Global Linear Neighborhoods for Efficient Label Propagation

Ze Tian†
Rui Kuang†

Abstract
Graph-based semi-supervised learning improves classification by combining labeled and unlabeled data through label propagation. It was shown that the sparse representation of graph by weighted local neighbors provides a better similarity measure between data points for label propagation. However, selecting local neighbors can lead to disjoint components and incorrect neighbors in graph, and thus, fail to capture the underlying global structure. In this paper, we propose to learn a nonnegative low-rank graph to capture global linear neighborhoods, under the assumption that each data point can be linearly reconstructed from weighted combinations of its direct neighbors and reachable indirect neighbors. The global linear neighborhoods utilize information from both direct and indirect neighbors to preserve the global cluster structures, while the low-rank property retains a compressed representation of the graph. An efficient algorithm based on a multiplicative update rule is designed to learn a nonnegative low-rank factorization matrix minimizing the neighborhood reconstruction error. Large scale simulations and experiments on UCI datasets and high-dimensional gene expression datasets showed that label propagation based on global linear neighborhoods captures the global cluster structures better and achieved more accurate classification results.

1 Introduction
In many real world applications of data mining and machine learning, much more unlabeled data could be obtained and utilized with the labeled data to build more robust and accurate classification models. Graph-based semi-supervised learning algorithms perform a combined analysis of labeled and unlabeled data in a similarity graph between the data points [17, 24, 8, 23]. The algorithms initialize the vertices of the labeled data with +1/-1 and the vertices of the unlabeled data with 0, and then assign labels to the unlabeled data by label propagation on graph. The label information on the vertices is iteratively propagated between the neighboring vertices and the propagation process will finally converge to the unique global optimum minimizing a quadratic criterion [4]. These algorithms are based on the “cluster assumption”: nearby data points in the graph should be labeled similarly and data points in the same global cluster in the graph should also be labeled similarly.

How to estimate the similarities between data points is a core problem in the label propagation algorithms. The commonly used measure is Gaussian kernel similarity, which is sensitive to the variance parameter. A more rigorous treatment of the problem is to learn a similarity measure based on relations between data points. Under the assumption of linear neighborhoods, i.e. each data point can be reconstructed by a weighted linear combination of its neighbors, [19] proposed to represent the graph by a sparse matrix of the positive linear coefficients for reconstructing a data point by the k-nearest neighbors. Furthermore, [6] and [22] proposed to select the reconstruction neighbors with L1 norm constraint to learn a sparse matrix. A limitation of these methods is that the neighbors are selected “locally”—the decision is only based on the individual relation between the reconstructed data point and the other data points. The local neighbors can lead to disjoint components and incorrect neighbors in the graph. We demonstrate the problems in a toy example in Fig. 1(a). In this example, the algorithm proposed by [19] is applied to classify data points from two moon clusters shown in Fig. 1(a). When k is too small, several disconnected subgraphs (components) are observed in the top plots in Fig. 1(b). Since there is no label information that can be propagated into the components, the data points in the components are not classifiable by label propagation in the bottom plots in Fig. 1(b). Note that, since the learned similarity matrix is not symmetric, a connected graph might not be strongly connected (Some of the vertices are not reachable due to the direction of the edges). When k is too large such as k = 5 in Fig. 1(b), edges are introduced to connect data points in different classes. The classification performance will deteriorate with the wrong neighborhood information. In both cases, instead of providing a better similarity measure for label propagation, the new similarity matrix fails to capture the underlying global structure of the data.

