A Hyperplane-based Algorithm for Semi-supervised Dimension Reduction

Huang Fang
Dept. of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4, Canada
Email: hgfang@cs.ubc.ca

Minhao Cheng
Dept. of Computer Science
University of California, Davis
Davis, CA, 95616, USA
Email: mhcheng@ucdavis.edu

Cho-Jui Hsieh
Depts. of Statistics and Computer Science
University of California, Davis
Davis, CA, 95616, USA
Email: chohsieh@ucdavis.edu

***Abstract*—We consider the semi-supervised dimension reduction problem: given a high dimensional dataset with a small number of labeled data and huge number of unlabeled data, the goal is to find the low-dimensional embedding that yields good classification results. Most of the previous algorithms for this task are linkage-based algorithms. They try to enforce the must-link and cannot-link constraints in dimension reduction, leading to a nearest neighbor classifier in low dimensional space. In this paper, we propose a new hyperplane-based semi-supervised dimension reduction method—the main objective is to learn the low-dimensional features that can both approximate the original data and form a good separating hyperplane. We formulate this as a non-convex optimization problem and propose an efficient algorithm to solve it. The algorithm can scale to problems with millions of features and can easily incorporate non-negative constraints in order to learn interpretable non-negative features. Experiments on real world datasets demonstrate that our hyperplane-based dimension reduction method outperforms state-of-art linkage-based methods when very few labels are available.**

***Keywords*—Semi-supervised learning, Dimension reduction, optimization**

I. Introduction

Real world datasets, such as images and documents, are usually unstructured and have very high dimensionality. Dimension reduction methods, which aim to extract low-dimensional informative features from high dimensional data, thus become an important technique for data analytics. Most dimension reduction methods are either unsupervised or supervised. Unsupervised dimension reduction methods, such as Principal Component Analysis [1] do not use any label information, so they aim to project data to a lower dimensional space that preserves the distances or data geometric. Supervised methods such as Fisher Linear Discriminant [2] require to have class labels for each sample during the training process, and the goal is to find a low dimensional space that maximizes the split between labels.

However, many real word datasets are "partially" labeled since labeled data are expensive to get, and this is especially true when number of samples is large and getting labels for all of them is too expensive. In this scenario, we cannot use supervised methods since labels are only given for a small subset of samples. On the other hand, using purely unsupervised methods will lose all the label information. For those partially labeled data, a family of "semi-supervised dimension reduction algorithms" [4], [21] naturally comes into play, where they aim to make use of all available labeled and unlabeled information for dimension reduction.

In the literature of semi-supervised dimension reduction, supervision is usually given in the form of pairwise constraints, known as "must-links" and "cannot-links" between pairs of samples [20], [4], [21]. Two samples are linked by "must-link" if they are in the same class, and "cannot-link" if they belong to different classes. By adding these constraints to the optimization problem [4], samples belong to the same class will tend to be clustered together.

However, this formulation is built on the idea of clustering and may be less favourable from the perspective of forming a separating hyperplane in the lower dimensional space. Take spam email classification as an example, spam emails may have very different forms and they are probably not close to each other in the low-dimensional space. In this case, separating hyperplane-based methods such as Support Vector Machine(SVM) is more promising. Moreover, Linkage-based methods could be viewed as a combination of nearest neighbour classification and dimension reduction, while k-Nearest Neighbours(kNN) tends to be unstable when number of data is relatively small.

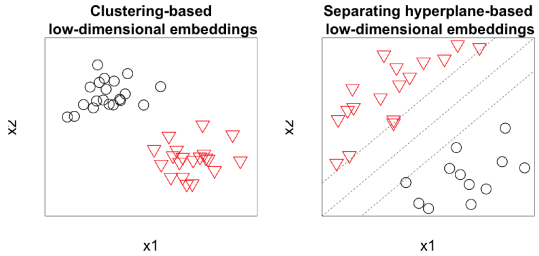


Fig. 1: Ideal Low-dimensional embeddings: Linkage-based method v.s. Hyperplane-based method

In this paper, instead of adding pairwise constraints, we propose a novel non-convex optimization formulation to incorporate dimensional reduction with linear support vector classification, in which we consider both L1-hinge loss and L2-hinge loss. The main objective of our method is to learn the low-dimensional features that could approximate the original high-dimensional data and form a good separating hyperplane simultaneously. Figure 1 illustrates the difference between previous linkage-based methods and our hyperplane-based method (a more straightforward illustration is presented in V-D). The non-convex optimization problem can be solved efficiently and the convergence to stationary points can be guaranteed.

Furthermore, for datasets whose elements are all non-negative, such as images and texts, we can easily incorporate the non-negative constraints into our formulation. This leads to better interpretability of the low-dimensional features. Previous studies [17], [18] have demonstrated that non-negativity is important for these applications, and in this case our model becomes a combiniation of SVM and Non-negative Matrix Factorization(NMF), or can also be viewed as a semi-supervised NMF method.

Our contributions can be summarized below:

- We propose a novel hyperplane-based semi-supervised dimension reduction approach to learn the low-dimensional embedding and separating hyperplane simultaneously.
- We design an efficient block coordinate descent algorithm to solve the problem, and the algorithm can be easily scaled to datasets with millions of features. Moreover, our algorithm is quite flexible so it is easy to incoporate non-negative constraints and learn interpretable features for non-negative data.
- For problems with many unlabeled data and few labeled data, we show that our proposed method can get a low dimensional embedding with higher classification accuracy compared with other state-of-art semi-supervised dimension reduction methods.

II. Related Work

Dimension reduction is an important technique for large-scale and high dimensional data analytic. Traditionally, dimension reduction is conducted in an unsupervised way. Given a set of n samples with m features, Principal Component Analysis(PCA) is used to reduce the dimensionality from m to k by conducting SVD decomposition on the feature matrix. Several other techniques have also been proposed to enforce specific structures in the lower dimensional space; for example NMF [14] is used to get the non-negativity and ICA [22] can be used to split mixed signals. Matrix completion and inductive matrix completion [10] are used for partially observed data, and robust PCA [9] can be used to deal with corrupted data matrix.

A. Semi-supervised dimension reduction

The main focus of this paper is to study how to conduct dimension reduction with little supervision. With some small but useful supervised information, it is possible to guide or bias the dimension reduction into what we expect. A common way to encode supervision is by providing a list of "must-link" (similar) and "cannot-link" (dissimilar), which are pairwise constraints between instances [20]. In this setting, the instance pairs with must-link should be close to each other while pairs with cannnot-link should be far away in the low dimensional space after dimension reduction.

SSDR [4] uses these pairwise constraints generated from supervised label to create a Laplacian graph to guide the dimension reduction process. GCDR-LP [21] uses a similar idea as [4] but allows different weights on observed must-links and cannot-links. However, these methods are graph-based algorithms and may not perform well in the SVM setting. We have compared with them in the experimental results.

For non-negative data, several methods including [6] and [16] have been proposed to improve conventional non-negative matrix factorization by making use of labeled data. Experimental results in [6], [16] demonstrated that their methods work well for document clustering problem. However, our numerical experiments show that the performance of non-negative dimension reduction is more sensitive to the setting of parameters and it is generally hard to conduct cross validation with only few labeled data. Therefore, in this paper we only point out the potential connection between these non-negative dimension reduction methods and our proposed method and exclude non-negative constraints in the experiments.

B. Fasthals

Fasthals [5] is one of state-of-art algorithms for large-scale non-negative matrix factorization. It breaks the original low-rank approximation into k rank-one approximation subproblems and each subproblem can be solved efficiently. Although it is an unsupervised dimension reduction algorithm, in this work we show that the idea of *Fasthals* can also be used to solve our semi-supervised low-dimensional approximation problem and help us to develop a scalable algorithm.

III. Notations

The notations used in this paper is summarized in Table I. Note that our goal is to conduct low-rank approximation of X by $X \approx BH$ and learn the low-dimensional separating hyperplane $\{x : \boldsymbol{w}^T x = 0\}$ simultaneously. As shown in Figure 2, the first ntr samples in X are labeled samples, and the rest are unlabeled samples. The prediction of our model for the unlabeled data j will be $\text{sign}(\boldsymbol{w}^T \boldsymbol{h}_j)$.

