A Unified Semi-Supervised Dimensionality Reduction Framework for Manifold Learning

Ratthachat Chatpatanasiri and Boonserm Kijsirikul

Department of Computer Engineering, Chulalongkorn University,
Bangkok 10330, THAILAND.
ratthachat.c@gmail.com and boonserm.k@chula.ac.th

May 10, 2009

Abstract

We present a general framework of semi-supervised dimensionality reduction for manifold learning which naturally generalizes existing supervised and unsupervised learning frameworks which apply the spectral decomposition. Algorithms derived under our framework are able to employ both labeled and unlabeled examples and are able to handle complex problems where data form separate clusters of manifolds. Our framework offers simple views, explains relationships among existing frameworks and provides further extensions which can improve existing algorithms. Furthermore, a new semi-supervised kernelization framework called "KPCA trick" is proposed to handle non-linear problems.

Keywords: Semi-supervised Learning, Transductive Learning, Spectral Methods, Dimensionality Reduction, Manifold Learning, KPCA Trick.

1 Introduction

In many real-world applications, high-dimensional data indeed lie on (or near) a low-dimensional subspace. The goal of *dimensionality reduction* is to reduce complexity of input data while some desired intrinsic information of the data is preserved. The desired information can be discriminative [1, 2, 3, 4, 5, 6], geometrical [7, 8, 9, 10] or both [11]. Fisher discriminant analysis (FDA) [12] is the most popular method among all supervised dimensionality reduction algorithms. Denote c as the number of classes in a given training set. Provided that training examples of each class lie in a linear subspace and *do not* form several separate clusters, i.e. *do not* form multi-modality, FDA is able to discover a low-dimensional linear subspace (with at most $c - 1$ dimensionality) which is efficient for classification. Recently, many works have improved the FDA algorithm in several aspects [11, 1, 2, 3, 4, 5, 6]. These *extended FDA algorithms* are able to discover a nice low-dimensional subspace even when training examples of each class lie in separate clusters of complicated non-linear manifolds. Moreover, a subspace discovered by these algorithms has no limitation of $c - 1$ dimensionality.

Although the extended FDA algorithms work reasonably well, a considerable number of labeled examples is required to achieve satisfiable performance. In many real-world applications such as image classification, web page classification and protein function prediction, a labeling process is costly and time consuming; in contrast, unlabeled examples can be easily obtained. Therefore, in such situations, it can be beneficial to incorporate the information which is contained

in unlabeled examples into a learning problem, i.e., semi-supervised learning (SSL) should be applied instead of supervised learning [13].

In this paper, we present a general semi-supervised dimensionality reduction framework which is able to employ information from both labeled and unlabeled examples. Contributions of the paper can be summarized as follows.

- As the extended FDA algorithms, algorithms developed in our framework are able to discover a nice low-dimensional subspace even when training examples of each class form separate clusters of complicated non-linear manifolds. In fact, those previous supervised algorithms can be casted as instances in our framework. Moreover, our framework explains previously unclear relationships among existing algorithms in a simple viewpoint.

- We present a novel technique called the *Hadamard power operator* which improves the use of unlabeled examples in previous algorithms. Experiments show that the Hadamard power operator improves the classification performance of a semi-supervised learner derived from our framework.

- We show that recent existing semi-supervised frameworks applying spectral decompositions known to us [14, 15] can be viewed as special cases of our framework. Moreover, empirical evidence shows that semi-supervised learners derived from our framework are superior to existing learners in many standard problems.

- A new non-linearization framework, namely, a *KPCA trick* framework [16] is extended into a semi-supervised learning setting. In contrast to the standard kernel trick, the KPCA trick does not require users to derive new mathematical formulas and to re-implement the kernel version of the original learner.

2 The Framework

Let $\{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$ denote a training set of ℓ labeled examples, with inputs $\mathbf{x}_i \in \mathbb{R}^{d_0}$ generated from a fixed but unknown probability distribution $\mathcal{P}_{\mathbf{x}}$, and corresponding class labels $y_i \in \{1, ..., c\}$ generated from $\mathcal{P}_{y|\mathbf{x}}$. In addition to the labeled examples, let $\{\mathbf{x}_i\}_{i=\ell+1}^{\ell+u}$ denote a set of u unlabeled examples also generated from $\mathcal{P}_{\mathbf{x}}$. Denote $X \in \mathbb{R}^{d_0 \times (\ell+u)}$ as a matrix of the input examples $(\mathbf{x}_1, ..., \mathbf{x}_{\ell+u})$. We define $n = \ell + u$. The goal of semi-supervised learning (SSL) dimensionality reduction is

Goal. Using the information of both labeled and unlabeled examples, we want to map $(\mathbf{x} \in \mathbb{R}^{d_0}) \mapsto (\mathbf{z} \in \mathbb{R}^d)$ where $d < d_0$, such that in the embedded space $\mathcal{P}_{y|\mathbf{z}}$ can be accurately estimated (i.e., unknown labels are easy to predict) by *a simple classifier*.

Here, following the previous works in the supervised setting [11, 1, 2], the *nearest neighbor* algorithm is used for representing a simple classifier mentioned in the goal. Note that important special cases of SSL problems are *transductive* problems where we only want to predict the labels $\{y_i\}_{i=\ell+1}^{\ell+u}$ of the given unlabeled examples. In order to make use of unlabeled examples in the learning process, we make the following so-called *manifold assumption* [13]:

Semi-Supervised Manifold Assumption. The support of $\mathcal{P}_{\mathbf{x}}$ is on a low-dimensional manifold. Furthermore, $\mathcal{P}_{y|\mathbf{x}}$ is smooth, as a function of $\mathbf{x}$, with

respect to the underlying structure of the manifold.

At first, to fulfill our goal, we linearly parameterize $\mathbf{z}_i = A\mathbf{x}_i$ where $A \in \mathbb{R}^{d \times d_0}$. Thus, $AX = (A\mathbf{x}_1, ..., A\mathbf{x}_n) \in \mathbb{R}^{d \times n}$ is a matrix of embedded points. An efficient non-linear extension is presented in Section 2.2. In our framework, we propose to cast the problem as a constrained optimization problem:

$$A^* = \arg\min_{A \in \mathcal{A}} f^\ell(AX) + \gamma f^u(AX), \tag{1}$$

where $f^\ell(\cdot)$ and $f^u(\cdot)$ are objective functions based on labeled and unlabeled examples, respectively, γ is a parameter controlling the weights between the two objective functions and $\mathcal{A}$ is a constraint set in $\mathbb{R}^{d \times d_0}$. The two objective functions determine "how good the embedded points are"; thus, their arguments are AX, a matrix of embedded points. Up to *orthogonal and translational transformations*, we can identify embedded points via their pairwise distances instead of their individual locations. Therefore, we can base the objective functions on pairwise distances of embedded examples. Here, we define the objective functions to be *linear* with respect to the pairwise distances:

$$f^\ell(AX) = \sum_{i,j=1}^n c_{ij}^\ell \, \text{dist}(A\mathbf{x}_i, A\mathbf{x}_j) \text{ and } f^u(AX) = \sum_{i,j=1}^n c_{ij}^u \, \text{dist}(A\mathbf{x}_i, A\mathbf{x}_j),$$

where $\text{dist}(\cdot, \cdot)$ is an arbitrary distance function between two embedded points, c_{ij}^ℓ and c_{ij}^u are costs which penalize an embedded distance between two points i and j. A specification of c_{ij}^ℓ and c_{ij}^u are based on the label information and unlabel information, respectively, as described in Section 2.1.

