) * max(|v_{k+1} + b_k^v| - \frac{1}{\lambda_4}, 0)$
12:    $d_{k+1}^w = sign(w_{k+1} + b_k^w) * max(|w_{k+1} + b_k^w| - \frac{1}{\lambda_5}, 0)$
13:    $b_{k+1}^u = b_k^u + u_{k+1} - d_{k+1}^u$
14:    $b_{k+1}^v = b_k^v + v_{k+1} - d_{k+1}^v$
15:    $b_{k+1}^w = b_k^w + w_{k+1} - d_{k+1}^w$
16:    $k = k + 1$
17: end while
```

To test the algorithm, we use the same data construct as was used to test the classifiers on data that had been denoised via the BSSVD. The data consists of 100 observations and 10 variables and is represented as a mesh plot in Figure 5.6. The second figure, Figure 5.7 shows the rank one SVD approximation to the data. Directly below that, in Figure 5.8 is the rank one approximation to the data based on the BSSVD$_{l1}$. Note that it has decomposed the data into largely one variable, which describes the split in the data. The decomposition is less focused on capturing the variance and more focused to capture the split in the classification. This can of course be tuned using the parameters so that more emphasis is put on capturing variance. However, for demonstration, the rank one approximation was tuned to so as to split the training data without error.

Of the most interest is the plot of the left singular vectors. As seen in Figure 5.9 the BSSVD$_{l1}$ left vector is a clean classification of the data into categories, while the SVD based left singular vector used in the same manner would provide poor results. The structure of the BSSVD$_{l1}$ vector is a factor of having the classifier built into the optimization problem. With the classifier included, the observations are automatically used in an optimal sense to represent the data for classifying the data. Figure 5.10 shows the BSSVD$_{l1}$ right singular vector and the SVD based right singular vector. The thing to note in this figure is the emphasis placed by the BSSVD right vector on the variable used for classification. Again, this is an artifact of including the classifier into the optimization problem. This is made even more clear by Figure 5.11, which shows the BSSVD right vector and the very similar ℓ_1 classifier decision vector. Note how the BSSVD right vector mimics the classifier, identifying the variable created for classification.

This example shows is that the utility of including the classifier in the optimization problem is not solely its effect on the classifier, but also the effects it has on the decomposition of the data.

However,the results involving the classifier constrained left vectors indicate that the vector just trains itself to follow the labels presented to the algorithm. To verify this, a test data set was constructed which would illustrate whether or not this phenomenon was occurring. The data consists of 10 variables, all of which were random Gaussian with zero mean and variance equal to one. There were 100 observations with the first fifty being assigned a label of one and the next fifty assigned a label of negative one. Thus, the variables and the labels themselves are constructed to be entirely unrelated, and the influence of the classifier on the total optimization problem and the left vectors in particular

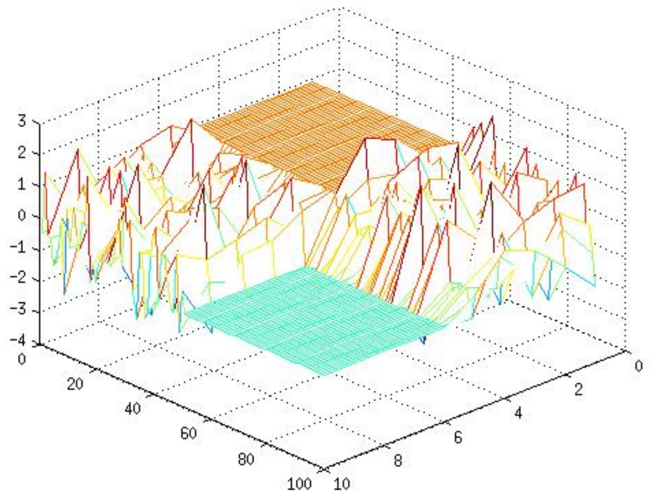

Figure 5.6: Mesh plot of the data used to test the BSSVD$_{l1}$ algorithm

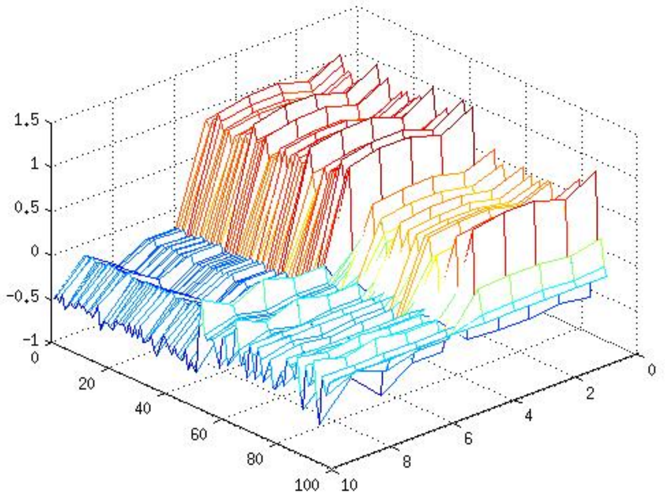

Figure 5.7: Mesh plot of the rank one SVD approximation to the data used to test the BSSVD$_{l1}$ algorithm

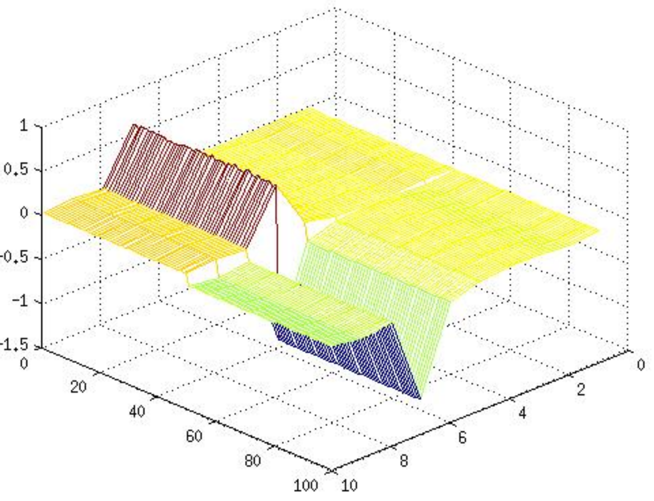

Figure 5.8: Mesh plot of the rank one BSSVD$_{l1}$ approximation to the data used to test the BSSVD$_{l1}$ algorithm

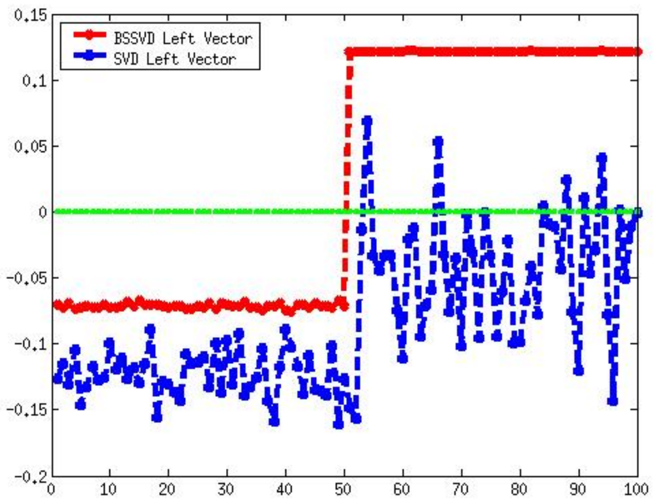

Figure 5.9: Plot of the BSSVD$_{l1}$ left singular vector, the SVD left singular vector and a green line at zero. Note how the BSSVD$_{l1}$ left singular vector provides a classification of the data.

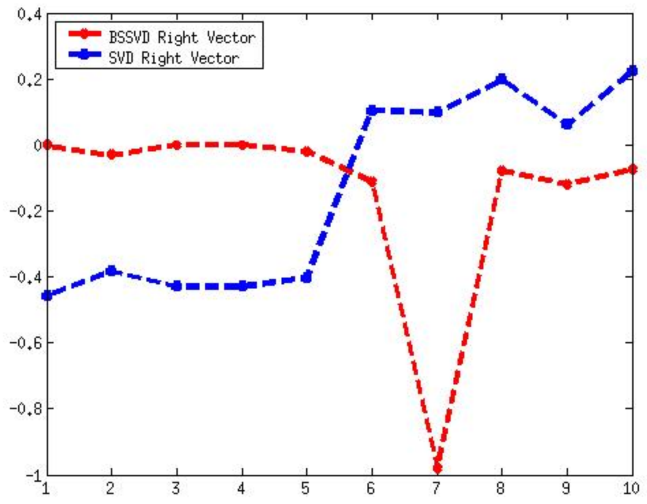

Figure 5.10: Plot of the BSSVD_{l1} right singular vector and the SVD right singular vector.