$X \in \mathbb{R}^{m \times n}$	Data matrix
m	Number of features
n	Number of samples
ntr	Number of labeled samples
nte	Number of unlabeled samples
k	Reduced number of dimensions
$Y \in \mathbb{R}^{ntr \times 1}$	Labels
$X_l \in \mathbb{R}^{m \times ntr}$	Labeled data matrix
$X_u \in \mathbb{R}^{m \times nte}$	Unlabeled data matrix
$B \in \mathbb{R}^{m \times k}$ or $\mathbb{R}_+^{m \times k}$	Basis matrix in low-dimensional space
$\boldsymbol{b}_i \in R^{m \times 1}$	The i-th column of B
$H \in \mathbb{R}^{k \times n}$ or $\mathbb{R}_+^{k \times n}$	Low-dimensional embedding
$\boldsymbol{h}_i \in R^{k \times 1}$	The i-th column of H
$H_l \in \mathbb{R}^{k \times ntr}$ or $\mathbb{R}_+^{k \times ntr}$	Labeled Low-dimensional embedding
$H_u \in \mathbb{R}^{k \times nte}$ or $\mathbb{R}_+^{k \times nte}$	Unlabeled Low-dimensional embedding
$\boldsymbol{h}_{l,i}^T \in R^{ntr \times 1}$	The i-th row of H_l
$H_l^{(j)} \in \mathbb{R}^{(k-1) \times ntr}$	H_l without the j-th row
$\boldsymbol{w} \in \mathbb{R}^{k \times 1}$	Parameters for the separating hyperplane
$\boldsymbol{w}^{(j)} \in \mathbb{R}^{(k-1) \times 1}$	$\boldsymbol{w}$ without the j-th element
$\lambda_1, \lambda_2, \lambda_3$	Weights Parameters
$\odot$	Element-wise product
$\oslash$	Element-wise division

TABLE I: Notations used in this paper.

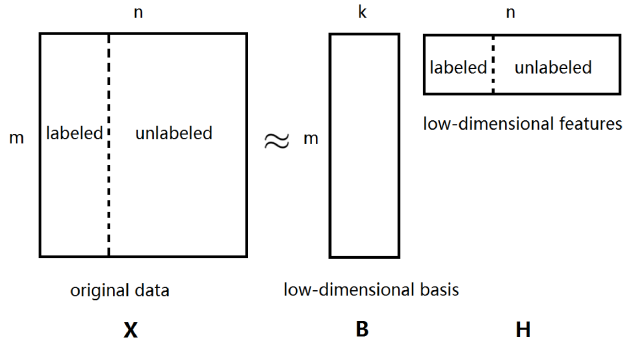


Fig. 2: The structure of our data, B is the low-dimensional basis, H is the low-dimensional features.

IV. Proposed Method

In this section, we introduce our proposed method for semi-supervised dimension reduction given a subset of instance labels. The main idea is to combine the objective function of dimension reduction with linear support vector machine, and solve the optimization problems jointly. Our algorithm is called Dimension Reduction SVM (DRSVM).

A. Mathematical Formulation

The basic idea is to combine the objective function of SVM and PCA (SVD) together, which lead to the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{w},B,H} & \sum_{i=1}^{ntr} l(y_i, \boldsymbol{w}^T \boldsymbol{h}_i) + \frac{\lambda_1}{2}\|\boldsymbol{w}\|_2^2 + \frac{\lambda_2}{2}\|X - BH\|_F^2 \\ & + \frac{\lambda_3}{2}(\|B\|_F^2 + \|H\|_F^2) \\ \textbf{s.t.} \quad & B \geq 0, \quad H \geq 0 \quad (Optional) \end{aligned} \tag{IV.1}$$

In problem (IV.1), λ_1, λ_2 and λ_3 are parameters to control the weights of each part. The optional constraints $B \geq 0$, $H \geq 0$ are added only if we want to enforce the non-negativity in the low-dimensional feature space (e.g., useful in the NMF setting). The third term in objective function ($\|X - BH\|_F^2$) is used for dimension reduction (it is the loss for SVD or PCA), where the i-th column of H will be a low-dimensional embedding of the i-th sample. The first two terms are the SVM objective function that corresponds to label information. Overall, the goal of solving problem (IV.1) is to learn low-dimensional embedding h_i which can lead to small classification error. $\boldsymbol{w}$ is the classification model obtained from DRSVM.

In this paper, we consider L1-hinge and L2-hinge loss as our loss functions, where $l_1(y_i, \boldsymbol{w}^T \boldsymbol{h}_i) = \max\{0, 1 - y_i \boldsymbol{w}^T \boldsymbol{h}_i\}$ for L1-hinge loss and $l_2(y_i, \boldsymbol{w}^T \boldsymbol{h}_i) = \max\{0, 1 - y_i \boldsymbol{w}^T \boldsymbol{h}_i\}^2$ for L2-hinge loss.

Note that the above formulation works only for binary classification. We can generalize it to multi-class classification based on the One-versus-Rest strategy, and the resulting formulation will be:

$$\begin{aligned} \min_{\boldsymbol{w},B,H} & \sum_{j=1}^{r} \sum_{i=1}^{ntr} l(y_{ij}, \boldsymbol{w}_j^T \boldsymbol{h}_i) + \frac{\lambda_1}{2} \sum_{j=1}^{r} \|\boldsymbol{w}_j\|_2^2 + \frac{\lambda_2}{2}\|X - BH\|_F^2 \\ & + \frac{\lambda_3}{2}(\|B\|_F^2 + \|H\|_F^2) \\ \textbf{s.t.} \quad & B \geq 0, \quad H \geq 0 \quad (Optional) \end{aligned} \tag{IV.2}$$

where r is the total number of classes and y_{ij} is constructed by the standard One-versus-Rest strategy.

B. Optimization

In this section, we focus on how to solve the minimization problem (IV.1) efficiently. Clearly, (IV.1) is non-convex. In the literature of non-convex optimization, *Block Coordinate Descent* (BCD) [3] is a widely used technique and has been shown to work well in the field of non-negative matrix factorization [5], [8]. This motivates us to adopt the framework BCD to solve (IV.1).

The concept of *Block Coordinate Descent* is simple and straightforward. For the optimization problem:

$$\min f(\boldsymbol{x}) \quad \textbf{s.t.} \quad \boldsymbol{x} \in \mathcal{X}, \tag{IV.3}$$

where $\mathcal{X} \subseteq \mathbb{R}^N$ and $\mathcal{X}$ is convex. Assume $\mathcal{X}$ can be decomposed into the Cartesian product of L convex subset such that $\mathcal{X} = \bigtimes_{j=1}^{L} \mathcal{X}_j$ where $\mathcal{X}_j \subseteq \mathbb{R}^{N_j}$ and $\mathcal{X}_j$ is closed and convex and $N = \sum_{i=1}^{L} N_j$. Also we denote the partition of $\boldsymbol{x}$ by $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_L]$. BCD tries to update $\boldsymbol{x}_j$ block by block in cyclic order by the following rule until convergence:

$$\boldsymbol{x}_j^{(t+1)} = \arg\min_{\xi \in \mathcal{X}_j} f(\boldsymbol{x}_1^{(t+1)}, \boldsymbol{x}_2^{(t+1)}, ..., \boldsymbol{x}_{j-1}^{(t+1)}, \xi, \boldsymbol{x}_{j+1}^{(t)}, ..., \boldsymbol{x}_L^{(t)}) \tag{IV.4}$$

Lemma 1. *Assume f is continuously differentiable over the set $\mathcal{X}$. Furthermore, suppose that for each j and $\boldsymbol{x} \in \mathcal{X}$, the minimum of*

$$\arg\min_{\xi \in \mathcal{X}_j} f(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{j-1}, \xi, \boldsymbol{x}_{j+1}, ..., \boldsymbol{x}_L)$$

is uniquely attained. Let $\boldsymbol{x}^{(k)}$ be the sequence generated by the update rule IV.4. Then every limit point is a stationary point.