If we restrict ourselves to consider only the cases that (I) $\text{dist}(\cdot, \cdot)$ is a squared Euclidean distance function, i.e. $\text{dist}(A\mathbf{x}_i, A\mathbf{x}_j) = \|A\mathbf{x}_i - A\mathbf{x}_j\|^2$, (II) c_{ij}^ℓ and c_{ij}^u are symmetric, and (III) $A \in \mathcal{A}$ is in the form of $ABA^T = I$ where B is a positive semidefinite (PSD) matrix, Eq.(1) will result in a general framework which indeed generalizes previous frameworks as shown in Section 3. Define $c_{ij} = c_{ij}^\ell + \gamma c_{ij}^u$. We can rewrite the weighted combination of the objective funtions in Eq.(1) as follows:

$$
\begin{aligned}
&f^\ell(AX) + \gamma f^u(AX) \\
&= \sum_{i,j=1}^n c_{ij}^\ell \text{dist}(A\mathbf{x}_i, A\mathbf{x}_j) + \gamma \sum_{i,j=1}^n c_{ij}^u \, \text{dist}(A\mathbf{x}_i, A\mathbf{x}_j) \\
&= \sum_{i,j=1}^n (c_{ij}^\ell + \gamma c_{ij}^u) \text{dist}(A\mathbf{x}_i, A\mathbf{x}_j) = \sum_{i,j=1}^n c_{ij} \text{dist}(A\mathbf{x}_i, A\mathbf{x}_j) \\
&= \sum_{i,j=1}^n c_{ij} \|A\mathbf{x}_i - A\mathbf{x}_j\|^2 = 2 \sum_{i,j=1}^n c_{ij} (\mathbf{x}_i^T A^T A\mathbf{x}_i - \mathbf{x}_i^T A^T A\mathbf{x}_j) \\
&= 2\text{trace}\left(A\Big(\sum_{i,j=1}^n (\mathbf{x}_i c_{ij} \mathbf{x}_i^T) - \sum_{i,j=1}^n (\mathbf{x}_i c_{ij} \mathbf{x}_j^T) \Big) A^T \right) \\
&= 2\text{trace}(AX(D - C)X^T A^T),
\end{aligned}
$$

where C is a symmetric cost matrix with elements c_{ij} and D is a diagonal matrix with $D_{ii} = \sum_j c_{ij}$[1]. Thus, the optimization problem (1) can be restated as

$$A^* = \underset{ABA^T=I}{\arg\min} \operatorname{trace}(AX(D - C)X^T A^T), \tag{2}$$

Note that the constraint $ABA^T = I$ prevents trivial solutions such as every $A\mathbf{x}_i$ is a zero vector. If B is a positive definite (PD) matrix, a solution of the above problem is given by the bottom d eigenvectors of the following generalized eigenvalue problem [12, 17]

$$X(D - C)X^T \mathbf{a}^{(j)} = \lambda_j B \mathbf{a}^{(j)}, \quad j = 1, ..., d. \tag{3}$$

Then the optimal linear map is

$$A^* = (\mathbf{a}^{(1)}, ..., \mathbf{a}^{(d)})^T. \tag{4}$$

Note that, in terms of solutions of Eq.(3), it is more convenient to represent A^* by its rows $\mathbf{a}^{(i)}$ than its columns $\mathbf{a}_i$. Moreover, note that

$$\|\mathbf{z} - \mathbf{z}'\| = \|A^*\mathbf{x} - A^*\mathbf{x}'\|. \tag{5}$$

Therefore, kNN in the embedded space can be performed. Consequently, an algorithm implemented under our framework consists of three steps as shown in Figure 1.

Input: 1. training examples: $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_\ell, y_\ell), \mathbf{x}_{\ell+1}, ..., \mathbf{x}_{\ell+u}\}$
        2. a new example: $\mathbf{x}'$
        3. a positive-value parameter: γ
Algorithm:
(1) Construct cost matrices, C^ℓ, C^u and $C = C^\ell + \gamma C^u$,
    and a constraint matrix B (see Section 2.1).
(2) Obtain an optimal matrix A^* by solving Eq.(3).
(3) Perform kNN classification in the obtained subspace by using Eq.(5).

Figure 1: Our semi-supervised learning framework.

2.1 Specification of the Cost and Constraint Matrices

In this section, we present various reasonable approaches of specifying the two cost matrices, C^ℓ and C^u, and the constraint matrix, B, by using the label and unlabel information. We use the two words "unlabel information" and "neighborhood information" interchangeably in this paper.

2.1.1 The Cost Matrix C^ℓ and the Constraint Matrix B

Normally, based on the label information, classical supervised algorithms usually require an embedded space to have the following two desirable conditions:

---

[1]To simplify our notations, in this paper whenever we define a cost matrix C' having elements c'_{ij}, we always define its associated diagonal matrix D' with elements $D'_{ii} = \sum_j c'_{ij}$.
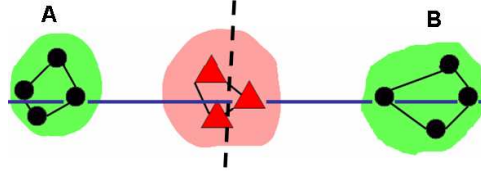
Figure 2: An example when data form a multi-modal structure. An algorithm, e.g. FDA, which imposes the condition (1) will try to discover a new subspace (a dashed line) which merges two clusters **A** and **B** altogether. An obtained space is undesirable as data of the two classes are mixed together. In contrast, an algorithm which imposes the condition (1*) (instead of (1)) will discover a subspace (a thick line) which does not merge the two clusters **A** and **B** as there are no nearby examples (indicated by a link between a pair of examples) between the two clusters.

(1) two examples of the same class stay close to one another, and
(2) two examples of different classes stay far apart.

The two conditions are imposed in classical works such as FDA. However, the first condition is too restrictive to capture manifold and multi-modal structures of data which naturally arise in some applications. Thus, the first condition should be relaxed as follows:

(1*) two *nearby examples* of the same class stay close to one another

where "nearby examples", defined by using the neighborhood information, are examples which should stay close to each other in both original and embedded spaces. The specification of "nearby examples" has been proven to be successful in discovering manifold and multi-modal structure [11, 1, 2, 3, 4, 5, 6, 18, 19, 20, 21, 22]. See Figure 2 for explanations. In some cases, it is also appropriate to relax the second condition to

(2*) two *nearby examples* of different classes stay far apart.

In this section, we give three examples of cost matrices which satisfy the conditions (1*) and (2) (or (2*)). These three examples are recently introduced in previous works, namely, Discriminant Neighborhood Embedding (DNE) [2], Marginal Fisher Analysis (MFA) [1] and Local Fisher Discriminant Analysis (LFDA) [11], with different presentations and motivations but they can be unified under our general framework.

Firstly, to specify nearby examples, we construct two matrices C^I and C^E based on any sensible distance (Euclidean distance is the simplest choice). For each $\mathbf{x}_i$, let $Neig^I(i)$ be the set of k nearest neighbors having the *same* label y_i, and let $Neig^E(i)$ be the set of k nearest neighbors having *different* labels from y_i. Define C^I and C^E as follows: let $c_{ij}^I = c_{ij}^E = 0$ if points $\mathbf{x}_i$ and/or $\mathbf{x}_j$ are unlabeled, and

$$c_{ij}^I = \begin{cases} 1, & \text{if } j \in Neig^I(i) \vee i \in Neig^I(j), \\ 0, & \text{otherwise, and} \end{cases}$$

$$c_{ij}^E = \begin{cases} 1, & \text{if } j \in Neig^E(i) \vee i \in Neig^E(j), \\ 0, & \text{otherwise.} \end{cases}$$

The specification $c_{ij}^I = 1$ and $c_{ij}^E = 1$ represent nearby examples in the conditions (1*) and (2*). Then, C^ℓ and B of existing algorithms (see Eq. (2)) are:

Discriminant Neighborhood Embedding (DNE)
$C^\ell = C^I - C^E \qquad B = I$ (an identity matrix)

Marginal Fisher Analysis (MFA)
$C^\ell = -C^E \qquad\qquad B = X(D^I - C^I)X^T$

Local Fisher Discriminant Analysis (LFDA)
Let $n_1, ..., n_c$ be the numbers of examples of classes $1, ..., c$, respectively. Define matrices C^{bet} and C^{wit} as:

$$c_{ij}^{bet} = \begin{cases} c_{ij}^I(\frac{1}{n_k} - \frac{1}{n}), & \text{if } y_i = y_j = k, \\ -\frac{1}{n}, & \text{otherwise,} \end{cases} \quad \text{and} \quad c_{ij}^{wit} = \begin{cases} \frac{1}{n_k}c_{ij}^I, & \text{if } y_i = y_j = k, \\ 0, & \text{otherwise,} \end{cases}$$

$$C^\ell = C^{bet} \qquad\qquad B = X(D^{wit} - C^{wit})X^T$$

Within our framework, relationships among the three previous works can be explained. The three methods exploit different ideas in specifying matrices C^ℓ and B to satisfy two desirable conditions in an embedded space. In DNE, C^ℓ is designed to penalize an embedded space which does not satisfy the condition (1*) and (2*). In MFA, the constraint matrix B is designed to satisfy the condition (1*) and C^ℓ is designed to penalize an embedded space which does not satisfy the condition (2*).