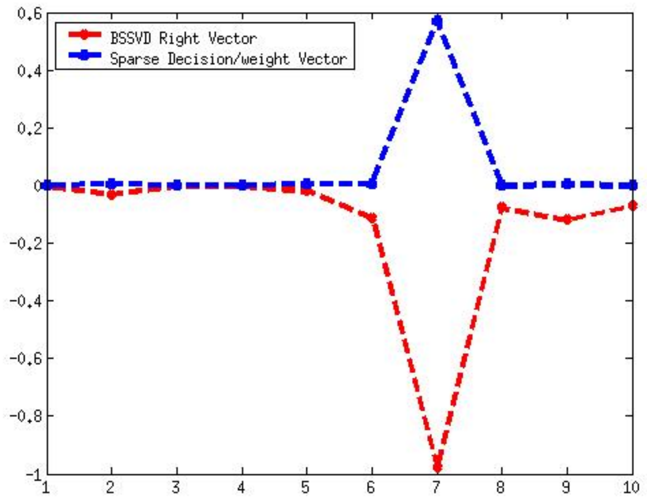

Figure 5.11: Plot of the BSSVD_{l1} right singular vector and the ℓ_1 weighting/decision vector

can be made clear. Figure 5.12 shows the classifier constrained left vector as the penalty parameter associated with the classifier increases. It is clear that the vector can be made to exactly mirror the labels associated with the observations given a parameter that is sufficiently large. It is also clear that it can be made to mirror the left singular vector found from the unconstrained SVD of the data if the parameter is sufficiently small.

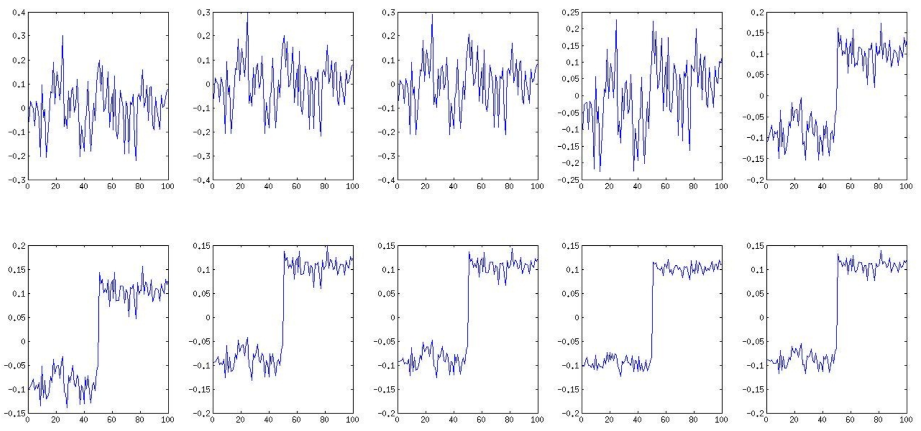

Figure 5.12: Plot of the classifier constrained left vector for the test data. The penalty parameter associated with the classifier increases from left to right and top to bottom.

However, the most interesting results are found in between these two extremes. In this case, the left vector is optimizing the balance between variance explained in the data set for a rank one decomposition, and mimicking the labels supplied. To show this, a simple experiment was constructed. The data was comprised of 50 observations with gaussian random noise with mean 2 and unit variance, N (2,1), in the first five variables. The next five variables were N (0,.5). The next 50 observations were N (-2,1) for the first five variables and N (0,.5) for the second five. To introduce ambiguous records, three observations were altered in the first fifty observations and in the second 50 observations. Specifically, for the altered records in the first fifty observations, the values of the first five variables were set to -2. For the altered records in the second fifty observations, the values of the first five variables were set to 2. The BSSVD$_{l1}$ algorithm was then run with increasing penalty parameters attached to the classifier term.

In Figures 5.13 and 5.14, we see the difference between what the classical SVD will calculate as its first left singular vector and what the BSSVD$_{l1}$ will calculate as its first left singular vector. In particular, we see that as the penalty parameter for the classifier is increased, the BSSVD$_{l1}$ left vector goes from approximating the SVD based left vector, to producing a nice split of the data with visuals of the anomalistic records in the top right and bottom left records, to the split based solely on label as the parameter continues to increase. The records of interest are those in the middle that provide evidence and a means of removing the ambiguous records.

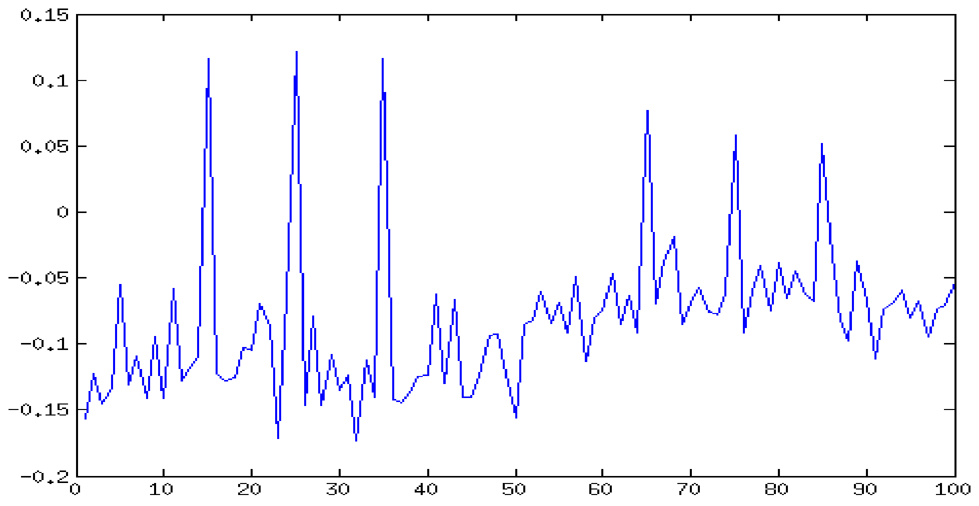

Figure 5.13: The first left singular vector of the example data based on the classic SVD

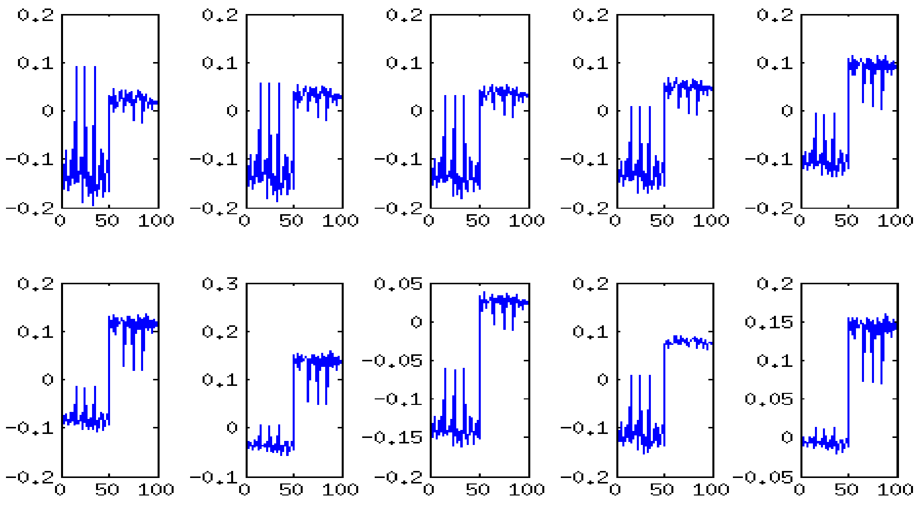
Figure 5.14: The first left singular vector of the example data based on the BSSVD$_{l1}$ algorithm. The penalty parameter increases left to right and top to bottom.