The proof and details of the above Lemma can be found in [3].

Difficulties: In our optimization problem, the most straightforward way is to use 3 blocks, B, H and $\boldsymbol{w}$, and alternatively updating each of them. However, the subproblem with respect to H is hard to solve since H is in both first term and third term of the objective function. Furthermore, this 3-block approach cannot easily incorporate non-negative constraints. Another intuitive way is to optimize H column by column since each column of H is independent to each other, and the corresponding subproblem is:

$$\min_{\boldsymbol{h}_i} l(y_i, \boldsymbol{w}^T \boldsymbol{h}_i) + \|\boldsymbol{x}_i - B\boldsymbol{h}_i\|_2^2 + \text{const}, \quad 0 < i \leq ntr.$$

However, this subproblem cannot be solved exactly (needs another inner loop to solve) when our loss function is L1-hinge loss or L2-hinge loss.

Proposed update rule: In the following section, we show that updating one row of H at a time is the most efficient way to solve DRSVM optimization problem, and we will develop an efficient algorithm based on this.

For simplicity, we denote

$$f(\boldsymbol{w}, B, H) = \sum_{i=1}^{ntr} l(y_i, \boldsymbol{w}^T \boldsymbol{h}_i) + \frac{\lambda_1}{2}\|\boldsymbol{w}\|_2^2 + \frac{\lambda_2}{2}\|X - BH\|_F^2 + \frac{\lambda_3}{2}(\|B\|_F^2 + \|H\|_F^2) \tag{IV.5}$$

and

$$B = [\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_k], \quad H = [\boldsymbol{h}_{\cdot 1}, \boldsymbol{h}_{\cdot 2}, ..., \boldsymbol{h}_{\cdot k}]^T.$$

We propose to update one column of B and one row of H at a time, and then after the updates of B and H, we can fix them and update the vector $\boldsymbol{w}$. Therefore the overall update sequence will be in the following order:

$$\boldsymbol{b}_1 \to \boldsymbol{b}_2 \to ... \to \boldsymbol{b}_k \to \boldsymbol{h}_{\cdot 1} \to \boldsymbol{h}_{\cdot 2} \to ... \to \boldsymbol{h}_{\cdot k} \to \boldsymbol{w}$$

Note that this order is inspired by the HALS algorithm [5], which is widely used in NMF. Next we discuss the detailed update rules and time complexity for each block of variables.

1) Updating B: First, we consider the subproblem with respect to $\boldsymbol{b}_j$:

$$\min_{\boldsymbol{b}_j} f(w, [\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_k], H) \quad \Leftrightarrow$$

$$\min J_B^{(j)}(\boldsymbol{b}_j) = \frac{\lambda_2}{2}\|X - \sum_{i=1}^{k} \boldsymbol{b}_i \boldsymbol{h}_{\cdot i}^T + \boldsymbol{b}_j \boldsymbol{h}_{\cdot j}^T - \boldsymbol{b}_j \boldsymbol{h}_{\cdot j}^T\|_F^2 + \frac{\lambda_3}{2}\|\boldsymbol{b}_j\|_2^2$$

Denote $X^{(j)} = X - \sum_{i=1}^{k} \boldsymbol{b}_i \boldsymbol{h}_{\cdot i}^T + \boldsymbol{b}_j \boldsymbol{h}_{\cdot j}^T$

$$\min J_B^{(j)}(\boldsymbol{b}_j) = \frac{\lambda_2}{2}\|X^{(j)} - \boldsymbol{b}_j \boldsymbol{h}_{\cdot j}^T\|_F^2 + \frac{\lambda_3}{2}\|\boldsymbol{b}_j\|_2^2$$

When $\boldsymbol{h}_{\cdot j}$ is fixed, $J_B^{(j)}(\boldsymbol{b}_j)$ is a convex function for $\boldsymbol{b}_j$, so we can solve it by setting $\nabla_{\boldsymbol{b}_j} J_B^{(j)} = 0$ to get the optimal solution $\boldsymbol{b}_j^*$:

$$\nabla_{\boldsymbol{b}_j} J_B^{(j)} = \lambda_2(\|\boldsymbol{h}_{\cdot j}\|_2^2 \boldsymbol{b}_j - X^{(j)} \boldsymbol{h}_{\cdot j}) + \lambda_3 \boldsymbol{b}_j = 0$$

$$\boldsymbol{b}_j^* = \frac{X^{(j)} \boldsymbol{h}_{\cdot j}}{\|\boldsymbol{h}_{\cdot j}\|_2^2 + \frac{\lambda_3}{\lambda_2}} = \frac{(X - BH + \boldsymbol{b}_j \boldsymbol{h}_{\cdot j}^T)\boldsymbol{h}_{\cdot j}}{\|\boldsymbol{h}_{\cdot j}\|_2^2 + \frac{\lambda_3}{\lambda_2}} = \frac{[XH^T]_j - B[HH^T]_j + \boldsymbol{b}_j \|\boldsymbol{h}_{\cdot j}\|_2^2}{\|\boldsymbol{h}_{\cdot j}\|_2^2 + \frac{\lambda_3}{\lambda_2}} \tag{IV.6}$$

When the non-negative constraint $\boldsymbol{b}_j \geq 0$ is added, we can split $\boldsymbol{b}_j = [b_{j,1}, b_{j,2}, ..., b_{j,m}]^T$, and write $J_B^{(j)}(\boldsymbol{b}_j)$ as:

$$\sum_{i=1}^{m} \{\frac{\lambda_2}{2}\|X_{i\cdot}^{(j)} - b_{j,i} \boldsymbol{h}_{\cdot j}^T\|_F^2 + \frac{\lambda_3}{2} b_{j,i}^2\},$$

where $X_{i\cdot}^{(j)}$ denote the ith row of $X^{(j)}$.

From this decomposition, we can see that every components $b_{j,i}$ in $\boldsymbol{b}_j$ are "independent" to each other, and each part $\frac{\lambda_2}{2}\|X_{i\cdot}^{(j)} - b_{j,i} \boldsymbol{h}_{\cdot j}^T\|_F^2 + \frac{\lambda_3}{2} b_{j,i}^2$ is just a quadratic optimization problem of the one-dimensional variable $b_{j,i}$.

Then it is easy to see that if

$$b_{j,i}^* = \arg\min_{b_{j,i}} \frac{\lambda_2}{2}\|X_{i\cdot}^{(j)} - b_{j,i} \boldsymbol{h}_{\cdot j}^T\|_F^2 + \frac{\lambda_3}{2} b_{j,i}^2$$

then the optimal solution after adding the non-negative constraints will be

$$\max\{0, b_{j,i}^*\} = \arg\min_{b_{j,i} \geq 0} \frac{\lambda_2}{2}\|X_{i\cdot}^{(j)} - b_{j,i} \boldsymbol{h}_{\cdot j}^T\|_F^2 + \frac{\lambda_3}{2} b_{j,i}^2$$

This implies that

$$\{\boldsymbol{b}_j^*\}_+ = \arg\min_{\boldsymbol{b}_j \geq 0} J_B^{(j)}(\boldsymbol{b}_j),$$

where $\boldsymbol{b}_j^*$ is the solution of Eq. (IV.6).

To sum up, for updating $\boldsymbol{b}_j$, the optimal solution can be computed by (IV.6) without nonnegative constraints, and furthermore we just need a simple element-wise projection even after adding the non-negative constraints.