In this paper, we propose to learn a nonnegative
In graph-based semi-supervised learning, a similarity graph is used to capture global linear neighborhoods. The local linear neighborhoods assume that each data point can be linearly reconstructed from weighted combinations of its direct neighbors and indirect neighbors reachable by any steps of random walk. The influence of the neighbors is controlled by a decay factor inversely exponentially proportional to the number of steps of random walk. The larger the distance, the smaller the influence to the reconstruction. The global linear neighborhoods preserve the global cluster structures by exploring both the direct neighbors and the indirect neighbors, and thus, rarely contain isolated components. We demonstrate that global linear neighborhoods can be approximated by a low-rank factorization of the unknown reconstruction matrix. Therefore, even if the reconstruction matrix of global linear neighborhoods is not sparse, the low-rank factorization still provides a compressed representation of the graph. We further demonstrate that, to require all entries to be nonnegative edge weights in a graph for label propagation, the low-rank factorization matrix can be efficiently learned by a multiplicative update rule that minimizes the reconstruction error at a local optimum.

2 Related Work

In a given dataset $\mathcal{X} = \{x_1, \ldots, x_l, \ldots, x_n\}$ and a given label set $\mathcal{L} = \{1, \ldots, c\}$, $\{x_1, \ldots, x_l\}$ are data points in $\mathbb{R}^m$ labeled by $\{y_1, \ldots, y_l|y_l \in \mathcal{L}, l = 1, \ldots, l\}$ and $\{x_{l+1}, \ldots, x_n\}$ are unlabeled data points in $\mathbb{R}^m$. In graph-based semi-supervised learning, a similarity graph $G = (V, E)$ is first constructed from the dataset $\mathcal{X}$, where the vertex set $V = \mathcal{X}$ and the edges $E$ are weighted by adjacency matrix $W$. By relaxing the class label variables as real numbers, label propagation algorithm iteratively updates the predicted label $Z$ by

$$ Z^{t+1} = \alpha W Z^t + (1 - \alpha) Y, $$

where $t$ is the time step, $\alpha \in (0, 1)$ is a constant and $Y$ is a $n \times c$ binary matrix encoding the labeling of data points against each class. To derive a reliable $W$ for label propagation, simple strategies are suggested in [1]. A graph is constructed by choosing the nearest neighbors with

- $\epsilon$-neighbors: Nodes $i$ and $j$ are neighbors if the Euclidean distance $\|x_i - x_j\|^2 < \epsilon$.
- $k$-nearest neighbors: Nodes $i$ and $j$ are neighbors if $i$ is among the $k$-nearest neighbors of $j$ or vice versa.

After the neighborhood of each node is decided, the edges between the neighbors are weighted by $W_{ij} = 1$ or by a Gaussian kernel $W_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$. However, as it was analyzed by [19] and [3], the difficulty to determine the optimal $\sigma$ causes unstable classification performance by label propagation.

Based on ideas similar to Local Linear Embedding (LLE) [14], [19], [6] and [22] proposed several approaches for estimating more robust similarity matrix $W$ for label propagation. These methods are based on
a common linear neighborhood assumption that each data point can be reconstructed by a weighted linear combination of its selected neighbors, i.e. 
\[ x_i = \sum_{j: x_j \in N(x_i)} W_{ij} x_j, \]
where \( N(x_i) \) is the neighborhood of node \( i \). Linear Neighborhood Propagation (LNP) proposed by [19] learns the \( W \) matrix by minimizing the reconstruction error:

\[
\varepsilon(W) = \sum_i \left\| x_i - \sum_{j: x_j \in N(x_i)} W_{ij} x_j \right\|^2
\]

subject to \( \sum_{j: x_j \in N(x_i)} W_{ij} = 1 \), \( W_{ij} \geq 0 \) and \( W_{ij} = 0 \) if \( x_j \notin N(x_i) \). \( N(x_i) \) is chosen as the set of \( k \)-nearest neighbors of \( x_i \) in Euclidean distance. To avoid a fixed neighborhood for each point, Sparsity Induced Similarity measure (SIS) was proposed by [20] to learn \( W \) regularized by \( L_1 \) norm as follows,

\[
\min_W \left\| W \right\|_{l_1}
\]

subject to \( x_i = \sum_{j: x_j \in N(x_i)} W_{ij} x_j \) for any \( i \). By minimizing the \( L_1 \) norm, \( W \) is expected to be a sparse adjacency matrix. The approach proposed by [22] is similar to SIS, based on an algorithm for finding nonnegative sparse representation. [21] also proposed a slightly different objective function to deduce a sparse \( W \) and used a parameter-free label propagation algorithm for semi-supervised learning.