5.3 Arrhythmia Data

According to the National Heart, Lung, and Blood Institute and the National Institution of Health, Arrhythmia is a condition where there is a problem with either the rate or the rhythm of the heartbeat. This means that it can be too slow, fast or just beating irregularly [63]. Arrhythmia can be life threatening at times and can be quite costly to treat [71, 49]

The data used for this section contains information on 452 patients, of which a portion have been diagnosed with an Arrhythmia [1]. There are 279 variables included with the data that are used to construct the classifier. In order to test the BSSVD with ℓ_1-constrained classifier, in particular, Equation (5.14) we will use this data and compare results to the stand-alone elastic net SVM classifier and several other classification methods [24, 66, 39].

As noted in the BSSVD$_{l1}$ section, the application of this algorithm to this data will illustrate the use of the algorithm to remove observations or rows of the data from the data prior to building the classifier. We will analyze the effects of removing observations from both a geometric perspective and also see how it impacts a classifier constructed using the restricted data.

Geometric Analysis of Removed Observations

As shown in chapter five, the BSSVD with ℓ_1 Classifier algorithm (BSSVD$_{l1}$), produced a left vector that was shown to model, based on parameter selection, the labels of the data provided. However, this was only the case when the classifier penalty was set high enough. When the classifier penalty was set at a level that was balanced between a very small value, resulting in the BSSVD left vector, and a very high value, which resulted in mirroring the provided labels, it appeared that it may be removing observations where the label and the optimal variance level value were at odds. It was hypothesized that the algorithm may be removing observations which were ambiguous in their labeling, meaning, it was removing observations where the variables provided were not consistent with the provided label. This was shown using a simple example with two different Gaussian distributions and observations that had been manually altered to be anomalies when compared to other records with the same label. We will now apply this same technique to real world data, namely the Arrhythmia data mentioned at the beginning of this chapter.

The data starts with 16 different classes into which the data is classified. There is one normal heart rate class, 14 different Arrhythmia classes, and one undetermined class. For the following experiments the data corresponding to the undetermined class is dropped, which leaves 430 observations. The response variable is formed by setting it equal to one for the normal heart rate class and to negative one for the remaining classes. In this data set 0.33% of the data is missing [39], to handle this, the missing data is set to zero, then the mean for that variable is used to fill in the missing values.

Again, the data is put through the BSSVD$_{l1}$ algorithm with an increasing penalty parameter on the classifier term. This is done in order to gain a visual look at what levels of the parameter will provide optimal splitting of the data. Figure 5.15 shows the possible splits for 10 different values of the penalty parameter. Again, towards the top right and bottom left of the figure there seem to be the best balance of data removal and retention.

However, since the purpose of this section is to explore any geometric insights gained from removing data points, we will focus on penalties that will in general remove a larger number of points.

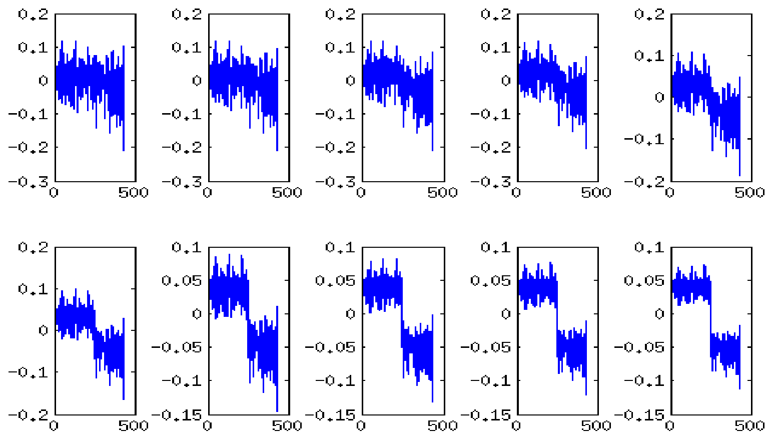

Figure 5.15: The first left singular vectors as determined by the BSSVD$_{l1}$ algorithm for increasing classifier penalties.

Data points are removed from the data if the label provided does not match the sign of the corresponding entry of the first left vector from the BSSVD$_{l1}$ algorithm. In order to visualize the effects of this removal, we put the data through the Laplacian Eigenmaps algorithm to produce a low-dimensional embedding [8]. The first figure below, Figure 5.16 shows the 3D embedding prior to any rotation, with data points in blue corresponding to observations with a class of "normal", and the data points in green corresponding to the "Arrhythmia" data points. Figure 5.17 shows the initial embedding rotated and while much of the data is clumped on the left side of the image, there is a concentration or distribution of green data points on the right. Note in particular the set of points toward the top, with little to no blue points present. Figure 5.18 shows the points in red that represent the "Normal" data points that would be removed based on the BSSVD$_{l1}$ algorithm. See how the removed points are concentrated within the right side mainly in the distribution of green points. Figure 5.19 show in black, the data points that have a class of "Arrhythmia", but would be removed based on the BSSVD$_{l1}$ algorithm. This figure has been rotated in the opposite direction 180 degrees as the black points were generally found below the data from the vantage point of the first two figures. Again, note that these points are heavily concentrated in the ”Normal”, blue points. Finally, Figure 5.20 shows what the original, rotated data would look like if the points that were to be removed, switched their class. The split between the two sets of data becomes much more evident.

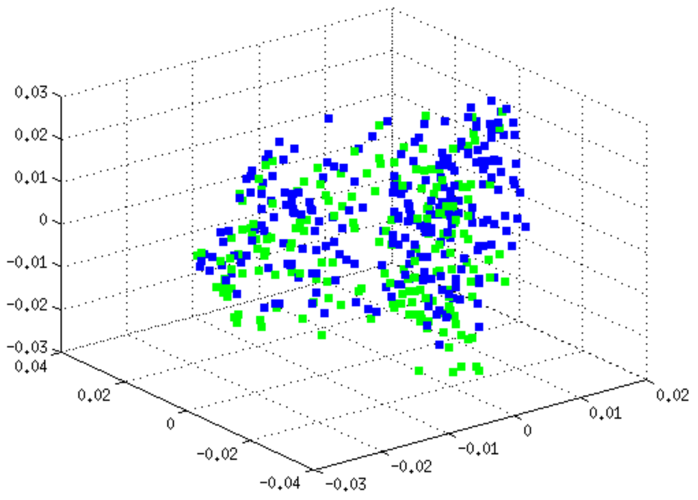

Figure 5.16: The 3D Laplacian Eigenmaps embedding. Normal data points in blue, Arrhythmia data points in green.

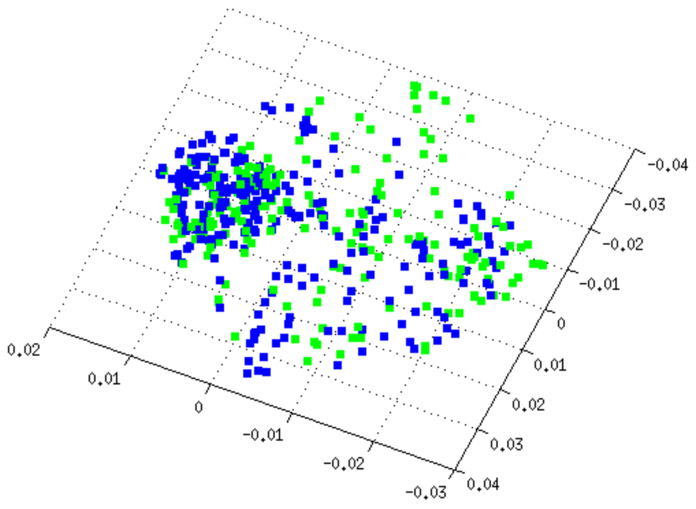

Figure 5.17: The 3D Laplacian Eigenmaps embedding after rotation. Note the concentration/distribution of green points to the right.

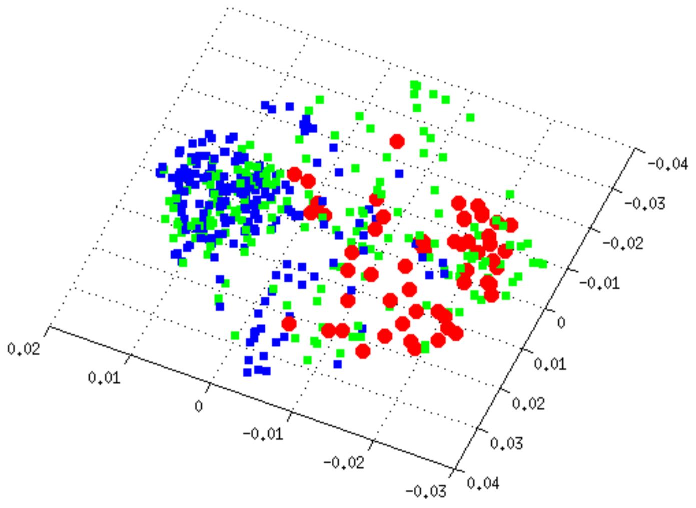

Figure 5.18: In red are the Normal data points that are indicated for removal. Note the location of the points within the distribution of the green Arrhythmia data points.