2) Updating H: We split H into $[H_l, H_u]$ and X into $[X_l, X_u]$, where H_l, X_l correspond to instances with seen labels (see Figure 2).

$$\begin{aligned}&\min_H f(w, B, [H_l, H_u]) \quad \Leftrightarrow \\ &(\min_{H_l} \sum_{i=1}^{ntr} l(y_i, \boldsymbol{w}^T \boldsymbol{h}_i) + \frac{\lambda_2}{2}\|X_l - BH_l\|_F^2 + \frac{\lambda_3}{2}\|H_l\|_F^2) \\ &+ (\min_{H_u} \frac{\lambda_2}{2}\|X_u - BH_u\|_F^2 + \frac{\lambda_3}{2}\|H_u\|_F^2)\end{aligned} \tag{IV.7}$$

For the second part:

$$\begin{aligned}&\min_{H_u} \frac{\lambda_2}{2}\|X_u - BH_u\|_F^2 + \frac{\lambda_3}{2}\|H_u\|_F^2 \\ =&\min_{H_u^T} \frac{\lambda_2}{2}\|X_u^T - H_u^T B^T\|_F^2 + \frac{\lambda_3}{2}\|H_u^T\|_F^2\end{aligned}$$

The update of H_u is essentially the same as the update of B, so here we can get the update formula as follows:

Denote $H_u = [\boldsymbol{h}_{u\cdot 1}, \boldsymbol{h}_{u\cdot 2}, ..., \boldsymbol{h}_{u\cdot k}]^T$, then we have

$$\boldsymbol{h}_{u\cdot j}^* = \frac{[X_u^T B]_j - H_u^T [B^T B]_j + \boldsymbol{h}_{u\cdot j}\|\boldsymbol{b}_j\|_2^2}{\|\boldsymbol{b}_j\|_2^2 + \frac{\lambda_3}{\lambda_2}}, \tag{IV.8}$$

and if the non-negative constraints are added, we have

$$\boldsymbol{h}_{u\cdot j}^* = \left\{ \frac{[X_u^T B]_j - H_u^T [B^T B]_j + \boldsymbol{h}_{u\cdot j}\|\boldsymbol{b}_j\|_2^2}{\|\boldsymbol{b}_j\|_2^2 + \frac{\lambda_3}{\lambda_2}} \right\}_+ .$$

The update of H_l is more sophisticated and it is an important step since it is connected with both the supervised learning model and dimension reduction.

Denote $H_l = [\boldsymbol{h}_{l\cdot 1}, \boldsymbol{h}_{l\cdot 2}, ..., \boldsymbol{h}_{l\cdot k}]^T$

Fix $\boldsymbol{h}_{l\cdot 1}, \boldsymbol{h}_{l\cdot 2}, ..., \boldsymbol{h}_{l\cdot j-1}, \boldsymbol{h}_{l\cdot j+1}, ..., \boldsymbol{h}_{l\cdot k}$ and minimize Eq (IV.7) over $\boldsymbol{h}_{l\cdot j}$ we get

$$\begin{aligned}&\min_{\boldsymbol{h}_{l\cdot j}} f(\boldsymbol{w}, B, H) \Leftrightarrow \\ &\sum_{i=1}^{ntr} l(y_i, \boldsymbol{w}^{(j)T} \boldsymbol{h}_i^{(j)} + \boldsymbol{w}_j h_{j,i}) + \frac{\lambda_2}{2}\|X_l^{(j)} - \boldsymbol{b}_j \boldsymbol{h}_{l\cdot j}\|_F^2 + \frac{\lambda_3}{2}\|\boldsymbol{h}_{l\cdot j}\|_2^2 \\ &= \sum_{i=1}^{ntr} \left\{ l(y_i, \boldsymbol{w}^{(j)T} \boldsymbol{h}_i^{(j)} + \boldsymbol{w}_j h_{j,i}) + \frac{\lambda_2}{2}\|X_{li}^{(j)} - \boldsymbol{b}_j h_{j,i}\|_2^2 + \frac{\lambda_3}{2} h_{j,i}^2 \right\}\end{aligned}$$

This decomposition breaks the original problem into ntr "independent" subproblems. In addition to L1-hinge and L-2 hinge loss, this strategy is also suitable for other loss functions as long as they are *linear* predictors. This is the reason why *HALS*'s framework can be generalized to our semi-supervised scenario.

Under L-1 hinge loss

For each subproblem:

$$\begin{aligned}\min_{h_{j,i}} D_{ji}(h_{j\cdot i}) :=& \max(0, 1 - y_i(\boldsymbol{w}^{(j)T} \boldsymbol{h}_i^{(j)} + \boldsymbol{w}_j h_{j,i})) \\ &+ \frac{\lambda_2}{2}\|X_{li}^{(j)} - \boldsymbol{b}_j h_{j,i}\|_2^2 + \frac{\lambda_3}{2} h_{j,i}^2\end{aligned} \tag{IV.9}$$

Denote $\mathcal{S}_+ = \{h_{j,i} : 1 - y_i(\boldsymbol{w}^{(j)T} \boldsymbol{h}_i^{(j)} + \boldsymbol{w}_j h_{j,i}) \geq 0\}, \mathcal{S}_- = \{h_{j,i} : 1 - y_i(\boldsymbol{w}^{(j)T} \boldsymbol{h}_i^{(j)} + \boldsymbol{w}_j h_{j,i}) \leq 0\}$ and $\tau = \mathcal{S}_+ \cap \mathcal{S}_- = \{h_{j,i} : 1 - y_i(\boldsymbol{w}^{(j)T} \boldsymbol{h}_i^{(j)} + \boldsymbol{w}_j h_{j,i}) = 0\}$

Lemma 2. *When $\boldsymbol{w}_j \neq 0$, let $h_{j,i}^{*+} = \arg\min_{h_{j,i} \in \mathcal{S}_+} D_{ji}(h_{j,i})$ and $h_{j,i}^{*-} = \arg\min_{h_{j,i} \in \mathcal{S}_-} D_{ji}(h_{j,i})$. Then at least one of $h_{j,i}^{*+}$ and $h_{j,i}^{*-}$ equals to τ. If $h_{j,i}^{*+} = h_{j,i}^{*-} = \tau$, the solution for Eq. IV.16 is τ. Otherwise, the solution is the one that not equals to τ.*

Proof. Since $\max(0, 1 - y_i(\boldsymbol{w}^{(j)T} \boldsymbol{h}_i^{(j)} + \boldsymbol{w}_j h_{j,i})$ a convex function of $h_{j,i}$ and $\frac{\lambda_2}{2}\|X_{li}^{(j)} - \boldsymbol{b}_j h_{j,i}\|_2^2 + h_{j,i}^2$ is strictly convex, so D_{ji} is a strictly convex function of $h_{j,i}$. Obviously $\mathcal{S}_+$ and $\mathcal{S}_-$ are convex sets, so $h_{j,i}^{*+} = \arg\min_{h_{j,i} \in \mathcal{S}_+} D_{ji}(h_{j,i})$ and $h_{j,i}^{*-} = \arg\min_{h_{j,i} \in \mathcal{S}_-} D_{ji}(h_{j,i})$ exist and they are unique.

Assume that $h_{j,i}^{*+} \neq \tau$ and $h_{j,i}^{*-} \neq \tau$, then there exist $\lambda > 0$, s.t. $\tau = \lambda h_{j,i}^{*+} + (1 - \lambda) h_{j,i}^{*-}$, since $D_{ji}(h_{j,i}^{*+}) \leq D_{ji}(\tau)$ and $D_{ji}(h_{j,i}^{*-}) \leq D_{ji}(\tau)$, we have $\lambda D_{ji}(h_{j,i}^{*+}) + (1-\lambda) D_{ji}(h_{j,i}^{*-}) \leq D_{ji}(\tau) = D_{ji}(\lambda D_{ji}^{*+} + (1-\lambda) D_{ji}^{*-})$, which is contradicted with the strictly convexity of D_{ji}. So we prove that at least one of $h_{j,i}^{*+}$ and $h_{j,i}^{*-}$ equals to τ.