Things are not quite obvious in the case of LFDA. In LFDA, the constraint matrix B is designed to satisfy the condition (1*) since elements C^{wit} are proportional to C^I; nevertheless, since weights are inversely proportional to n_k, elements in a small class have larger weights than elements in a bigger class, i.e. a pair in a small class is more likely to satisfy the condition (1*) than a pair in a bigger class. To understand C^ℓ, we recall that

$$\text{trace}(AX(D^\ell - C^\ell)X^T A^T) = \sum_{i,j} c_{ij}^\ell \|A\mathbf{x}_i - A\mathbf{x}_j\|$$
$$= \sum_{y_i = y_j} c_{ij}^I(\frac{1}{n_k} - \frac{1}{n})\|A\mathbf{x}_i - A\mathbf{x}_j\| - \sum_{y_i \neq y_j} \frac{1}{n}\|A\mathbf{x}_i - A\mathbf{x}_j\|$$
$$= d - \frac{1}{n}\left(\sum_{y_i = y_j} c_{ij}^I \|A\mathbf{x}_i - A\mathbf{x}_j\| + \sum_{y_i \neq y_j} \|A\mathbf{x}_i - A\mathbf{x}_j\|\right),$$

where at the third equality we use the constraint $AXBX^T A^T = I$ and hence

$$\text{trace}(AX(D^{wit} - C^{wit})X^T A^T) = \sum_{y_i = y_j} \frac{c_{ij}^I}{n_k}\|A\mathbf{x}_i - A\mathbf{x}_j\|$$
$$= \text{trace}(I) = d.$$

Hence, we observe that *every* pair of labeled examples coming from *different* classes has a corresponding cost of $-\frac{1}{n}$. Therefore, C^ℓ is designed to penalize an embedded space which does not satisfy the condition (2). Surprisingly, in

LFDA, nearby examples of the *same* class (having $c_{ij}^I = 1$) also has a cost of $-\frac{1}{n}$. As a cost proportional to $-\frac{1}{n}$ is meant to preserve a pairwise distance between each pair of examples (see Section 3.1), LFDA tries to preserve a local geometrical structure between each pair of nearby examples of the same class, in contrast to DNE and MFA which try to squeeze nearby examples of the same class into a single point

We note that other recent supervised methods for manifold learning can also be presented and interpreted in our framework with different specifications of C^ℓ, for examples, *Local Discriminant Embedding* of Chen et al. [5] and *Supervised Nonlinear Local Embedding* of Cheng et al [6].

2.1.2 The Cost Matrix C^u and the Hadamard Power Operator

One important implication of the manifold assumption is that "nearby examples are likely to belong to a same class". Hence, by the assumption, it makes sense to design C^u such that *it prevents any pairs of nearby examples to stay far apart in an embedded space.*

Among methods of extracting the neighborhood information to define C^u, methods based on *the heat kernel* (or *the gaussian function*) are most popular. Beside using the heat kernel, other methods of defining C^u are invented, see [13, Chap. 15] and [17] for more details. The simplest specifications of nearby examples based on the heat kernel are:

$$c_{ij}^u = \exp(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}). \tag{6}$$

Each pair of nearby examples will be penalized with different costs depended on their similarity, and a similarity between two points is based on the Euclidean distance between them in the input space. Incidentally, with this specification of C^u, the term $f^u(AX)$ in Eq. (1) can be interpreted as an approximation of the *Laplace-Beltrami operator* on a data manifold. A learner which employs $C = C^u$ ($C^\ell = 0$) is named *Locally Preserving Projection* (LPP) [9].

The parameter σ is crucial as it controls the scale of a cost c_{ij}^u. Hence, the choice of σ must be sensible. Moreover, an appropriate choice of σ may vary across the support of $\mathcal{P}_\mathbf{x}$. Hence, the *local scale* σ_i for each point $\mathbf{x}_i$ should be used. Let $\mathbf{x}_i'$ be the kth nearest neighbor of $\mathbf{x}_i$. A local scale is defined as

$$\sigma_i = \|\mathbf{x}_i' - \mathbf{x}_i\|,$$

and a weight of each edge is then defined as

$$c_{ij}^u = \exp(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_i \sigma_j}). \tag{7}$$

Using this local scaling method is proven to be efficient in previous experiments [23] on clustering. A specification of k to define the local scale of each point is usually more convenient than a specification of σ since a space of possible choices of k is considerably smaller than that of σ.

Instead of proposing yet another method to specify a cost matrix, here we present a novel method which can be used to modify any existing cost matrix. Let Q and R be two matrices of equal size and have q_{ij} and r_{ij} as their elements. Recall that the *Hadamard product* P [24] between Q and R, $P = Q \odot R$, has

elements $p_{ij} = q_{ij}r_{ij}$. In words, the Hadamard product is a pointwise product between two matrices. Here, we define the Hadamard α^{th} power operator as

$$\overset{\alpha}{\bigodot} Q = \overbrace{Q \odot Q \odot ... \odot Q}^{\alpha\ times}. \tag{8}$$

Given a cost matrix C^u and a positive integer α, we define a new cost matrix C^{u^α} as

$$C^{u^\alpha} = \overset{\alpha}{\bigodot} C^u \frac{\|C^u\|_F}{\|\bigodot^\alpha C^u\|_F}, \tag{9}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. The multiplication of $\frac{\|C^u\|_F}{\|\bigodot^\alpha C^u\|_F}$ make $\|C^{u^\alpha}\|_F = \|C^u\|_F$. Note that if C^u is symmetric and non-negative, C^{u^α} still has these properties.

The intuition of C^{u^α} will be explained through experiments in Section 4 where we show that C^{u^α} can further improve the quality of C^u so that the classification performance of a semi-supervised learner is increased.

Any combinations of a label cost matrix C^ℓ of those in previous works such as DNE, MFA and LFDA with an unlabel cost matrix C^u result in new SSL algorithms, and we will call the new algorithms SS-DNE, SS-MFA and SS-LFDA.

2.2 Non-Linear Parameterization Using the KPCA Trick

By the linear parameterization, however, we can only obtain a linear subspace defined by A. Learning a non-linear subspace can be accomplished by the standard *kernel trick* [25]. However, applying the kernel trick can be inconvenient since new mathematical formulas have to be derived and new implementation have to be done separately from the linear implementations. Recently, Chatpatanasiri et al. [16] have proposed an alternative kernelization framework called *the KPCA trick*, which does not require a user to derive a new mathematical formula or re-implement a kernelized algorithm. Moreover, the KPCA trick framework avoids troublesome problems such as singularity, etc.

2.2.1 The KPCA-Trick Algorithm

In this section, the KPCA trick framework is extended to cover learners implemented under our semi-supervised learning framework. Let $k(\cdot, \cdot)$ be a PSD kernel function associated with a non-linear function $\phi(\cdot) : \mathbb{R}^{d_0} \mapsto \mathcal{H}$ such that $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ [26] where $\mathcal{H}$ is a Hilbert space. Denote ϕ_i for $\phi(\mathbf{x}_i)$ for $i = 1, ..., \ell + u$ and ϕ' for $\phi(\mathbf{x}')$. The central idea of the KPCA trick is to represent each ϕ_i and ϕ' in a new "finite"-dimensional space, with dimensionality bounded by $\ell + u$, without any loss of information. Within the framework, a new coordinate of each example is computed "explicitly", and each example in the new coordinate is then used as the input of any existing semi-supervised learner without any re-implementations.

To simplify the discussion, we assume that $\{\phi_i\}$ is linearly independent and has its center at the origin, i.e. $\sum_i \phi_i = 0$. Since we have $n = \ell + u$ total examples, the span of $\{\phi_i\}$ has dimensionality n by our assumption. According to [16], each example ϕ_i can be represented as $\varphi_i \in \mathbb{R}^n$ with respect to a new

orthonormal basis $\{\psi_i\}_{i=1}^{n}$ such that $span(\{\psi_i\}_{i=1}^{n})$ is the same as $span(\{\phi_i\}_{i=1}^{n})$ without loss of any information. More precisely, we define

$$\varphi_i = \Big(\langle\phi_i, \psi_1\rangle, \dots, \langle\phi_i, \psi_n\rangle\Big) = \Psi^T\phi_i. \tag{10}$$

where $\Psi = (\psi_1, \dots, \psi_n)$. Note that although we may be unable to numerically represent each ψ_i, an inner-product of $\langle\phi_i, \psi_j\rangle$ can be conveniently computed by KPCA where each ψ_i is a principal component in the feature space. Likewise, a new test point ϕ' can be mapped to $\varphi' = \Psi^T\phi'$. Consequently, the mapped data $\{\varphi_i\}$ and φ' are finite-dimensional and can be explicitly computed.