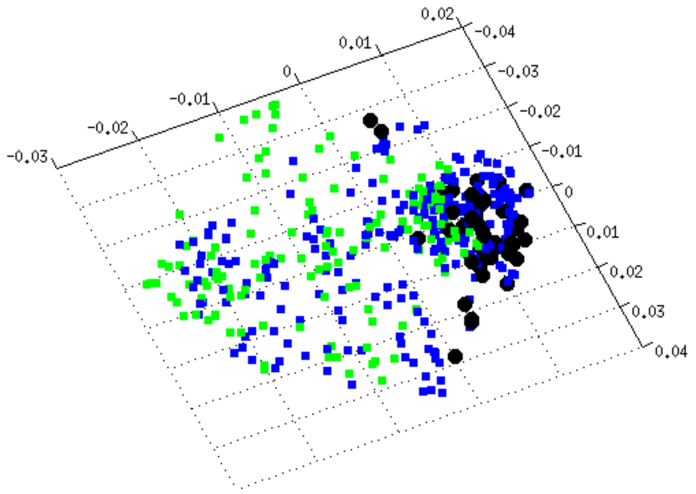

Figure 5.19: In black are the Arrhythmia data points that are indicated for removal. These are generally found in the concentration of blue Normal points.

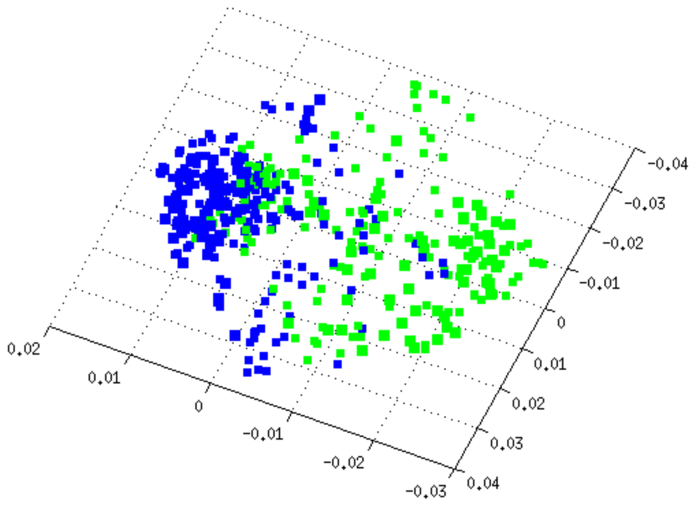

Figure 5.20: The original, rotated data after the points indicated for removal have had their classes switched, revealing a better split between the data.

Next was an experiment to test whether or not removing these observations would improve the accuracy of a classifier. For choosing the parameter that would indicate the points to be removed, the plots in Figure 5.15 were used, meaning that the parameter chosen was based on a visual inspection. Two classifiers were constructed, one that was based on restricted training data, and a normal elastic net SVM. The algorithm used to test and compare the two classifiers was K-fold cross validation with K = 10. The algorithm details are as follows.

1. Separate the data into 10 mutually exclusive, identically sized data sets.
2. Remove one of the 10 data sets and set aside for testing.
3. For the BSSVD_{l1} classifier, run the algorithm on the remaining 9 data sets (training data), using the visually chosen parameter to determine which points to remove from the training data.
4. Using the remaining 9 data sets train and optimize both classifiers.
5. Test the classifiers on the tenth data set, which had been set aside
6. Repeat until all 10 data sets have been used for testing.

By taking the means of the results of all ten tests, the results of the K-fold cross validation show that both classifiers have overall accuracy rates that are identical since the accuracies on the diagonal sum to the same number. See the confusion matrices in Table 5.3.

However, the BSSVD_{l1} classifier identifies a higher number of Arrhythmia cases, and a higher number of false positives. Meaning, the BSSVD_{l1} classifier is predicting more people to have Arrhythmia than are labeled that way. It is important to keep in mind that the labels for this data were created in 1998 by a medical doctor using their expert opinion based on the variables present. This means that there is some ambiguity as to the validity of some of the labels. In either regard, while the accuracy was identical overall,

Table 5.3: BSSVD$_{l1}$ Confusion matrix

BSSVD$_{l1}$	Predicted Arrhythmia	Predicted Normal
Arrhythmia	$\frac{11}{43}$	$\frac{7.5}{43}$
Normal	$\frac{5.7}{43}$	$\frac{18.8}{43}$

Normal SVM	Predicted Arrhythmia	Predicted Normal
Arrhythmia	$\frac{10.5}{43}$	$\frac{8}{43}$
Normal	$\frac{5.2}{43}$	$\frac{19.3}{43}$

the BSSVD$_{l1}$ classifier did show promise in identifying more Arrhythmia cases, and in the medical setting, false positives can at times be more favorable than false negatives.

In terms of accuracy, the both classifiers predicted on average 69.3% of the cases correctly. This is higher than the original accuracy of 63% found in the original paper [39], below what is found in [66], and below the 84% accuracy found in [24].

The accuracy results show that there is still quite a bit of work to do for this algorithm, however, the geometric and confusion matrix results point to promising results. Therefore, future work will be focused on optimal, automated parameter selection and additional penalty terms that may improve the overall accuracy of the classifier. Applications of the geometric interpretations presented in this chapter also deserve more in-depth analysis as they could point to manners in which to identify anomalies or incorrectly labeled data.

5.4 Summary

This section covered the fitting of two ℓ_1-constrained classifiers, namely an Elastic Net SVM and a Least Squares SVM to data using the Split Bregman algorithm. Next, the Elastic Net SVM classifier was fit to data that had been decomposed using the BSSVD and it was shown that in some cases the lack of denoising seen using the SVD rather than the BSSVD led to errors. The Elastic Net SVD and BSSVD were then combined into one optimization problem and it was shown that the BSSVD left vectors could be used to identify ambiguous records within the data being classified. Carrying this notion to real-world data, namely Arrhythmia data, the geometry of the removed observations was analyzed by visualizing the data in a low dimensional space. Finally, the overall affect of removing the ambiguous observations on a classifier were explored.

Novel contributions in this section include fitting these classifiers to the Split Bregman framework, and the inclusion of the classifier into the BSSVD algorithm. We explored the geometric meaning of the classifier constrained decomposition, specifically, we tied identification of ambiguous results to the left vector of the BSSVD decomposition when the classifier was included in the objective function. This algorithm (BSSVD$_{l1}$) was applied to Arrhythmia data and it was shown, using low-dimensional embeddings that the observations being removed could be interpreted as overlapping, and hence confusing to a classifier being applied to the data. We also tested a classifier on the data with these observations being removed and compared the results to a non-restricted classifier, and existing results from the literature.

6 Modularity Maximization in Networks and Anomaly Detection

A network, or graph of data is a group of nodes (points) and edges (lines) connecting them. The edges can be directed or undirected, weighted or unweighted, and not all nodes need be connected to another. A network where each node is connected to at least one other node is referred to as complete.

Networks are used to model many real world applications including social networks [84], disease networks [79], citation networks [75], brain function [72], climate networks [85], and cellular tower data analysis [31] among many others.

Of particular interest is finding subgraphs or communities within a larger network [30, 61, 60, 59, 62, 91, 68, 23, 9, 67]. Many of the techniques to find smaller communities within a larger network are based on Newman's Modularity metric [61, 60, 59, 62]. The modularity metric aims to maximize the number of edges that are found within communities versus the number of edges connecting the communities. In other words it attempts to maximize internal connectedness versus external connectedness.

Formally, let G be a graph with a set of vertices V and edges E. An adjacency matrix A is formed from G, V, E as follows: if two vertices in V are connected by an edge from E, then $A_{ij} = 1$, else $A_{ij} = 0$. Let k be a vector containing the total number of connections (degree) of each node, i.e. if node 1 is connected to 7 other nodes, then the first entry in k would be 7. Then, if we let g_i be the community to which node i belongs, and define $\delta(g_i g_j)$ to be 1 when $i = j$ and 0 otherwise, then the modularity is given as

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(g_i g_j) \tag{6.1}$$

where m is the number of edges in the network. The modularity matrix is defined to be

$$B = A - \frac{kk^T}{2m} \tag{6.2}$$

The modularity matrix gives a measure at each node of the actual connections A minus the expected number of connections $\frac{kk^T}{2m}$.