$h_{j,i}^* = \arg\min_{h_{j,i} \in \{h_{j,i}^{*+}, h_{j,i}^{*-}\}} D_{ji}(h_{j,i})$. If $h_{j,i}^{*+}$ and $h_{j,i}^{*-}$ equals to τ, then $h_{j,i}^* = \tau$ obviously. Otherwise, we assume that $h_{j,i}^{*+} \neq \tau$ without loss of generality, then we have $h_{j,i}^{*-} = \tau$ since one of $h_{j,i}^{*+}$ and $h_{j,i}^{*-}$ must equals to τ. Because $\tau \in \mathcal{S}_+$, so $D_{ji}(h_{j,i}^{*+}) \leq D_{ji}(\tau) = D_{ji}(h_{j,i}^{*-})$, then we have $h_{j,i}^* = h_{j,i}^{*+}$. So the solution $h_{j,i}^*$ is the one of $h_{j,i}^{*+}$ and $h_{j,i}^{*-}$ that not equals to τ. So we have completed the proof of Lemma 2. □

Note that $h_{j,i}^{*+} = \arg\min_{h_{j,i} \in \mathcal{S}_+} D_{ji}(h_{j,i})$ and let $h_{j,i}^{*-} = \arg\min_{h_{j,i} \in \mathcal{S}_-} D_{ji}(h_{j,i})$ are both very simple one-dimensional quadratic optimization problems.

$$\begin{aligned}h_{j,i}^{*+} &= \mathrm{proj}_{\mathcal{S}_+} \left\{ \frac{\lambda_2 X_{li}^{(j)T} \boldsymbol{b}_j + y_i \boldsymbol{w}_j}{\lambda_2 \|\boldsymbol{b}_j\|_2^2 + \lambda_3} \right\} \\ &= \mathrm{proj}_{\mathcal{S}_+} \left\{ \frac{\lambda_2([X_{li}^T B]_j - H_{li}[B^T B]_j + \|\boldsymbol{b}_j\|_2^2 h_{j,i}) + y_i \boldsymbol{w}_j}{\lambda_2 \|\boldsymbol{b}_j\|_2^2 + \lambda_3} \right\}\end{aligned} \tag{IV.10}$$

And

$$\begin{aligned}h_{j,i}^{*-} &= \mathrm{proj}_{\mathcal{S}_-} \left\{ \frac{\lambda_2 X_{li}^{(j)T} \boldsymbol{b}_j}{\lambda_2 \|\boldsymbol{b}_j\|_2^2 + \lambda_3} \right\} \\ &= \mathrm{proj}_{\mathcal{S}_-} \left\{ \frac{\lambda_2([X_{li}^T B]_j - H_{li}^T [B^T B]_j + \|\boldsymbol{b}_j\|_2^2 h_{j,i})}{\lambda_2 \|\boldsymbol{b}_j\|_2^2 + \lambda_3} \right\}\end{aligned} \tag{IV.11}$$

When $\boldsymbol{w}_j \neq 0$

$$\tau = \frac{1 - y_i \boldsymbol{w}^{(j)T} \boldsymbol{h}_i^{(j)}}{y_i \boldsymbol{w}_j},$$

and by Lemma 2, we can easily find $h_{j,i}^*$ after we get $h_{j,i}^{*+}, h_{j,i}^{*-}$ and τ.

When $\boldsymbol{w}_j = 0$, the L1-hinge loss part is ignored, and $h^*_{j,i}$ will simply be $h^{*-}_{j,i}$.

When the non-negativity constraint is considered, we just need to project $h^*_{j,i}$ to $[0,+\infty]$ to get $\{h^*_{j,i}\}_+$

The above update rules can be vectorized and efficiently implemented by vector and matrix operations:

$$\boldsymbol{\tau} = (\boldsymbol{e} - Y \odot (H_l^{(j)T} \boldsymbol{w}^{(j)})) \oslash (Y \boldsymbol{w}_j) \tag{IV.12}$$

where $\boldsymbol{e} = [1,1,...,1]^T_{1\times ntr}$

$$\boldsymbol{h}^{*+}_{l\cdot j} = \text{proj}_{\mathcal{S}_+} \left\{ \frac{\lambda_2([X_l^T B]_j - H_l^T [B^T B]_j + \|\boldsymbol{b}_j\|_2^2 \boldsymbol{h}_{l\cdot j}) + Y \boldsymbol{w}_j}{\lambda_2 \|\boldsymbol{b}_j\|_2^2 + \lambda_3} \right\} \tag{IV.13}$$

$$\boldsymbol{h}^{*-}_{l\cdot j} = \text{proj}_{\mathcal{S}_-} \left\{ \frac{\lambda_2([X_l^T B]_j - H_l^T [B^T B]_j + \|\boldsymbol{b}_j\|_2^2 \boldsymbol{h}_{l\cdot j})}{\lambda_2 \|\boldsymbol{b}_j\|_2^2 + \lambda_3} \right\} \tag{IV.14}$$

And the update rule could be simply expressed as:

$$\boldsymbol{h}^*_{l\cdot j} = \boldsymbol{h}^{*+}_{l\cdot j} + \boldsymbol{h}^{*-}_{l\cdot j} - \boldsymbol{\tau} \tag{IV.15}$$

Under L-2 hinge loss

For each subproblem:

$$\begin{aligned} \min_{h_{j,i}} G_{ji}(h_{j\cdot i}) := &\max(0, 1 - y_i(\boldsymbol{w}^{(j)T} \boldsymbol{h}_i^{(j)} + \boldsymbol{w}_j h_{j,i}))^2 \\ &+ \frac{\lambda_2}{2} \|X_{li}^{(j)} - \boldsymbol{b}_j h_{j,i}\|_2^2 + \frac{\lambda_3}{2} h_{j,i}^2 \end{aligned} \tag{IV.16}$$

The idea here is exactly same with L-1 hinge loss. We can show that Lemma 2 also holds for L-2 hinge loss using the same technique. So we skip the details and just give the key results directly.

$$\boldsymbol{h}^{*+}_{l\cdot j} = \text{proj}_{\mathcal{S}_+} \left\{ \frac{\lambda_2([X_l B]_j - H_l [B^T B]_j + \|\boldsymbol{b}_j\|_2^2 \boldsymbol{h}_{l\cdot j}) + M}{\lambda_2 \|\boldsymbol{b}_j\|_2^2 + \lambda_3 + 2\boldsymbol{w}_j^2} \right\} \tag{IV.17}$$

Where $M = 2(Y\boldsymbol{w}_j)\boldsymbol{e} - 2(Y \odot Y) \odot \boldsymbol{w}_j (H_l^{(j)T} \boldsymbol{w}^{(j)}) = 2(Y\boldsymbol{w}_j)\boldsymbol{e} - \boldsymbol{w}_j (H_l^{(j)T} \boldsymbol{w}^{(j)})$ since $y_i \in \{-1,+1\}$ and $Y \odot Y = [1,1,...,1]^T$.

$h^{*-}_{j,i}$ is exactly the same under L2 or L1-hinge loss.

$$\boldsymbol{h}^{*-}_{l\cdot j} = \text{proj}_{\mathcal{S}_-} \left\{ \frac{\lambda_2([X_l^T B]_j - H_l^T [B^T B]_j + \|\boldsymbol{b}_j\|_2^2 \boldsymbol{h}_{l\cdot j})}{\lambda_2 \|\boldsymbol{b}_j\|_2^2 + \lambda_3} \right\}$$

3) Updating $\boldsymbol{w}$:

The update of $\boldsymbol{w}$ is a standard linear support vector classification problem, we use the state-of-art solver for large-scale linear SVM - **LIBLINEAR** [11].

Since we have reduced the dimensionality, our problem here is a standard $n \gg p$ problem and it is usually more efficient to solve the primal problem of SVM in this situation.

L1-SVM:

In **LIBLINEAR**, L1-SVM can only be solved in dual form by coordinate descent [12]. In addition, we cannot use the previous $\boldsymbol{w}^{t-1}$ as our initial point to get $\boldsymbol{w}^t$ since we don't know how to initialize the dual variable that corresponding to $\boldsymbol{w}^{t-1}$. Moreover, we need to solve the dual precisely (set a very small tolerance) in order to decrease the primal objective value in each iteration.

L2-SVM:

In **LIBLINEAR**, the primal problem of L2-SVM can be efficiently solved by Newton method [13]. It can take $\boldsymbol{w}^{t-1}$ as the initial point and speed up the computation.