The KPCA-trick algorithm consisting of three simple steps is shown in Figure 3. All semi-supervised learners can be kernelized by this simple algorithm. In the algorithm, we denote a semi-supervised learner by ssl which outputs the best linear map A^*.

Input: 1. training examples: $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), \mathbf{x}_{\ell+1}, \dots, \mathbf{x}_{\ell+u}\}$
        2. a new example: $\mathbf{x}'$
        3. a kernel function: $k(\cdot, \cdot)$
        4. a linear semi-supervised learning algorithm: ssl (see Figure 1)
Algorithm:
(1) Apply $\mathsf{kpca}(k, \{\mathbf{x}_i\}_{i=1}^{\ell+u}, \mathbf{x}')$ such that $\{\mathbf{x}_i\} \mapsto \{\varphi_i\}$ and $\mathbf{x}' \mapsto \varphi'$.
(2) Apply ssl with new inputs $\{(\varphi_1, y_1), \dots, (\varphi_\ell, y_\ell), \varphi_{\ell+1}, \dots, \varphi_{\ell+u}\}$
    to achieve A^*.
(3) Perform kNN based on the distance $\|A^*\varphi_i - A^*\varphi'\|$.

Figure 3: The KPCA-trick algorithm for semi-supervised learning.

2.3 Remarks

1. The main optimization problem shown in Eq.(2) can be restated as follows: [12]

$$\underset{A\in\mathbb{R}^{d\times d_0}}{\arg\min}\ \mathrm{trace}\Big((ABA^T)^{-1}AX(D-C)X^TA^T\Big).$$

Within this formulation, the corresponding optimal solution is invariant under a non-singular linear transformation; i.e., if A^* is an optimal solution, then TA^* is also an optimal solution for any non-singular $T \in \mathbb{R}^{d\times d}$ [12, pp.447]. We note that four choices of T which assign a weight to each new axis are natural: (1) $T = I$, (2) T is a diagonal matrix with $T_{ii} = \frac{1}{\|\mathbf{a}^{(i)}\|}$, i.e. T normalizes each axis to be equally important, (3) T is a diagonal matrix with $T_{ii} = \sqrt{\lambda_i}$ as $\sqrt{\lambda_i}$ determines how well each axis $\mathbf{a}^{(i)}$ fits the objective function $\mathbf{a}^{(i)T}X(D-C)X^T\mathbf{a}^{(i)}$, and (4) T is a diagonal matrix with $T_{ii} = \frac{\sqrt{\lambda_i}}{\|\mathbf{a}^{(i)}\|}$, i.e. a combination of (2) and (3).

2. The matrices B defined in Subsection 2.1 of the two algorithms, SS-MFA and SS-LFDA, are guarantee to be positive semidefinite (PSD) but may not be positive definite (PD), i.e., B may not be full-rank. In this case, B is singular and we cannot immediately apply Eq.(3) to solve the optimization problems.

One common way to solve this difficulty is to use $(B + \epsilon I)$, for some value of ϵ, which is now guaranteed to be full-rank instead of B in Eq.(2). Since ϵ acts in a role of regularizer, it makes sense to set $\epsilon = \gamma$, the regularization parameter specified in Section 2.1. Similar settings of ϵ have also been used by some existing algorithms, e.g. [27, 14].

Also, in a small sample size problem where $X(D-C)X^T$ is not full-rank, the obtained matrix A^* (or some columns of A^*) lie in the null space of $X(D-C)X^T$. Although this matrix does optimize our optimization problem, it usually overfits the given data. One possible solution to this problem is to apply PCA to the given data in the first place [28] so that the resulted data have dimensionality less than or equal to the rank of $X(D-C)X^T$. Note that in our KPCA trick framework this pre-process is automatically accomplished as KPCA has to be applied to a learner as shown in Figure 3.

3   Related Work: Connection and Improvement

As we already described in Section 2.1, our framework generalizes various existing supervised and unsupervised manifold learners [11, 1, 2, 3, 4, 5, 6, 17, 9, 23]. The KPCA trick is new in the field of semi-supervised learning.

There are some supervised manifold learners which cannot be represented in our framework [18, 19, 20, 21, 22] because their cost functions are not linear with respect to distances among examples. Extension of these algorithms to handle semi-supervised learning problems is an interesting future work.

Yang et al. [29] present another semi-supervised learning framework which solves entirely different problems to problems considered in this paper. They propose to extend unsupervised algorithms such as ISOMAP [7] and Laplacian Eigenmap [13, Chapter 16] to cases to which information about exact locations of some points is available.

To the best of our knowledge, there are currently two existing semi-supervised dimensionality reduction frameworks in literatures which have similar goal to ours; both are very recently proposed. Here, we subsequently show that these frameworks can be restated as *special cases* of our framework.

3.1   Sugiyama et al. [14]

Sugiyama et al. [14] extends the LFDA algorithm to handle a semi-supervised learning problem by adding the PCA objective function $f^{PCA}(A)$ into the objective function $f^{\ell}(A)$ of LFDA described in Section 2.1. To describe Sugiyama et al.'s algorithm, namely 'SELF', without loss of generality, we assume that training data are centered at the origin, i.e. $\sum_{i=1}^{n} \mathbf{x}_i = 0$, and then we can write $f^{PCA}(A) = -\sum_{i=1}^{n} \|A\mathbf{x}_i\|^2$. Sugiyama et al. propose to solve the following problem:

$$A^* = \underset{ABA^T=I}{\arg\min} \left(\sum_{i,j=1}^{\ell} c_{ij}^{\ell} \|A\mathbf{x}_i - A\mathbf{x}_j\|^2 - \gamma \sum_{i=1}^{n} \|A\mathbf{x}_i\|^2 \right) \quad (11)$$

Interestingly, it can be shown that this formulation can be formulated in our framework with unlabel cost c_{ij}^u being *negative*, and hence our framework sub-

sumes SELF. To see this, let $c_{ij}^u = -1/2n$, for all $i, j = 1, ..., n$. Then, the objective $f^u(A)$ is equivalent to $f^{PCA}(A)$:

$$f^u(A) = \sum_{i,j=1}^n -\frac{1}{2n}\|A\mathbf{x}_i - A\mathbf{x}_j\|^2 = -\frac{1}{2n}\sum_{i,j=1}^n \langle A\mathbf{x}_i - A\mathbf{x}_j, A\mathbf{x}_i - A\mathbf{x}_j\rangle$$

$$= -\frac{1}{2n}\left(2\sum_{i,j=1}^n \langle A\mathbf{x}_i, A\mathbf{x}_i\rangle - 2\sum_{i,j=1}^n \langle A\mathbf{x}_i, A\mathbf{x}_j\rangle\right)$$

$$= -\frac{1}{2n}\left(2n\sum_{i=1}^n \|A\mathbf{x}_i\|^2 - 2\langle A\sum_{i=1}^n \mathbf{x}_i, A\sum_{j=1}^n \mathbf{x}_j\rangle\right)$$

$$= f^{PCA}(A),$$

where we use the fact that $\sum_{i=1}^n \mathbf{x}_i = 0$. This proves that SELF is a special case of our framework.

Note that the use of negative unlabel costs $c_{ij}^u = -1/2n$ results in an algorithm which tries to preserve a *global* structure of the input data and does not convey the manifold assumption where only a *local* structure should be preserved. Therefore, when the input unlabeled data lie in a complicated manifold, it is not appropriate to apply $f^u(A) = f^{PCA}(A)$.