Much as with the Laplacian within spectral clustering, an exhaustive search to optimize the modularity is intractable for all but the smallest networks. Again, as with the Laplacian and spectral clustering, it has been shown in [62] that the eigenvectors of the modularity matrix are a good approximation to the optimal split via a relaxation of the dichotomous set of values that indicate community membership. Namely, the sign of the entries of the eigenvectors can be used to split the network into communities based on optimizing the modularity metric.

In Figure 6.1 a sample social network is displayed. The clusters in this group are formed using the modularity metric. In this instance, with a small number of nodes and high levels of community structure, large and small subgraphs/communities are apparent. However, in many networks, the community structure is not nearly as strong and anomalistic subgraphs can remain hidden inside the network. For example, in Figure 6.2 a 1024 node graph created by the RMAT algorithm is shown [21]. The graph is again clustered based on modularity, but with much less success. Also, there is an eight node subgraph embedded in this graph which is not visible to the naked eye. More sophisticated techniques are needed to tease out the nodes of this subgraph.

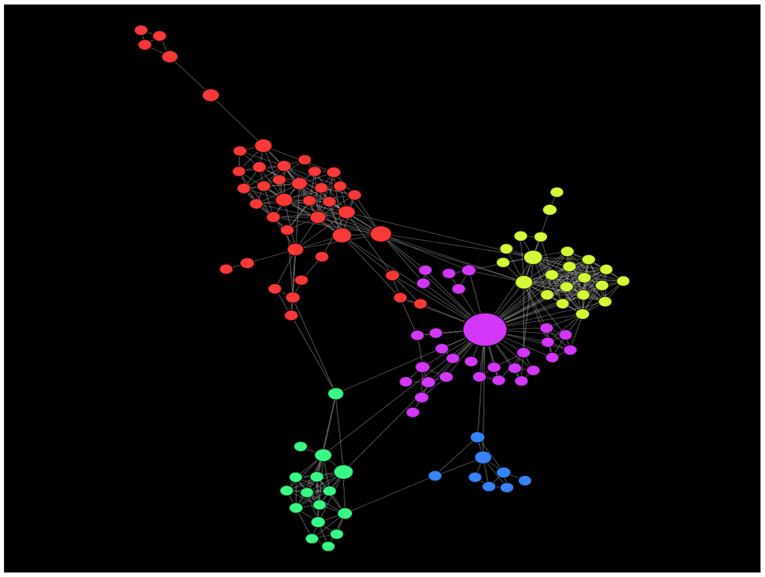

Figure 6.1: A sample social network clustered using modularity. Clusters correspond to different groupings of family and friends based on chronological and other factors.

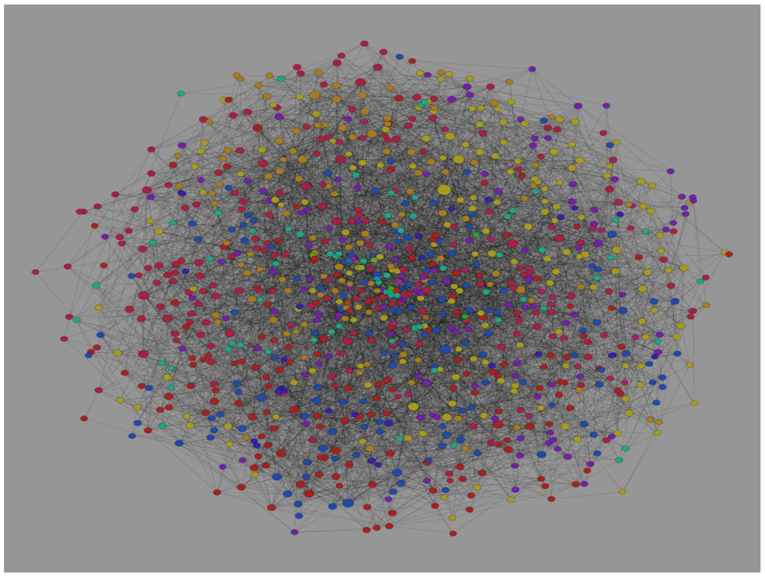

Figure 6.2: A sample RMAT network clustered using modularity, with a hidden eight node subgraph embedded within the graph.

Recently, Miller et al. [57] found that the ℓ_1-norm of the eigenvectors of the modularity matrix could indicate the presence of an anomalistic subgraph. They found this by creating graphs using the RMAT algorithm [21] and analyzing the curve of one norms of the eigenvectors. In the presence of a small, highly interconnected subgraph, one of the eigenvectors tended to have a statistically significantly lower than expected one-norm. In practice, the eigenvectors that corresponded to the subgraphs had large weights for the nodes of the subgraph and smaller weights for the rest of the graph. This strongly suggests that a sparse approximation to this vector could be used to identify unambiguously the nodes of the subgraph.

Given the fact that the modularity matrix is symmetric, the task of finding eigenvectors is equivalent to a PCA or SVD decomposition of this matrix. Singh, Miller et al. [77] applied a semidefinite programming relaxation [26] to calculate a Sparse PCA of an RMAT generated modularity matrix with good results. They were able to detect these small, anomalistic subgraphs using this technique. In this paper they do not calculate more than one sparse eigenvector and the task of calculating subsequent eigenvectors is mentioned in their conclusions as a future problem to be analyzed. Also, it is unclear if the algorithm they have chosen can accept additional penalty terms to adapt to different types of network data. Both of these issues can be addressed using the Split Bregman Sparse PCA algorithm.

6.1 Detecting Graph Anomalies using Bregman Iterations

Firstly, the Split Bregman Sparse PCA algorithm was tested to see if it could identify the eight nodes in the embedded subgraph from Figure 6.2. Figure 6.3 shows the first loading vector when the algorithm is applied to the modularity matrix. The eight nodes are clearly identifiable from the vector.

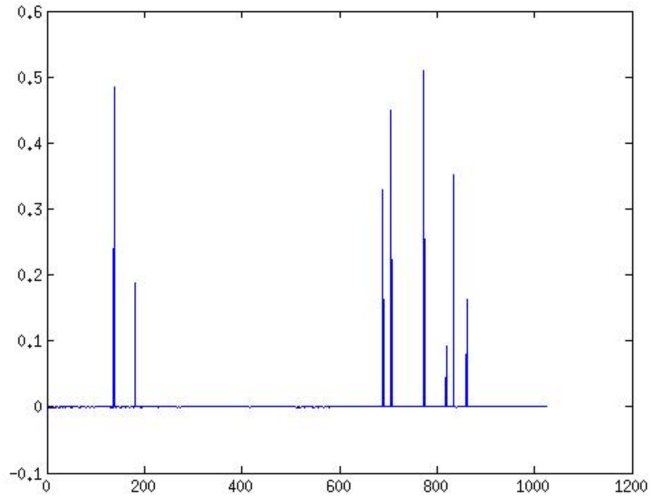

Figure 6.3: The first loading vector from the sparse pca of the modularity matrix associated with Figure 6.2

Next, a second seven node subgraph was embedded using different nodes than in the first. A rank two sparse PCA was performed on the modularity matrix, and as expected, the second sparse loading vector distinctly picks out the seven nodes comprising the subgraph, see Figure 6.4.