Above analysis showed that the primal solver for L2-SVM is more efficient than the dual solver for L1-SVM. Numerical experiments show that L1-SVM and L2-SVM usually have similar performance in real-world datasets. So L2-hinge loss is recommended to be used.

C. Convergence

By Lemma 1, our algorithm is guaranteed to have convergence to stationary point.

D. Initialization

Since our algorithm is only able to find stationary points, different initializations could converge to different points and further affect the classification performance. A good initialization is thus important since it can lead to a better convergent point and meanwhile speeding up the convergence speed.

Here we use the top k SVD to get B_0 and H_0 since it is the exact solution for Low-rank approximation (the right part of Eq. (IV.1)). Let $X \approx U_k \Sigma_k V_k^T$, then our $B_0 = U_k \sqrt{\Sigma_k}$ and $H_0 = \sqrt{\Sigma_k} V_k^T$. Numerical experiments demonstrate that our algorithm could usually lead to faster convergence and better convergent points compared with random initialization. Note that the top k SVD decomposition for large sparse matrix can be computed efficiently by using PROPACK [23], so the time spent on initialization is negligible.

When non-negative constraints are considered, we simply use random initialization.

E. Complexity analysis

Time complexity

Let $S = nnz(X)$. In each iteration, we need to compute HH^T, XH^T, B^TB and X^TB, the complexities are $O(k^2n), O(kS), O(k^2m)$ and $O(kS)$ respectively. The update of B requires $O(k^2m)$ operations and the update of H needs $O(k^2n)$ operations. So the total complexity for each iteration is $O(k^2(n+m) + kS) + \text{O(SVM-Solver)}$. Numerical experiments show that the time spent on updating $\boldsymbol{w}$ (SVM-Solver) is far smaller than the time spent on updating B and H.

Space complexity

The space complexity is straightforward since all matrices we need to store are $X, Y, B, H, \boldsymbol{w}$, so the space complexity is $O(k(m+n) + S)$.

The detailed algorithm can be found in Algorithm 1.

Algorithm 1 DRSVM

Input: $Y_l, X_l, X_u, k, \lambda_1, \lambda_2, \lambda_3, loss, nng, maxiter$
% nng is the short-cut for $non-negative$
$X = [X_l, X_u]$, $[m, n] = size(X)$, $ntr = size(X_l, 2)$
Initialize $B \in \mathbb{R}^{m \times k}$, $H \in \mathbb{R}^{k \times n}$ and $\boldsymbol{w}^{(0)} = zeros(k)$
if nng **then**
  $B = \{B\}_+, H = \{H\}_+$
end if
for $t = 1, 2, \ldots, maxiter$ **do**
  % Update B
  $HHT = HH^T, XHT = XH^T$
  for $i = 1, 2, \ldots, k$ **do**
    $\|\boldsymbol{h}_{\cdot i}\|_2^2 = HHT(i, i)$
    Call update rule IV.6
    if nng **then**
      $\boldsymbol{b}_j = \{\boldsymbol{b}_j\}_+$
    end if
  end for
  $BTB = B^T B, XlTB = X_l^T B, XuTB = X_u^T B$
  % Update H
  Split H into $[H_l, H_u]$
  for $i = 1, 2, \ldots, k$ **do**
    % Update $H_u{\cdot}k$
    Call update rule IV.8
    % Update $H_l{\cdot}k$
    if L1-hinge loss **then**
      Calculate $\boldsymbol{h}_{\cdot i}^{*+}$ by Eq. IV.13
    else if L2-hinge loss **then**
      Calculate $\boldsymbol{h}_{\cdot i}^{*+}$ by Eq. IV.17
    end if
    Calculate $\boldsymbol{\tau}, \boldsymbol{h}_{\cdot i}^{*-}$ by Eq. IV.12 and Eq. IV.14
    Call Update rule IV.15
    if nng **then**
      $\boldsymbol{h}_{\cdot j} = \{\boldsymbol{h}_{\cdot j}\}_+$
    end if
  end for
  % Update $\boldsymbol{w}$
  if L1-hinge loss **then**
    Call **LIBLINEAR** L1-dual Solver
  else if L2-hinge loss **then**
    Call **LIBLINEAR** L2-primal Solver
    with initial point $\boldsymbol{w}^{i-1}$
  end if
  if $stopping\ criterion\ is\ met$ **then**
    break
  end if
end for
Output: $B, H, \boldsymbol{w}$

V. Experimental Results

In this section, we compare DRSVM with other existing semi-supervised and unsupervised dimension reduction methods on classification accuracy. We vary the size of labeled data in the experiments to demonstrate our algorithm outperforms existing methods when observing a small subset of labels. The following algorithms are included in comparisons:

- DRSVM: Our semi-supervised dimension reduction method with both L1 and L2 hinge loss.
- SVD+SVM: Classical unsupervised Low-rank approximation approach—using PCA (or equivalently, SVD on the feature matrix) to conduct dimension reduction, and then run SVM on the low-dimensional features.
- SSDR+kNN [4]: Semi-supervised dimension reduction based on graph constraints. To run this method, we transform the observed labels into must-link and cannot-link constraints between all the labeled instances. For example, when there are p instances with label +1 and q instances with label -1, we generate $p^2 + q^2$ must-links between instances with same labels, and pq cannot link between instances with different labels.
- SSDR+SVM: Similar to SSDR+kNN, but the final classifier is changed to SVM. We use **LIBLINEAR** to train a linear SVM model on low-dimensional features computed by SSDR.

Here we consider the case when the number of labeled data is far less than number of unlabeled data, which is common in many real world applications. The performance of all algorithms are evaluated by the classification accuracy on unlabeled data. All experiments are conducted on a server with 32 Intel Xeon E5-2690 @ 2.90GHz CPUs and enough memory.

We use 5 datasets to conduct our experiments. Among them, three (rcv1, news20, webspam) are document data. All datasets used are available on the webpage of **LIBSVM** [7] https:/www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. Table III presents the statistics of each dataset.

dataset	# features	# samples	# nnz
adult	123	48,842	677,323
gisette	5,000	7,000	34,700,997
webspam	254	350,000	29,796,333
news20	1,355,191	19,996	9,097,916
rcv1	47,236	697,641	51,055,210

TABLE III: Dataset Statistics

A. Implementation

We implement SSDR+kNN, SVD+SVM and DRSVM by MATLAB. Note that SSDR needs to find the top k eigenvectors of a dense $m \times m$ matrix, and the naive implementation is not able to handle datasets with more than 100,000 features. Special tricks are used to solve the scalability issue, and our implemented SSDR is efficient in speed and only require $O(k(m+n)+S)$ space which is the same as DRSVM. These tricks are described in Appendix.