3.2   Song et al. [15]

Song et al. propose to extend FDA and another algorithm named *maximum margin criterion* (MMC) [30] to handle a semi-supervised learning problem. Their idea of semi-supervised learning extension is similar to ours as they add the term $f^u(\cdot)$ into the objective of FDA and MMC (hence, we call them, SS-FDA and SS-MMC, respectively). However, SS-FDA and SS-MMC cannot handle problems where data of each class form a manifold or several clusters as shown in Figure 2 because SS-FDA and SS-MMC satisfy the condition (1) but not (1*). In fact, SS-FDA and SS-MMC can both be restated as instances of our framework. To see this, we note that the optimization problem of SS-MMC can be stated as

$$A^* = \underset{AA^T=I}{\arg\min}\ \ \gamma'\mathrm{trace}(AS_wA^T) - \mathrm{trace}(AS_bA^T) + \gamma f^u(A), \tag{12}$$

where S_b and S_w are standard between-class and within-class scatter matrices, respectively [12]:

$$S_w = \sum_{i=1}^c \sum_{j|y_j=i}(\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T \ \ \text{and}\ \ S_b = \sum_{i=1}^c (\boldsymbol{\mu} - \boldsymbol{\mu}_i)(\boldsymbol{\mu} - \boldsymbol{\mu}_i)^T,$$

where $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$, $\boldsymbol{\mu}_i = \frac{1}{n_i}\sum_{i=1}^{n_i} \mathbf{x}_i$ and n_i is the number of examples in the ith class. It can be checked that $\mathrm{trace}(AS_wA^T) = \sum_{i,j=1}^\ell c_{ij}^w\|A\mathbf{x}_i - A\mathbf{x}_j\|^2$ and $\mathrm{trace}(AS_bA^T) = \sum_{i,j=1}^\ell c_{ij}^b\|A\mathbf{x}_i - A\mathbf{x}_j\|^2$ where

$$c_{ij}^b = \begin{cases}(\frac{1}{n_k} - \frac{1}{n}), & \text{if } y_i = y_j = k, \\ -\frac{1}{n}, & \text{otherwise,}\end{cases} \ \ \text{and}\ \ c_{ij}^w = \begin{cases}\frac{1}{n_k}, & \text{if } y_i = y_j = k, \\ 0, & \text{otherwise.}\end{cases}$$
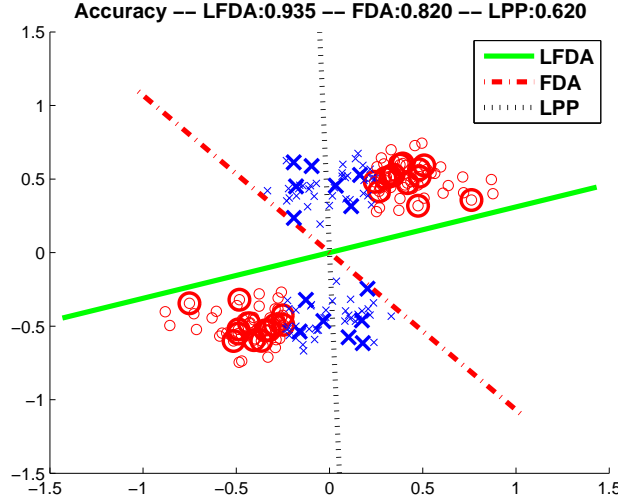
Figure 4: The first toy example. The projection axes of three algorithms, namely FDA, LFDA, and LPP, are presented. Big circles and big crosses denote labeled examples while small circles and small crosses denote unlabeled examples. Their percentage accuracy over the unlabeled examples are shown on the top.

Hence, by setting $c_{ij}^{\ell} = \gamma' c_{ij}^{w} - c_{ij}^{b}$ we finish our proof that SS-MMC is a special case of our framework. The proof that SS-FDA is in our framework is similar to that of SS-MMC.

3.3   Improvement over Previous Frameworks

In this section, we explain why SELF and SS-FDA proposed by Sugiyama et al. [14] and Song et al. [15] described above are *not* enough to solve some semi-supervised learning problems, even simple ones shown in Figure 4 and Figure 5.

In Figure 4, three dimensionality reduction algorithms, FDA, LFDA and LPP are performed on this dataset. Because of multi-modality, FDA cannot find an appropriate projection. Since the two clusters do not contain data of the same class, LPP which tries to preserve the structure of the two clusters also fails. In this case, only LFDA can find a proper projection since it can cope with multi-modality and can take into account the labeled examples. Note that since SS-FDA is a linearly combined algorithm of FDA and LPP, it can only find a projection lying in between the projections discovered by FDA and LPP, and in this case SS-FDA cannot find an efficient projection, unlike LFDA and, of course, SS-LFDA derived from our framework.

A similar argument can be given to warn an uncareful use of SELF in some situations. In Figure 5, four dimensionality reduction algorithms, FDA, PCA, LFDA and LPP are performed on this dataset. Because of multi-modality, FDA and PCA cannot find an appropriate projection. Also, since there are only a few labeled examples, LFDA fails to find a good projection as well. In this case, only LPP can find a proper projection since it can cope with multi-modality and can take the unlabeled examples into account. Note that since SELF is a linearly combined algorithm of LFDA and PCA, it can only find a projection lying in between the projections discovered by LFDA and PCA, and in this case SELF cannot find a correct projection, unlike a semi-supervised learner like SS-LFDA derived from our framework which, as explained in Section 2.1, employs
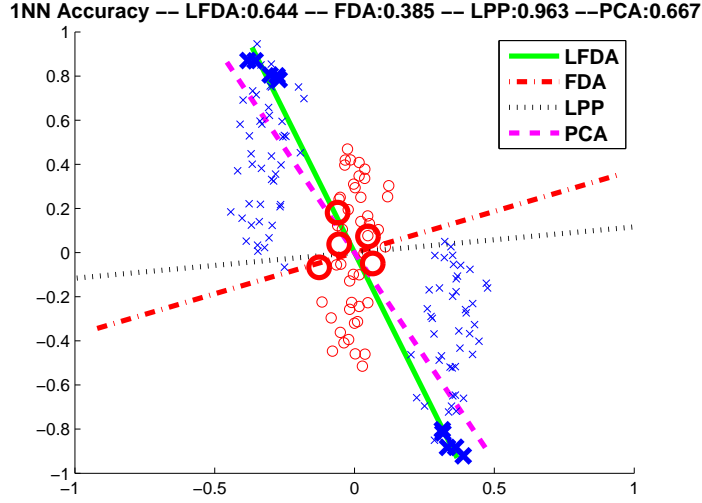
1NN Accuracy –– LFDA:0.644 –– FDA:0.385 –– LPP:0.963 ––PCA:0.667

Figure 5: The second toy example consisting of three clusters of two classes.

the LPP cost function as its C^u.

Since a semi-supervised manifold learner derived from our framework can be intuitively thought of as a combination of a supervised learner and an unsupervised learner. One may misunderstand that a semi-supervised learner cannot discover a good subspace if neither is a supervised learner nor an unsupervised learner able to discover a good subspace. The above two toy examples may also mislead the readers in that way. In fact, that intuition is incorrect. Here, we give another toy example shown in Figure 6 where only a semi-supervised learner is able to discover a good subspace but neither is its supervised and unsupervised counterparts. Intuitively, a semi-supervised learner is able to exploit useful information from both labeled and unlabeled examples.

4   Experiments

In this section, classification performances of algorithms derived from our framework are demonstrated. We try to use a similar experimental setting as those in previous works [14, 13, Chapter 21] so that our results can be compared to them.

4.1   Experimental Setting

In all experiments, two semi-supervised learners, SS-LFDA and SS-DNE, derived from our framework are compared to relevant existing algorithms, PCA, LPP*, LFDA, DNE and SELF [14]. In contrast to the standard LPP which does not apply the Hadamard power operator explained in Section 2.1, we denote LPP* as a variant of LPP applying the Hadamard power operator.