Finally, to illustrate the benefit of the sparsity of the solutions found using the sparse PCA, we embed the nodes into $\mathbb{R}^2$ using the coordinates as determined by the rows of the matrix formed by using the sparse loading vectors as columns, see Figures 6.5 and 6.6.. This is exactly how the eigenvectors are used in many spectral clustering algorithms, including

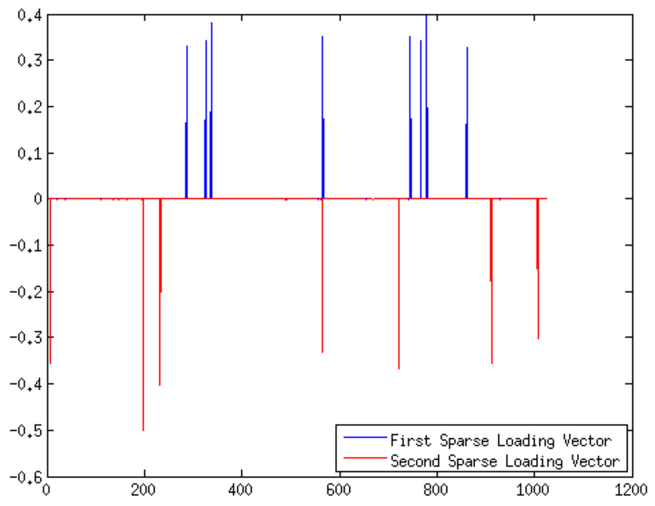

Figure 6.4: The first and second loading vector from the sparse pca of the modularity matrix associated with Figure 6.2, with two embedded subgraphs

Laplacian Eigenmaps. The difference is clear, the sparse version of the embedding sends all nodes not included in one of the subgraphs to zero while both of the subgraphs lie on orthogonal axes. In the traditional PCA embedding, without prior knowledge that there were only two subgraphs, it would be easy to falsely identify some of the smaller green clusters as anomalistic subgraphs.

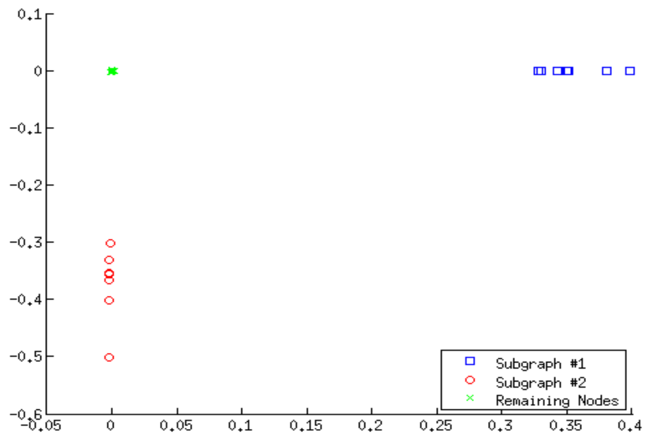

Figure 6.5: A 2D embedding of the graph using the sparse PCA vector rows as coordinates.

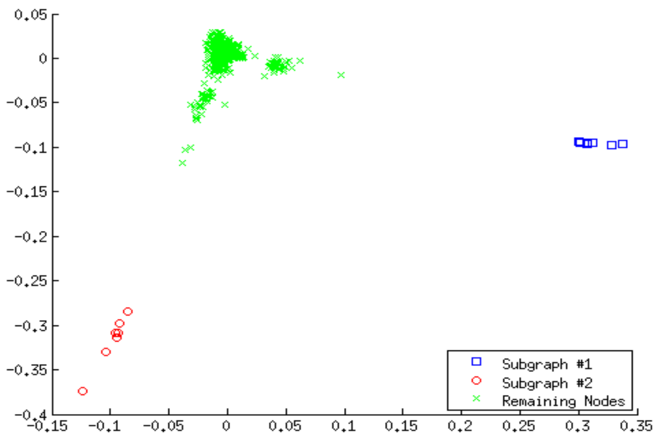

Figure 6.6: A 2D embedding of the graph using the traditional PCA vector rows as coordinates.

6.2 Summary

In this section we gave an overview of Modularity and how it is calculated using the Modularity matrix. Sample networks were explored and the through these sample networks, the need for methods identifying subgraphs in large networks was demonstrated. It was shown that the sparse eigenvectors of the Modularity matrix corresponded to an eight node, and a seven node anomalistic subgraph that had been embedded into a sample network.

Work in this area thus far generally has not used sparsity when looking for solutions, and this is the first work to be able to solve for multiple subgraphs using the Sparse PCA technique. Also, by using the Bregman version of the Sparse PCA algorithm to find the sparse eigenvectors of the modularity matrix additional penalty terms can be added if needed for a given network of data.

7 Conclusions and Future Work

7.1 Conclusions

In this dissertation, we have presented the Split Bregman framework for solving several multivariate analysis problems with applications. The convergence of the Split Bregman algorithm was shown under modest conditions on the terms in the optimization problems, namely convexity. It has been demonstrated in this thesis to easily optimize problems with non-differentiable terms, accept new penalty terms and have promising directions for future research.

The algorithms presented include Sparse PCA, the BSSVD and the BSSVD with an ℓ_1 constrained classifier. All of these algorithms fit the conditions of the Split Bregman algorithm and thus fit nicely into the class of problems that are able to be solved using this technique. For the Sparse PCA, images of faces were used to show how the algorithm selected faces that were linked to the clusters created naturally by the Laplacian Eigenmaps algorithm. For the BSSVD, Hyperspectral Imagery was denoised, allowing clear identification of the plume present in the frames. For the BSSVD with ℓ_1 constrained classifier, Arrhythmia data was used to demonstrate the observation and variable selection done by this technique and was shown via low-dimensional embeddings that it removed observations of interest that geometrically provided a more visually accessible data set. Finally, anomalistic subgraphs were detected using the sparse eigenvectors of a network's modularity matrix.

7.2 Future Work

In terms of future work, there are many options. Firstly, for the multivariate algorithms in which scaling was required, for example in the Sparse PCA where the final answer needed to have unit norm, the parameters for the Split Bregman were very sensitive. This tended to be true in general, but specifically for the multivariate problems, a future direction would be to determine methods for determining optimal penalty parameters.

Also, there is the possibility of improving sparsity or stepsize by allowing change in the penalty parameters as the algorithm iterates. Following the theory in [98] there could be benefit in terms of increased control of sparsity levels and convergence by changing parameters as the algorithm approaches convergence. Next, we can look to add sparsity to existing algorithms such as Laplacian Eigenmaps which has a very similar optimization problem to the modularity problem. We can also expand the data to which these problems are applied. Within modularity, it would be informative to attach hypothesis testing to the detected subgraphs, perhaps using prior information to identify the activity of the nodes as truly anomalistic.

References

[1] D. N. A. Asuncion, *UCI machine learning repository*, 2007.

[2] N. N. Abdelmalek and N. Otsu, *Restoration of images with missing high-frequency components by minimizing the l1 norm of the solution vector*, Appl. Opt., 24 (1985), pp. 1415–1420.

[3] G. Allen, L. Grosenick, and J. Taylor, *A generalized least squares matrix decomposition*, Rice University Technical Report No. TR2011-03, (2011).

[4] C. Bachmann, T. Ainsworth, and R. Fusina, *Exploiting manifold geometry in hyperspectral imagery*, IEEE Transactions on Geoscience and Remote Sensing, 43 (2005), pp. 441–454.

[5] ——, *Improved manifold coordinate representations of large-scale hyperspectral scenes*, IEEE Transactions on Geoscience and Remote Sensing, 44 (2006), pp. 2786–2803.

[6] C. Bachmann, T. Ainsworth, R. Fusina, M. Montes, J. Bowles, D. Korwan, and D. Gillis, *Bathymetric retrieval from hyperspectral imagery using manifold coordinate representations*, IEEE Transactions on Geoscience and Remote Sensing, 47 (2009), pp. 884–897.

[7] BaySpec, *Hyperspectral imaging.* "http://www.bayspec.com". Accessed August 3, 2012.

[8] M. Belkin and P. Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*, J. Amer. Statist. Assoc., 15 (2003), pp. 1373–1396.

[9] V. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, *Fast unfolding of communities in large networks*, Journal of Statistical Mechanics: Theory and Experiment, (2008), pp. 1–12.

[10] J. Boardman, F. Kruse, and R. Green, *Mapping target signatures via partial unmixing of aviris data*, Summaries of the Fifth Annual JPL Airborne Earth Science Workshop. Volume 1: AVIRIS Workshop, 5 (2009), pp. 63–26.

[11] B. Boser, I. Guyon, and V. Vapnik, *A training algorithm for optimal margin classifiers*, Fifth Annual Workshop on Computational Learning Theory, (1992).

[12] P. S. Bradley and O. L. Mangasarian, *Feature Selection via Concave Minimization and Support Vector Machines*, vol. pages, Morgan Kaufmann, 1998, pp. 82–90.

[13] L. M. Bregman, *The relaxation method of finding the commong points of convex sets and its application to the solution of problems in convex optimization*, USSR Comput. Math. and Math. Phys, 7 (1967), pp. 200–217.