dataset	# labeled data	L1-DRSVM	L2-DRSVM	SVD+SVM	SSDR+kNN	SSDR+SVM
adult	20	**76.39**	76.35	76.29	76.34	66.78
adult	40	**79.7**	78.31	79.51	78.10	75.76
adult	60	78.2	**81.3**	76.71	78.75	78.62
adult	80	**81.5**	80.51	80.03	78.43	77.38
adult	100	**82.07**	80.7	79.61	78.93	77.55
adult	150	**81.86**	81.73	77.22	80.17	78.97
adult	200	**82.17**	81.16	78.04	80.08	77.07
gisette	20	80.63	**82.43**	80.56	52.52	56.7
gisette	40	86.26	**87.5**	85.65	65.54	79.18
gisette	60	88	**88.78**	87.8	70.03	79.3
gisette	80	88.66	**89.49**	88.63	73.6	77.5
gisette	100	**89.26**	89.1	89.15	79.93	84.5
gisette	150	89.52	**90.03**	89.49	84.06	84.98
gisette	200	89.68	**90.43**	89.79	84.28	82.6
webspam	20	60.37	**62.05**	60.43	60.92	60.41
webspam	40	60.91	**76.28**	60.68	66.87	60.68
webspam	60	66.5	**73.69**	61.01	72.06	61.6
webspam	80	74.43	74.35	60.92	**77.82**	63.3
webspam	100	75	74.53	61.5	**80.43**	65.4
webspam	150	78.65	76.18	61.13	**80.84**	67.4
webspam	200	78.08	76.57	61.25	**82.49**	70.5
news20	20	**71.06**	70.52	59.99	49.98	68
news20	40	75.23	76.69	**76.89**	50.04	52.5
news20	60	**77.27**	74.65	75.91	50.01	56
news20	80	75.99	74.11	**76.79**	50.11	51.14
news20	100	77.3	76.03	**77.96**	50.24	51.56
news20	150	**76.14**	76.1	73.29	49.99	59.15
news20	200	73.12	**75.65**	72.99	50.04	60.47
rcv1	20	72.75	**73.36**	72.22	47.54	60.18
rcv1	40	78.34	**79.04**	76.98	47.55	59.26
rcv1	60	85.44	85.28	**85.51**	48.87	64.7
rcv1	80	87.41	87.03	**87.67**	59.04	70.19
rcv1	100	88.34	87.04	**88.46**	56.2	70.74
rcv1	150	**89.55**	89.13	88.9	63.54	69.49
rcv1	200	89.5	**89.61**	88.5	73.03	65

TABLE II: Prediction accuracy on unlabeled data.

B. Prediction for unlabeled data

Experiments setting

In the experiments, we consider the scenario when the number of labeled data is far less than the number of unlabeled data. For each data set, we try 20, 40, 60, 80, 100, 150 and 200 as the number of labeled data. For SSDR, we set $\alpha = 1$ and $\beta = 20$ which is the same parameter setting used in [4]. As for the kNN prediction, we set the number of nearest neighbours to be 5. For the SVM solver in SVD+SVM and SSDR+SVM, we set $C = 1$. For our model, we simply set $\lambda_1 = \lambda_2 = \lambda_3 = 1$. In subsection V-C, we will empirically show that our algorithm is robust to different parameter settings.

Prediction Accuracy

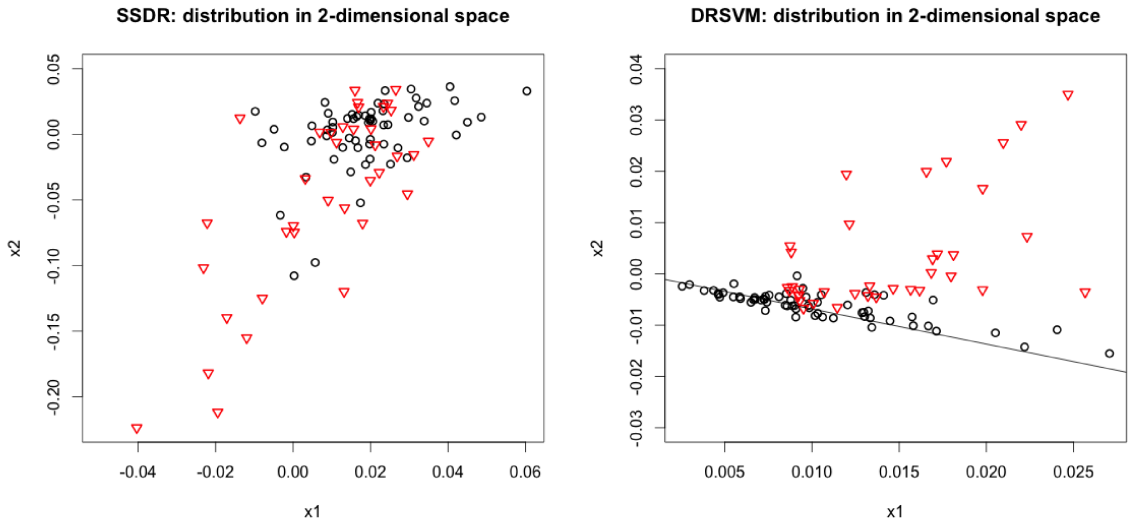
Table II presents the detailed results of prediction accuracy. DRSVM outperforms the other two methods in most cases, and L1-DRSVM and L2-DRSVM usually yield very similar accuracy. With some further investigation, we find that there are some cases where the performance of DRSVM and SVD+SVM are similar. This is mainly due to our particular initialization scheme: we use the solution of SVD+SVM to initialize our algorithm for training DRSVM, so their performances tend to be similar if there are stationary points close to the initialization.

However, in Table II, we observe SSDR+kNN outperforms our methods on webspam dataset with more than 80 observed samples. The main reason is that webspam dataset only has 254 features, so a linear hyperplane (used in SVD+SVM, SSDR+SVM, DRSVM) does not have enough capability to separate positive/negative data. In contrast, kNN is a nonlinear model and works very well on low-dimensional data. Despite this deficiency of linear SVM, our models are still better when the number of labeled data less than 80, which indicates that the proposed approach can better utilize the label information.

C. Sensitivity Analysis

Here we analyze the robustness of our algorithm. Many studies [7], [11] have shown that the parameter C (which is λ_1 in our model) in SVM is quite robust under both L-1 hinge loss and L-2 hinge loss, so here we only study the robustness of λ_2 and λ_3 due to limited space.

We try different values of λ_2 and λ_3 on news20 and rcv1 when the number of labeled data is 100. From Table IV, we can see that the test accuracy on unlabeled data is quite robust to different values of λ_2 and λ_3 in most cases. Therefore, we are able to set $\lambda_1 = \lambda_2 = \lambda_3 = 1$ to achieve good prediction accuracy on all 5 datasets.



(a) SSDR (b) DRSVM

Fig. 3: Distribution of samples in 2-dimensional space for RCV1 dataset. We can observe a clear difference between linkage-based dimension reduction (left figure) and hyperplane-based dimension reduction (right figure).

λ_1	λ_2	λ_3	Test accuracy(news20)	Test accuracy(rcv1)
1	0.1	0.1	71.34%	85.84%
1	1	0.1	76.38%	88.08%
1	10	0.1	77.01%	88.56%
1	0.1	1	70.25%	83.56%
1	1	1	76.03%	87.04%
1	10	1	76.98%	88.58%
1	0.1	10	78.71%	71.12%
1	1	10	73.75%	87.10%
1	10	10	77.10%	88.58%

TABLE IV: Algorithm Robustness

D. Geometric Interpretation

Here we try to plot the low-dimensional embeddings of DRSVM and SSDR and comapre their geom1etric patterns.

Using *rcv*1 and 100 available labels, we set the reduced feature dimension to $k = 2$ and keep all other parameters the same as previous experiments, and get the 2-dimensional embeddings of unlabeled data by DRSVM and SSDR. We randomly sample 100 unlabeled samples and plot their 2-dimensional embeddings under DRSVM and SSDR. The results are shown in Figure 3.

From Figure 3, we can see a clear difference between SSDR and DRSVM. Since SSDR tries to model the labeled information as a graph and force samples belong to the same class close to each other in the low-dimensional embedding, we can observe a clustering structure in SSDR embedding. In contrast, our method DRSVM aims to find a low-dimensional embedding that makes data linearly separable. This observation supports our analysis in section I. We hope this clear geometric interpretation can help users to choose the better low-dimensional embedding for their applications.

E. Runtime Details

dataset	Update B	Update H	Update L2$\boldsymbol{w}$	Update L1 $\boldsymbol{w}$
gisette	0.2725	0.3731	6.73e-04	1e-03
webspam	0.2135	0.5679	2.36e-04	2.62e-04
news20	0.4409	0.2671	2.34e-04	3.32e-04
rcv1	0.7049	1.4418	2.68e-04	2.93e-04

TABLE V: Runtime(sec) for each part of DRSVM.

Table V presents the average run time of DRSVM (k=10) to update B, H and $\boldsymbol{w}$ in each iteration. The update of B and H dominate the run time. The time cost for updating $\boldsymbol{w}$ with L2-hinge loss is slightly faster than using L1-loss, but the difference is negligible compared with the time spent on updating B and H.