Non-linear semi-supervised manifold learning is also experimented by applying the KPCA trick algorithm illustrated in Figure 3. Since it is not our intention to apply the "best" kernel *but* to compare efficiency between a "semi-supervised" kernel learner and its base "supervised" (and "unsupervised") kernel learners, we simply apply the 2nd-degree polynomial kernel $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^2$ to the kernel algorithms in all experiments.
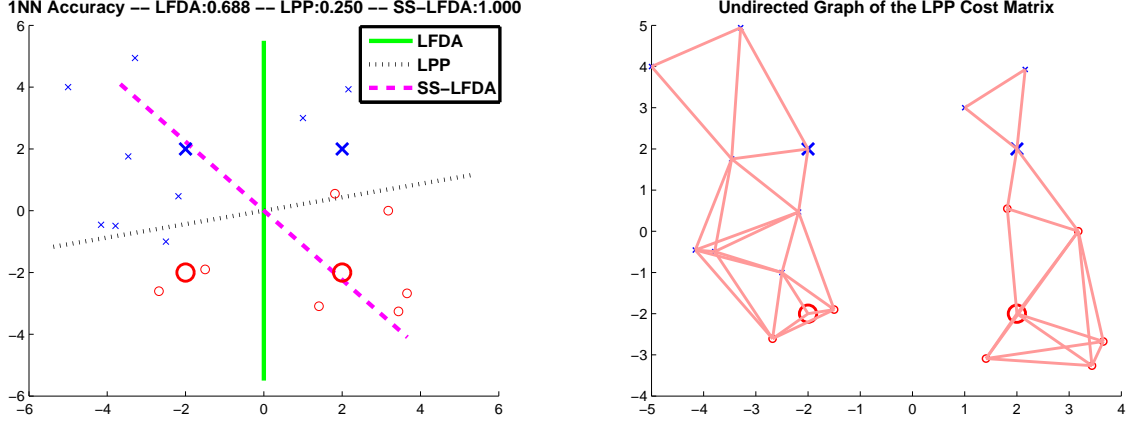
Figure 6: (Left) The third toy example where only a semi-supervised learner is able to find a good projection. (Right) An undirected graph corresponding to the values of C^u used by LPP and SS-LFDA. In this figure, a pair of examples i and j has a link if and only if $c_{ij}^u > 0.1$. This graph explains why LPP projects the data in the axis shown in the left figure; LPP, which does not apply the label information, tries to choose a projection axis which squeezes the two clusters as much as possible. Note that we apply a local-scaling method, Eq.(7), to specify C^u.

By using the nearest neighbor algorithm on their discovered subspaces, classification performances of the experimented learners are measured on five standard datasets shown on Table 1, the first two datasets are obtained from the UCI repository [31], the next two datasets mainly designed for testing a semi-supervised learner are obtained from http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html [13, Chapter 21]. The final dataset, *extended Yale B* [32], is a standard dataset of a face recognition task. The classification performance of each algorithm is measured by the average test accuracy over 25 realizations of randomly splitting each dataset into training and testing subsets.

Three parameters are needed to be tuned in order to apply a semi-supervised learner derived from our framework (see Section 2.1): γ, the regularizer, α, the degree of the Hadamard power operator and k, the kth-nearest neighbor parameter needed to construct the cost matrices. To make our learners satisfy the condition (1*) described in Section 2.1, it is clear that k should be small compared to n_c, the number of training examples of class c. From our experience, we found that semi-supervised learners are quite insensitive to various small values of k. Therefore, in all our experiments, we simply set $k = \min(3, n_c)$ so that only two parameters, γ and α, are needed to be tuned. We tune these two parameters via cross validation. Note that only α is needed to be tuned for LPP* and only γ is needed to be tuned for SELF.

The 'GOOD NEIGHBORS' score shown in Table 1 is due to Sugiyama et al. [14]. The score is simply defined as a training accuracy of the nearest neighbor algorithm when *all available data are labeled and are given to the algorithm*. Note that this score is not used by a dimensionality reduction algorithm. It just clarifies a usefulness of unlabeled examples of each dataset to the readers. Intuitively, if a dataset gets a high score, unlabeled examples should be useful since it indicates that each pair of examples having a high penalty cost c_{ij}^u should

Table 1: Details of each dataset: d_0, c, ℓ, u and t denote the numbers of input features, classes, labeled examples, unlabeled examples and testing examples, respectively. '*' denotes the transductive setting used in small datasets, where all examples which are not labeled are given as unlabeled examples and used as testing examples as well. d, determined by using prior knowledge, denotes the target dimensionality for each dataset. "GOOD NEIGHBORS" denotes a quantity which measures a goodness of unlabeled data for each dataset.

NAME	d_0	c	$\ell + u + t$	ℓ	u	d	GOOD NEIGHBORS	
							LINEAR	KERNEL
IONOSPHERE	34	2	351	10/100	*	2	0.866	0.843
BALANCE	4	3	625	10/100	300	1	0.780	0.760
BCI	117	2	400	10/100	*	2	0.575	0.593
USPS	241	2	1500	10/100	300	10	0.969	0.971
M-EYALE	504	5	320	20/100	*	10	0.878	0.850

belong to the same class. Note that on Table 1 there are two scores for each dataset: LINEAR is a score on a given input space while KERNEL measures a score on a feature space corresponding to the 2nd-degree polynomial kernel.

4.2 Numerical Results

Numerical results are shown in Table 2 for the case of $\ell = 10$ (except M-EYALE where $\ell = 20$) and Table 3 for the case of $\ell = 100$. In experiments, SS-DNE and SS-LFDA are compared their classification performances to their unsupervised and supervised counterparts: LPP* and DNE for SS-DNE, and LPP* and LFDA for SS-LFDA. SELF is also compared to SS-LFDA as they are related semi-supervised learners originated from LFDA. Our two algorithms will be highlighted if they are superior to their counterpart opponents.

From the results, our two algorithms, SS-LFDA and SS-DNE, outperform all their opponents in 32 out of 40 comparisons: in the first setting of small ℓ (Table 2), our algorithms outperform the opponents in 18 out of 20 comparisons while in the second setting of large ℓ (Table 3), our algorithms outperform the opponents in 14 out of 20 comparisons. Consequently, our framework offers a semi-supervised learner which consistently improves its base supervised and unsupervised learners.

Note that as the number of labeled examples increases, usefulness of unlabeled examples decreases. We will subsequently discuss and analyze the results of each dataset in details in the next subsections.

4.2.1 Ionosphere

IONOSPHERE is a real-world dataset of radar pulses passing through the ionosphere which were collected by a system in Goose Bay, Labrador. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not. Since we do not know the true decision boundary of IONOSPHERE, we simply set the target dimensionality $d = c = 2$. It can be observed that non-linearization does improve the classification performance of all algorithms.

Table 2: Percentage accuracies of SS-DNE and SS-LFDA derived from our framework compared to existing algorithms ($\ell = 10$, except M-Eyale where $\ell = 20$). SS-LFDA and SS-DNE are highlighted when they outperform their opponents (LPP* and DNE for SS-DNE, and LPP*, LFDA and SELF for SS-LFDA). Superscripts indicate %-confidence levels of the one-tailed paired t-test for differences in accuracies between our algorithms and their best opponents. No superscripts denote confidence levels which below 80%.

LINEAR	PCA	LPP*	DNE	LFDA	SELF	SS-DNE	SS-LFDA
IONOSPHERE	71±1.2	82±1.3	70±1.2	71±1.1	70±1.5	75±1.0	78.1±.9
BALANCE	49±1.9	61±1.9	63±2.2	70±2.2	69±2.3	**71±1.8**^{99}	**73±2.3**^{80}
BCI	49.8±.6	53.4±.3	51.3±.6	52.6±.5	52.1±.5	**57.1±.6**^{99}	**55.2±.3**^{99}
USPS	79±1.2	74±1.0	79.6±.6	80.6±.9	81.7±.8	**81.8±.5**^{99}	**83.0±.5**^{90}
M-EYALE	44.6±.7	67±1.1	66±1.2	71.6±1.0	67.2±.8	**76.9±.8**^{99}	**75.7±.9**^{99}
KERNEL	PCA	LPP*	DNE	LFDA	SELF	SS-DNE	SS-LFDA
IONOSPHERE	70±1.8	83.2±.9	70±1.6	71±1.3	74±1.5	**87.2±.9**^{99}	**88±1.0**^{99}
BALANCE	41.7±.8	47.9±.9	62±2.5	66±2.0	60±2.8	**66±1.8**^{80}	**69±1.9**^{80}
BCI	49.7±.3	53.7±.3	50.1 ±.4	50.3±.6	50.5±.4	**53.8±.3**	**54.1±.3**^{80}
USPS	77±1.1	76±1.1	79.9±.5	80.3±.8	80.9±.8	**82.0±.4**^{99}	**83.7±.6**^{99}
M-EYALE	42.1±.9	63.2±.7	58.0±.9	60.3±.8	58.8±.7	**69.9±.7**^{99}	**73.2±.8**^{99}

Table 3: Percentage accuracies of SS-DNE and SS-LFDA compared to existing algorithms ($\ell = 100$).