[14] M. Burger, G. Gilboa, S. Osher, and J. Xu, *Nonlinear inverse scale space methods*, Communications in Mathematical Sciences, 4 (2006), pp. 175–208.

[15] J. Cai, E. Cands, and Z. Shen, *A singular value thresholding algorithm for matrix completion*, SIAM Journal on Optimization, 20 (2010), pp. 1956–1982.

[16] B. Campbell, *Radar Remote Sensing of Planetary Surfaces*, Cambridge University Press, 2002.

[17] E. Candes and B. Recht, *Exact matrix completion via convex optimization*, Foundations of Computational Mathematics, 9 (2009), pp. 717–772.

[18] E. Candes and T. Tao, *Decoding by linear programming*, IEEE Transactions on Information Theory, 51 (2005), pp. 4203–4215.

[19] E. J. Candes, X. Li, Y. Ma, and J. Wright, *Robust Principal Component Analysis?*, ArXiv e-prints, (2009).

[20] T.-H. Chan, C.-Y. Chi, and Y.-M. Huang, *A convex analysis-based minimum-volume enclosing simplex algorithm for hyperspectral unmixing*, IEEE Transactions on Signal Processing, 57 (2009), pp. 4418–4432.

[21] D. Charkrabarti and C. Faloutsos, *R-MAT : A recursive model for graph mining*, Proc. Fourth SIAM Int'l Conference on Data Mining, 6 (2004), pp. 442–446.

[22] R. Clark, G. Swayze, K. Livo, R. Kokaly, S. Sutley, J. Dalton, R. McDougal, and C. Gent, *Imaging spectroscopy: Earth and planetary remote sensing with the usgs tetracorder and expert systems*, Journal of Geophysical Research, 108 (2003), pp. 5–44.

[23] A. Clauset, N. M, and C. Moore, *Finding community structure in very large networks*, Phys. Rev. E, 70 (2004), p. 066111.

[24] S. Cohen, G. Dror, and E. Ruppin, *A feature selection method based on the Shapley value*, Proceedings of the International Joint Conference on Artificial Intelligence, (2005).

[25] C. Cortes and V. Vapnik, *Support vector networks*, Machine Learning, 20 (1995), pp. 273–297.

[26] A. D'Aspremont, F. Bach, and L. El Ghaoui, *Optimal solutions for sparse principal component analysis*, Journal of Machine Learning Research, 9 (2008).

[27] C. Ding, D. Zhou, X. He, and H. Zha, *R1-pca: Rotational invariant l1 norm principal component analysis for robust subspace factorization*, Proceedings of the 23rd International Conference on Machine Learning, (2006).

[28] D. Donoho, *Compressed sensing*, IEEE Transactions on Information Theory, 52 (2006), pp. 1289–1306.

[29] D. Donoho and J. J, *Ideal spatial adaptation by wavelet shrinkage*, Biometrika, 81 (1994), pp. 425–455.

[30] J. Duch and A. Arenas, *Community detection in complex networks using extremal optimization*, Phys. Rev. E, 72 (2005), p. 027104.

[31] N. Eagle, J. Quinn, and A. Clauset, *Methodologies for continuous cellular tower data analysis*, in Pervasive Computing, H. Tokuda, M. Beigl, A. Friday, A. Brush, and Y. Tobe, eds., vol. 5538 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2009, pp. 342–353.

[32] E. Esser, *Applications of lagrangian-based alternating direction methods and connections to split bregman*, CAM Technical Report, (2009), pp. 1–32.

[33] J. Fan and L. R, *Variable selection via nonconcave penalized likelihood and its oracle properties*, J. Amer. Statist. Assoc., 96 (2001), pp. 1348–1360.

[34] G. Fung, S. Sandilya, and R. R.B., *Rule Extraction from Support Vector Machines*, Springer-Verlag, 2008, pp. 83–107.

[35] D. Gabay and B. Mercier, *A dual algorithm for the solutions of nonlinear variational problems via finite-element approximations*, Comp. Math. Appl., 2 (1976), pp. 17–40.

[36] J. Gao, *Robust l1 principal component analysis and its bayesian variational inference*, Neural Computation, 20 (2008), pp. 555–572.

[37] T. Goldstein, X. Bresson, and S. Osher, *Geometric applications of the split bregman method: Segmentation and surface reconstruction*, Journal of Scientific Computing, 45 (2010), pp. 272–293.

[38] T. Goldstein and S. Osher, *The split bregman method for l1 regularized problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 323–343.

[39] A. Guvenir, B. Acar, G. Demiroz, and A. Cekin, *A supervised machine learning algorithm for arrhythmia analysis*, Computers in Cardiology 1997, (1997), pp. 433–436.

[40] E. T. Hale, W. Yin, and Y. Zhang, *A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing*, CAAM Technical Report, TR07-07 (2007), pp. 1–45.

[41] R. E. Hausman, *Constrained multivariate analysis*, Studies in the Management Sciences, 19 (1982), pp. 137–151.

[42] J. N. R. Jeffers, *Two case studies in the application of principal components*, Applied Statistics, 16 (1967), pp. 225–236.

[43] R. Jenatton, O. Guiallaume, and B. Francis, *Structured sparse principal component analysis*, arXiv.org Machine Learning, (2009).

[44] H. Jiang, S. Fels, and J. Little, *A linear programming approach for multiple object tracking*, Computer Vision and Pattern Recognition, 2007, (2007), pp. 1–8.

[45] I. Jolliffe, *Principal Component Analysis*, Springer-Verlag, 2nd ed., 2002, pp. 150–160.

[46] I. Jolliffe, N. Trendafilov, and M. Uddin, *A modified principal component technique based on the lasso*, Journal of Computational and Graphical Statistics, 12 (2003), pp. 531–547.

[47] M. Kirby, *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns*, John Wiley and Sons, 2001.

[48] M. Kirby and L. Sirovich, *Application of the Karhunen-Loève procedure for the characterization of human faces*, IEEE Trans. Pattern Anal. Mach. Intell., 12 (1990), pp. 103–108.

[49] N. L. Kleinman, N. J. Rohrbacker, S. A. White, J. L. March, and M. R. Reynolds, *Economic impact to employers of treatment options for cardiac arrhythmias in the US health system*, Journal of Occupational and Environmental Medicine, 53 (2011), pp. 405–414.

[50] R. Lawrence, S. Wood, and R. Sheley, *Mapping invasive plants using hyperspectral imagery and breiman cutler classifications (randomforest)*, Remote Sensing of the Environment, 100 (2006), pp. 356–362.

[51] M. Lee, H. Shen, J. Z. Huang, and J. S. Marron, *Biclustering via sparse singular value decomposition*, Biometrics, 66 (2010), pp. 1087–1095.

[52] Z. Lu and Y. Zhang, *An augmented lagrangian approach for sparse principal component analysis*, Mathematical Programming, (2010), pp. 1–45. 10.1007/s10107-011-0452-4.

[53] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and bregman iterative methods for matrix rank minimization*, Mathematical Programming, 128 (2011), pp. 321–353. 10.1007/s10107-009-0306-5.

[54] D. Manalokis, D. Marden, and S. Gary, *Hyperspectral image processing for automatic target detection algorithms*, Lincoln Laboratory Journal, 14 (2003), pp. 79–116.

[55] A. Marquina and S. Osher, *Image super resolution by tv-regularization and bregman iteration*, Journal of Scientific Computing, 37 (2008), pp. 367–382.

[56] N. Meinshausen and P. Buhlmann, *Stability selection*, Journal of the Royal Statistical Society Series B, 72 (2010), pp. 417–473.

[57] B. Miller, N. Bliss, and P. Wolfe, *Subgraph detection using eigenvector l1 norms*, NIPS 2010, (2009).

[58] G. Mohler, A. Bertozzi, T. Goldstein, and S. Osher, *Fast tv regularization for 2d maximum penalized likelihood estimation*, Journal of Computational and Graphical Statistics, 20 (2011), pp. 479–491.

[59] M. Newman, *Detecting community structure in networks*, European Physical Journal B, (2004), pp. 321–330.

[60] ——, *Fast algorithm for detecting community structure in networks*, Physical Review E, (2004), pp. 1–16.

[61] ——, *Finding and evaluating community structure in networks*, Physical Review E, (2004), pp. 1–16.