VI. Conclusions

In this paper, we propose a novel hyperplane-based semi-suprevised dimension reduction approach, and develop an efficient algorithm for solving it. The resulting algorithm, DRSVM, is able to learn the low-dimensional embedding and separating hyperplane simultaneously. Experimental results on some real world high dimensional datasets show that DRSVM is able to classify unlabeled data more accurately than the state-of-art linkage-based SSDR when very few labeled data is available.

Currently we only consider binary classification problems in our experiments. In section IV, we proposed the formulation of multi-class DRSVM using One-versus-Rest strategy. Developing an efficient optimization algorithm for multi-class DRSVM is

sophisticated but doable, we left this as our future work.

Acknowledgement. This research was supported by NSF grant IIS-1719097. We also acknowledge the resource allocation from XSEDE.

References

[1] I. Joliffe, *Principal Component Analysis*, Springer, New York, NY, 1986.

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

[3] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific Belmont, MA, 1999.

[4] D. Zhang, Z. Zhou and S. Chen *Semi-supervised dimensional reduction*, Proceedings of the 7th SIAM International Conference on Data Mining(SDM'07), pp. 629–634, 2007

[5] A. Cichocki, and P. H. A. N. Anh-Huy. *Fast local algorithms for large scale nonnegative matrix and tensor factorizations*, IEICE transactions on fundamentals of electronics, communications and computer sciences 92.3: 708–721, 2009.

[6] H. Lee, J. Yoo, and S. Choi, *Semi-supervised nonnegative matrix factorization*, Signal Processing Letters, IEEE, Vol.17, No.1, pp. 4–7, 2010

[7] C.-C. Chang and C.-J. Lin, *LIBSVM : a library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011.

[8] C.-J. Hsieh and I.S. Dhillon. *Fast coordinate descent methods with variable selection for nonnegative matrix factorization*. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1064–1072. ACM, 2011.

[9] K.-Y. Chiang, C.-J. Hsieh and I.S. Dhillon. *Robust principal component analysis with side information*. In ICML, 2016.

[10] K.-Y. Chiang, C.-J. Hsieh and I.S. Dhillon. *Matrix completion with noisy side information*. In NIPS, 2015.

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. *LIBLINEAR: A Library for Large Linear Classification*, Journal of Machine Learning Research, pp. 1871-1874, 2008

[12] C.-J. Hsieh, KW Chang, C.-J. Lin, SS. Keerthi, S. Sundararajan, *A dual coordinate descent method for large-scale linear SVM*, Proceedings of the 25th international conference on Machine learning, pp. 408–415, 2008

[13] C.-J. Lin, R. C. Weng, and S. S. Keerthi, *Trust region Newton method for large-scale logistic regression*, Journal of Machine Learning Research, pp. 627–650, 2008.

[14] D. D. Lee and H. S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401(6755): pp. 788–791, 1999.

[15] D. D. Lee and H. S. Seung, *Algorithms for non-negative matrix factorization*, In Advances in Neural Information Processing, pp. 556–562, 2001.

[16] J. Choo, C. Lee, C. K. Reddy, and H. Park. *Weakly supervised nonnegative matrix factorization for user-driven clustering*, Data Mining and Knowledge Discovery (DMKD), 29(6),1598–1621, 2015

[17] W. Xu, X. Liu, Y. Gong *Document clustering based on non-negative matrix factorization*, Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pp 267–273, Toronto, Canada July 28 – August 01, 2003

[18] F. Shahnaz, M. W. Berry, V. P. Pauca, R. J. Plemmons, *Document clustering using nonnegative matrix factorization*, Information Processing and Management: an International Journal archive, Volume 42 Issue 2, pp 373–386, March 2006

[19] Z. Yang, W. Cohen, and R. Salakhutdinov. *Revisiting semi-supervised learning with graph embeddings*, In International Conference on Machine Learning (ICML), 2016.

[20] Wagstaff, Kiri, et al, *Constrained k-means clustering with background knowledge.*, In International Conference on Machine Learning (ICML). Vol. 1, 2001.

[21] Davidson. I, *Knowledge Driven Dimension Reduction for Clustering*, In International Joint Conference of Artificial Intelligence, pp. 1034–1039, 2009.

[22] J. Wang, C.-I Chang, *Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis*, IEEE Transactions on Geoscience and Remote Sensing (Volume: 44), pp. 1586–1600, June 2006

[23] Rasmus Munk Larsen, *Lanczos Bidiagonalization With Partial Reorthogonalization*, 1998.

[24] Lewis, D. D.; Yang, Y.; Rose, T.; and Li, F. *RCV1: A New Benchmark Collection for Text Categorization Research. Journal of Machine Learning Research*, 5:361–397, 2004.

Appendix

Implementation tricks for SSDR

We consider the case when our feature matrix $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n] \in \mathbb{R}^{m\times n}$ is a large sparse matrix.

The Linkage matrix for SSDR [4] is defined by

$$S(i,j) = \begin{cases} \frac{1}{n^2} + \frac{\alpha}{n_C} & \text{if } (\boldsymbol{x}_i, \boldsymbol{x}_j) \in C \\ \frac{1}{n^2} - \frac{\beta}{n_M} & \text{if } (\boldsymbol{x}_i, \boldsymbol{x}_j) \in M \\ \frac{1}{n^2} & \text{otherwise,} \end{cases}$$

Where M is the set of pairs that are labeled as "must-link", C is the set of pairs that are labeled as "cannot-link", n_M and n_C are the total number of pairs in M and C respectively.

D is a diagonal matrix whose diagonal elements are the column sum of S. In SSDR, we need to solve the top k eigen-decomposition of $X(D-S)X^T$, which is a $m \times m$ dense matrix.

However, by using Lanczos algorithm with partial reorthogonalization [23], we don't need to compute matrix $X(D-S)X^T$ explicitly. We just need to specify how to efficiently compute $X(D-S)X^T z$ for a given $z \in \mathbb{R}^m$.

Given $z \in \mathbb{R}^m, X(D-S)X^T z = X(D(X^T z)) - X(S(X^T z))$.

The first part of the right side – $X(D(X^T z))$ only needs $O(nnz(X)+n)$ to compute.

For the second part, it is easy to get $X^T z$. Let $y = X^T z$. Since the linkage matrix is dense, so the most difficult part now is how to compute Sy efficiently

Let $L_1 = \{i : y_i = +1\}$, $L_2 = \{i : y_i = -1\}$ and $L_u = \{i : \boldsymbol{x}_i \text{ is unlabeled}\}$. Then S can be expressed as the product of 2 low-dimensional matrices $U \in \mathbb{R}^{n\times 3}$, $V \in \mathbb{R}^{3\times n}$, where

$$\begin{cases} i \in L_1 : U_{i,1} = \frac{1}{n^2} - \frac{\beta}{n_M}, U_{i,2} = \frac{1}{n^2} + \frac{\alpha}{n_C}, U_{i,3} = \frac{1}{n^2} \\ i \in L_2 : U_{i,1} = \frac{1}{n^2} + \frac{\alpha}{n_C}, U_{i,2} = \frac{1}{n^2} - \frac{\beta}{n_M}, U_{i,3} = \frac{1}{n^2} \\ i \in L_u : U_{i,j} = \frac{1}{n^2} \quad \text{for } \forall j = 1,2,3 \end{cases}$$

and

$$\begin{cases} i \in L_1 : V_{1,i} = 1, V_{2,i} = 0, V_{3,i} = 0 \\ i \in L_2 : V_{1,i} = 0, V_{2,i} = 1, V_{3,i} = 0 \\ i \in L_u : V_{1,i} = 0, V_{2,i} = 0, V_{3,i} = 1 \end{cases}$$

Therefore, we can write Sy as $U(Vy)$ and the time complexity to compute Sy is only $O(n)$ now.

To sum up, the total time complexity to compute $X(D-S)X^T z$ is $O(nnz(X)+n)$, and the total space complexity is $O(k(m+n)+nnz(X))$ since all matrices we need to store are X, D, U, V.