LINEAR	PCA	LPP*	DNE	LFDA	SELF	SS-DNE	SS-LFDA
IONOSPHERE	72.8±.6	83.7±.6	77.9±.7	74±1.0	77.8±.5	**84.5±.6**^{80}	**84.9±.4**^{95}
BALANCE	57±2.2	80±1.3	86.4±.5	87.9±.3	87.2±.4	**88.2±.5**^{99}	86.3±.6
BCI	49.5±.5	54.9±.5	53.1±.7	67.9±.5	67.6±.6	**63.1±.5**^{99}	67.5±.6
USPS	91.4±.3	75.7±.3	91.1±.3	89.3±.4	92.2±.3	**92.2±.4**^{95}	91.6±.3
M-EYALE	69.4±.4	84.1±.4	92.3±.4	95.4±.3	94.3±.2	**93.5±.4**^{95}	**95.7±.2**
KERNEL	PCA	LPP*	DNE	LFDA	SELF	SS-DNE	SS-LFDA
IONOSPHERE	79.8±.4	89.7±.5	78.7±.9	81.3±.7	81.1±.5	**93.6±.2**^{99}	**93.7±.3**^{99}
BALANCE	42.5±.3	46.9±.5	84.0±.7	87.8±.7	79±1.6	**86.5±.7**^{99}	87.7±.9
BCI	49.7±.5	54.5±.4	51.6±.6	51.0±.8	52.4±.6	**57.6±.2**^{99}	**57.0±.4**^{99}
USPS	91.1±.3	81.5±.6	91.4±.4	91.2±.4	92.7±.3	**92.3±.3**^{95}	91.9±.3
M-EYALE	66.3±.3	81.9±.5	91.2±.3	89.1±.5	85.8±.6	91.2±.3	**94.3±.3**^{99}

It can be observed that LPP* is much better than PCA on this dataset, and therefore, unlike SELF, SS-LFDA much improves LFDA. In fact, the main reason that SS-LFDA, SS-DNE and LPP* have good classification performances are because of the Hadamard power operator. This is explained in Figures 7, 8 and 9. From Figures 7 and 8, defining "nearby examples" be a pair of examples with a link (having $c_{ij}^u \geq 0.36$), we see that *almost every link connects nearby examples of the same class* (i.e. connects good nearby examples). This indicates that our unlabel cost matrix C^u is quite accurate as bad nearby examples rarely have links. In fact, the *ratio of good nearby examples per total nearby examples* (shortly, the good-nearby-examples ratio) is $394/408 \approx 0.966$. Nevertheless, if we re-define "nearby examples" be a pairs of examples having, e.g., $c_{ij}^u \geq 0.01$, the same ratio then reduces to 0.75 as shown in Figure 9 (Left). This indicates that many pairs of examples having small values of c_{ij}^u are of different classes (i.e. bad nearby examples).
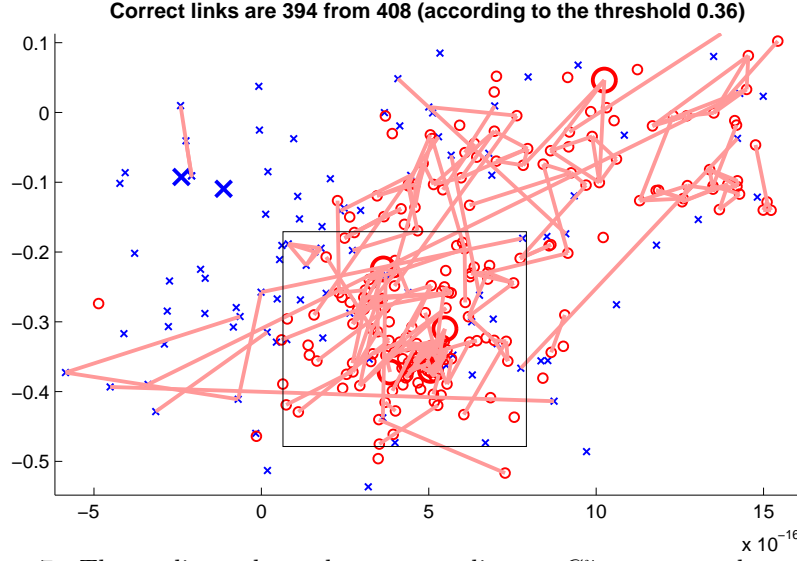
Correct links are 394 from 408 (according to the threshold 0.36)

Figure 7: The undirected graph corresponding to C^u constructed on IONO-SPHERE. Each link corresponds to a pair of nearby examples having $c_{ij}^u \geq 0.36$. The number '0.36' is just chosen for visualizability.

Since an algorithm derived from our framework minimizes the *cost-weighted average* distances of every pair of examples (see Eq. (2) and its derivation), it is beneficial to further increases the cost of a pair having large c_{ij}^u (since it usually corresponds to a pair of the same class) and decreases the cost of of a pair having small c_{ij}^u. From Eq. (9), it can be easily seen that the effect of the Hadamard power operator is exactly what we need. The good-nearby-examples ratios after applying the Hadamard power operator with $\alpha = 8$ are illustrated in Figure 9 (Right). Notice that, after applying the operator, even pairs with small values of c_{ij}^u are usually of the same class.

4.2.2 Balance

BALANCE is an artificial dataset which was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The 4 attributes containing integer values from 1 to 5 are LEFT_WEIGHT, LEFT_DISTANCE, RIGHT_WEIGHT, and RIGHT_DISTANCE. The correct way to find the class is the greater of (LEFT_DISTANCE × LEFT_WEIGHT) and (RIGHT_DISTANCE × RIGHT_WEIGHT). If they are equal, it is balanced. Therefore, there are $5^4 = 625$ total examples and 3 classes in this dataset. Moreover, the correct decision surface is 1-dimensional manifold lying in the feature space corresponding to the $\langle \cdot, \cdot \rangle^2$ kernel so that we set the target dimensionality $d = 1$.

This dataset illustrates another flaw of using PCA in a classification task. After centering, the covariance matrix of the 625 examples is just a multiple of I, the identity matrix. Therefore, any direction is a principal component with largest variance, and PCA is just return a random direction! Hence, we cannot expect much about the classification performance of PCA in this dataset. Thus, PCA cannot help SELF improves much the performance on LFDA, and sometimes SELF degrades the performance of LFDA due to overfitting. In
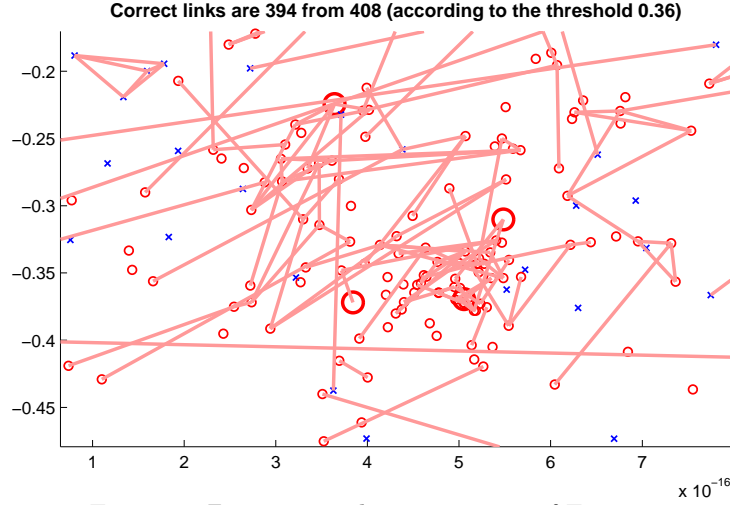
Figure 8: Zoom-in on the square area of Figure 8.

contrast, SS-LFDA often improves the performance of LFDA. Also, SS-DNE is able to improve the classification performance of DNE and LPP* in all settings.

4.2.3 BCI

This dataset originates from the development of a Brain-Computer Interface where a single person performed 400 trials in each of which he imagined movements with either the left hand (the 1st class) or the right hand (the 2nd class). In each trial, electroencephalography (EEG) was recorded from 39 electrodes. An autoregressive model of order 3 was fitted to each of the resulting 39 time series. The trial was represented by the total of $117 = 39*3$ fitted parameters. The target dimensionality is set to the number of classes, $d = c = 2$. Similar to the previous datasets, SS-LFDA and SS-DNE are usually able to outperform their opponents. Again, PCA is not appropriate for this real-world dataset, and hence SELF is inferior to SS-LFDA.