[62] ——, *Finding community structure in networks using the eigenvectors of matrices*, Physical Review B, (2006).

[63] NHLBI, *Arrhythmia.* "http://www.nhlbi.nih.gov/health/health-topics/topics/arr/". Accessed September 23, 2012.

[64] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An iterative regularization method for total variation-based image restoration*, Multiscale Model. Simul., 4(2) (2005), pp. 460–489.

[65] E. Parkhomenko, *Sparse canonical correlation analysis*, PhD Thesis University of Toronto, (2008).

[66] S. Perkins, K. Lacker, and J. Theiler, *Grafting: Fast, incremental feature selection by gradient descent in function space*, Journal of Machine Learning Research, 3 (2003), pp. 1333–1356.

[67] M. A. Porter, J.-P. Onnela, and P. J. Mucha, *Communities in networks*, Notices of the AMS, (2009), pp. 1082–1097.

[68] J. Reichardt and S. Bornholdt, *Statistical mechanics of community detection*, Phys. Rev. E, 74 (2006), p. 016110.

[69] R. Resmini, M. Kappus, W. Aldrich, J. Harsanyi, and M. Anderson, *Mineral mapping with hyperspectral digital imagery collection experiment (hydice) sensor data at cuprite, nevada, u.s.a.*, International Journal of Remote Sensing, 18 (2010), pp. 1553–1570.

[70] R. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[71] N. J. Rohrbacker, N. L. Kleinman, S. A. White, J. L. March, and M. R. Reynolds, *The burden of atrial fibrillation and other cardiac arrhythmias in an employed population: Associated costs, absences, and objective productivity loss*, Journal of Occupational and Environmental Medicine, 52 (2010), pp. 383–391.

[72] M. Rubinov and O. Sporns, *Complex network measures of brain connectivity: Uses and interpretations*, NeuroImage, 52 (2010), pp. 1059 – 1069.

[73] H. Shen and Z. Huang, *Sparse principal component analysis via regularized low rank matrix approximation*, Journal of Multivariate Analysis, 99 (2008), pp. 1015–1034.

[74] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000), pp. 888–905.

[75] N. Shibata, Y. Kajikawa, Y. Takeda, I. Sakata, and K. Matsushima, *Detecting emerging research fronts in regenerative medicine by the citation network analysis of scientific publications*, Technological Forecasting and Social Change, 78 (2011), pp. 274 – 282.

[76] M. Sill, S. Kaiser, A. Benner, and A. Kopp-Schneider, *Robust biclustering by sparse singular value decomposition incorporating stability selection*, bioinformatics, 27 (2011), pp. 2089–2097.

[77] N. Singh, B. Miller, N. Bliss, and P. Wolfe, *Anomalous subgraph detection via sparse principal component analysis*, IEEE Statistical Signal Processing Workshop, (2011), pp. 485–488.

[78] L. Sirovich and M. Kirby, *A low-dimensional procedure for the characterization of human faces.*, J. of the Optical Society of America A, 4 (1987), pp. 524–529.

[79] J. Stehle, N. Voirin, and A. Barrat et al., *Simulation of an seir infections disease model on the dynamic contact network of conference attendees*, BMC Medicine, 9 (2011), p. 87.

[80] A. Szlam, G. Zhaohui, and S. Osher, *A split bregman method for non-negative sparsity penalized least squares with applications to hyperspectral demixing*, Image Processing (ICIP), 2010 17th IEEE International Conference on, (2010), pp. 1917–1920.

[81] J. T. Tenenbaum, V. de Silva, and J. C. Langford, *A global geometric framework for nonlinear dimensionality reduction*, Science, 290 (2000), pp. 2319–2323.

[82] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society– Series B, 58 (1996), pp. 267–288.

[83] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, *Sparsity and smoothness via the fused lasso*, Journal of the Royal Statistical Society Series B, 67 (2005), pp. 91–108.

[84] A. L. Traud, P. J. Mucha, and M. A. Porter, *Social structure of facebook networks*, Physica A: Statistical Mechanics and its Applications, 391 (2012), pp. 4165 – 4180.

[85] A. Tsonis, G. Wang, K. Swanson, F. Rodrigues, and L. Costa, *Community structure and dynamics in climate networks*, Climate Dynamics, 37 (2011), pp. 933–940.

[86] E. Underwood, S. Ustin, and D. Dipietro, *Mapping nonnative plants using hyperspectral imagery*, Remote Sensing of Environment, 86 (2003), pp. 150–161.

[87] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.

[88] S. Vines, *Simple principal components*, Journal of the Royal Statistical Society– Series C, 49 (2000), pp. 441–451.

[89] L. Wang, J. Zhu, and H. Zou, *The double regularized support vector machine*, Statistic Sinica, 16 (2006), pp. 589–615.

[90] Y. Wang, J. Yang, W. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM Journal on Imaging Sciences, 1 (2008), pp. 248–272.

[91] S. White and P. Smyth, *A spectral clustering approach to finding communities in graphs*, Proceedings of the 5th SIAM International Conference, (2005), pp. 274–285.

[92] M. Winter, *N-findr: An algorithm for fast autonomous spectral end-member determination in hyperspectral data*, Proceedings of SPIE - The International Society for Optical Engineering. Vol. SPIE-3753, (1999), pp. 266–275.

[93] D. M. Witten, R. Tibshirani, and T. Hastie, *A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis*, Biostatistics, 10 (2009), pp. 515–534.

[94] J. Xu and S. Osher, *Iterative regularization and nonlinear inverse scale space applied to wavelet-based denoising*, IEEE Transactions on Image Processing, 16 (2006), pp. 534–544.

[95] J. Yang, D. Zhang, A. Frangi, and J.-y. Yang, *Two-dimensional pca: A new approach to appearance-based face representation and recognition*, IEEE Transactions on Patter Analysis and Machine Intelligence, 26 (2004), pp. 131–137.

[96] G. Ye and X. Xie, *Split bregman method for large scale fused lasso*, Computational Statistics and Data Analysis, 55(4) (2011), pp. 1552–1569.

[97] G.-B. Ye, Y. Chen, and X. Xie, *Efficient variable selection in support vector machines via the alternating direction method of multipliers*, Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, 15 (2011), pp. 832–840.

[98] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for l_1-minimization with applications to compressed sensing*, SIAM Journal of Imaging Sciences, 1 (2008), pp. 143–168.

[99] X. Zhang, M. Burger, and S. Osher, *A unified primal-dual algorithm framework based on bregman iteration*, Journal of Scientific Computing, 46 (2011), pp. 20–46.

[100] H. Zou, *The adaptive lasso and its oracle properties*, Journal of the American Statistical Association, 101 (2006), pp. 1418–1429.

[101] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, Journal of the Royal Statistical Society Series B, 67 (2005), pp. 301–320.

[102] H. Zou, T. Hastie, and R. Tibshirani, *Sparse principal component analysis*, Journal of Computational and Graphical Statistics, 15 (2006), pp. 265–286.

A Definitions

Definition 1. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is defined to be* ***convex*** *if for any convex set V in the domain of f and for any $x, y \in V$ we have that $f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$ where $t \in [0, 1]$. Equivalently, if f is twice differentiable, f is convex if the Hessian of f is positive semidefinite.*

Definition 2. *A function f is a* ***nonnegative convex functional*** *if $f : \mathbb{R}^n \to \mathbb{R}$, meaning f has a range of the real numbers, is convex and is nonnegative on its domain.*

Definition 3. *A vector g is called a* ***subgradient*** *of f at y if for any $x \in U$, we have that*

$$f(x) - f(y) - (g(y), x - y) \geq 0$$

The set of all subgradients of f at y is called the ***subdifferential*** *of f at y. If the subdifferential contains only one element, the function is differentiable at y and the subgradient is equal to the gradient.*

Below is the algorithm for constructing the data needed to test the classifiers on separable data as footnoted in Chapter 5. The data is moved away from the origin and overlaps by construction.

Non-Separable Data Construction

```
1: for $i = 1 : 10$ do
2:     p = i*100;
3:     c1 = rand(250,10) + 2 - i/10;
4:     c2 = -rand(250,10) + 2.25;
5:     c3 = rand(500,p - 10);
6:     X = [c1; c2];
7:     X = [X c3];
8:     y = [ones(250,1); -ones(250,1)];
9: end for
```