4.2.4 USPS

This benchmark is derived from the famous USPS dataset of handwritten digit recognition. For each digit, 150 images are randomly drawn. The digits '2' and '5' are assigned to the first class, and all others form the second class. To prevent a user to employ a domain knowledge of the data, each example is rescaled, noise added, dimension masked and pixel shuffled [13, Chapter 21]. Although there are only 2 classes in this dataset, the original data presumably form 10 clusters, one for each digit. Therefore, the target dimension d is set to 10.

Often, SS-LFDA and SS-DNE outperform their opponents. Nevertheless, note that SS-LFDA and SS-DNE do not improve much on LFDA and DNE when $\ell = 100$ because 100 labeled examples are quite enough to discriminating the data and therefore unlabeled examples offer relatively small information to semi-supervised learners.
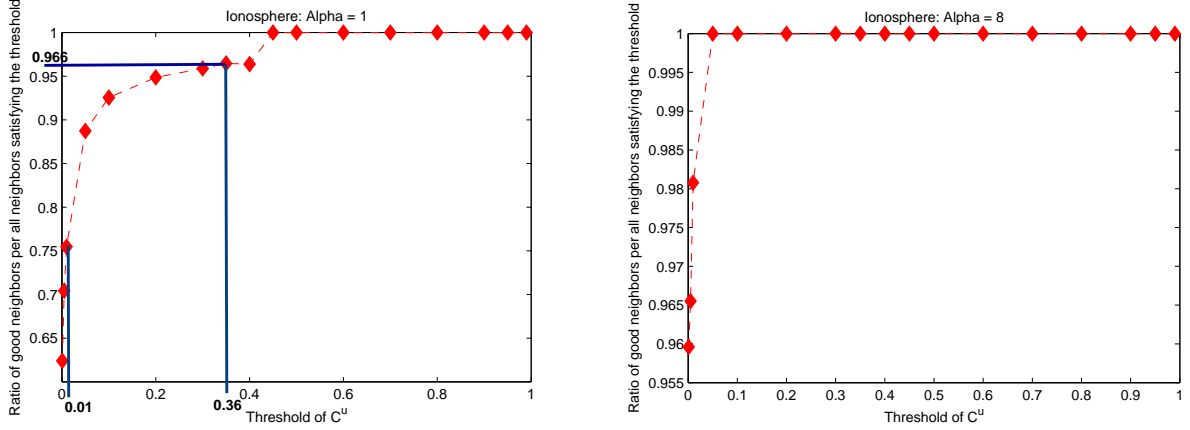
Figure 9: For each number x in the the x-axis, its corresponding value on the y-axis is the *ratio between the number of good nearby examples* (having $c_{ij}^u > x$ and belonging to the same class) *and the number of nearby examples* (having $c_{ij}^u > x$). The ratios with respect to C^{u^α} are demonstrated where (Left) $\alpha = 1$ (the standard LPP), and where (Right) $\alpha = 8$ (LPP*).

4.2.5 M-Eyale

This face recognition dataset is derived from *extended Yale B* [32]. There are 28 human subjects under 9 poses and 64 illumination conditions. In our M-EYALE (Modified Extended Yale B), we randomly chose ten subjects, 32 images per each subject, from the original dataset and down-sampling each example to be of size 21×24 pixels.

M-EYALE consists of 5 classes where each class consists of images of two randomly-chosen subjects. Hence, there should be two separated clusters for each class, and we should be able to see the advantage of algorithms employing the conditions (1*) and (2*) explained in Section 2.1. In this dataset, the number of labeled examples of each class is fixed to $\frac{\ell}{c}$ so that examples of all classes are observed. Since this dataset should consist of ten clusters, the target dimensionality is set to $d = 10$.

It is clear that LPP* performs much better than PCA in this dataset. Recall that PCA captures maximum-variance directions; nevertheless, in this face recognition task, maximum-variance directions are not discriminant directions but directions of lighting and posing [28]. Therefore, PCA captures totally wrong directions, and hence PCA degrades the performance of SELF from LFDA. In contrast, LPP* much better captures local structures in the dataset and discover much better subspaces. Thus, by cooperating LPP* with LFDA and DNE, SS-LFDA and SS-DNE are able to obtain very good performances.

5 Conclusion

We have presented a unified semi-supervised learning framework for linear and non-linear dimensionality reduction algorithms. Advantages of our framework are that it generalizes existing various supervised, unsupervised and semi-supervised learning frameworks employing spectral methods. Empirical evidences showing satisfiable performance of algorithms derived from our framework have been reported on standard datasets.

References

[1] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and S. Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):40–51, 2007.

[2] Wei Zhang, Xiangyang Xue, Zichen Sun, Yue-Fei Guo, and Hong Lu. Optimal dimensionality of metric space for classification. In *International Conference on Machine Learning*, 2007.

[3] Deng Cai, Xiaofei He, and Jiawei Han. Spectral regression for efficient regularized subspace learning. In *Computer Vision and Pattern Recognition*, 2007.

[4] Steven C. H. Hoi, Wei Liu, Michael R. Lyu, and Wei-Ying Ma. Learning distance metrics with contextual constraints for image retrieval. In *Computer Vision and Pattern Recognition*, 2006.

[5] Hwann-Tzong Chen, Huang-Wei Chang, and Tyng-Luh Liu. Local discriminant embedding and its variants. In *Computer Vision and Pattern Recognition*, volume 2, 2005.

[6] J. Cheng, Q. Liu, H. Lu, and Y.W. Chen. A supervised nonlinear local embedding for face recognition. *International Conference on Image Processing*, 1, 2004.

[7] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, December 2000.

[8] S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding, 2000.

[9] X. He and P. Niyogi. Locality Preserving Projections. In *Advances in Neural Information Processing Systems 16*, 2004.

[10] L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee. Spectral methods for dimensionality reduction. *Semisupervised Learning. MIT Press: Cambridge, MA*, 2006.

[11] M. Sugiyama. Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis. *The Journal of Machine Learning Research*, 8:1027–1061, 2007.

[12] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[13] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. The MIT Press, 2006.

[14] M. Sugiyama, T. Ide, S. Nakajima, and J. Sese. Semi-Supervised Local Fisher Discriminant Analysis for Dimensionality Reduction. *PAKDD*, pages 333–344, 2008.

[15] Yangqiu Song, Feiping Nie, Changshui Zhang, and Shiming Xiang. A Unified framework for semi-supervised dimensionality reduction. *Pattern Recognition*, 41:2789–2799, 2008.

[16] Ratthachat Chatpatanasiri, Teesid Korsrilabutr, Pasakorn Tangchanacha-ianan, and Boonserm Kijsirikul. On Kernelizing Mahalanobis Distance Learning Algorithms. *Arxiv preprint. http://arxiv.org/abs/0804.1441*, 2008.

[17] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[18] Jacob Goldberger, Sam Roweis, Geoffrey Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. *NIPS*, pages 513–520, 2005.

[19] A. Globerson and S. Roweis. Metric learning by collapsing classes. *NIPS*, 18:451–458, 2006.

[20] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. *NIPS*, 18:1473–1480, 2006.

[21] Liu Yang, Rong Jin, Rahul Sukthankar, and Yi Liu. An efficient algorithm for local distance metric learning. In *AAAI*, 2006.

[22] Lorenzo Torresani and Kuang C. Lee. Large margin component analysis. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1385–1392. MIT Press, Cambridge, MA, 2007.

[23] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. *Advances in Neural Information Processing Systems*, 17:1601–1608, 2004.

[24] James R. Schott. *Matrix Analysis for Statistics*. Wiley, 2nd edition, 2005.

[25] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, June 2004.

[26] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. The MIT Press, December 2001.

[27] Jerome H. Friedman. Regularized Discriminant Analysis. *Journal of the American Statistical Association*, 84:165–175, 1989.

[28] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[29] X. Yang, H. Fu, H. Zha, and J. Barlow. Semi-supervised nonlinear dimensionality reduction. In *ICML*, pages 1065–1072, 2006.

[30] Haifeng Li, Tao Jiang, and Keshu Zhang. Efficient and Robust Feature Extraction by Maximum Margin Criterion. *IEEE Transactions on Neural Networks*, 17(1):157–165, 2006.

[31] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007.

[32] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.