The “local neighborhood” assumption is similar to Markov assumption (conditional independence), where each sample is not related to any samples other than its \( k \)-nearest neighbors. A common problem of the above approaches is that, when \( k \) is very small or a certain sparsity level is expected, the graph edges defined by a few selected neighbors might define a weakly connected graph or more severely, separate the graph into several disjoint components (Fig. 1). In general, the limitation is introduced by neighborhood selection with local measures. In other words, these algorithms tend to select a few most similar data points as neighbors and the connectivities of these neighbors with other nodes are not considered. To see this mathematically, each row of the sparse matrix representing the local neighborhoods is actually learned independently of the other rows as in LLE. Another limitation is that the learned \( W \) is not a symmetric matrix that could be interpreted for cluster structures. To ensure symmetry, \( W \) can be symmetrized as \( \frac{W_{ij} + W_{ji}}{2} \). However, the symmetrization might not be reasonable when \( W_{ij} \) and \( W_{ji} \) are very different.

3 The Algorithm
Let \( X \) be the \( n \times m \) data matrix where \( X_{ij} \) is the value of the \( j \)th feature in the \( i \)th data point. Let \( W \) be the symmetric \( n \times n \) similarity matrix between the data points to be learned. Instead of explicit selecting \( k \) neighbors to make \( W \) sparse as in Eqn. (2.1), or enforcing the sparsity by minimizing the \( L_1 \)-norm as in Eqn. (2.2), we propose to learn a rank-\( k \) non-negative symmetric \( W \) by the following optimization problem:

\[
\min Q(F) = \left\| X - F F^T X \right\|^2
\]

subject to \( F_{ij} \geq 0 \) where \( F \) is a \( n \times k \) matrix and \( W = FF^T \). We first explain the motivation of Eqn. (3.3) under global linear neighborhoods.

3.1 Global Linear Neighborhood. Suppose \( W \) is the compressed representation of the graph. Let \( S = D^{-1/2} WD^{-1/2} \), where \( D \) is a diagonal matrix with \( D_{ii} \) equal to the sum of the \( i \)th row of \( W \). In global linear neighborhoods, we assume each data point can be reconstructed by itself, its neighbors, its neighbors’ neighbors, etc. The influence of the neighbors is decayed by a factor \( \alpha \) for each increase in distance. Specifically, we assume the data can be reconstructed by global linear neighborhoods as follows:

\[
X = (1-\alpha)(X + \alpha S X + \alpha^2 S^2 X + \ldots) = (1-\alpha)\sum_{i=0}^{\infty} (\alpha S)^i X,
\]

where \( \alpha \in (0,1) \). This assumption is a generalization of local linear neighborhood [14], in which each data point is only reconstructed by its selected neighbors (the \( i = 1 \) term in global linear neighborhood). Since \( \sum_{i=0}^{\infty} (\alpha S)^i X = (I - \alpha S)^{-1} X \), it is equivalent to find \( S \) to minimize

\[
\left\| X - (1-\alpha)(I - \alpha S)^{-1} X \right\|^2.
\]

Since \( S = D^{-1/2} WD^{-1/2} \) is a real symmetric matrix, its eigen decomposition can be written as \( S = Q \Lambda Q^T \) where \( Q \) is an orthonormal matrix and \( \Lambda \) is real and diagonal. By replacing \( S \) with \( Q \Lambda Q^T \),

\[
(1-\alpha)(I - \alpha S)^{-1} X = (1-\alpha)(I - \alpha Q \Lambda Q^T)^{-1} X = (1-\alpha)Q(I - \alpha \Lambda)^{-1} Q^T X.
\]

Since \( S \) is similar to a stochastic matrix, the diagonal entries in \( \Lambda \) are in \([-1,1]\). Thus, \( (I - \alpha \Lambda)^{-1} \) is a positive definite diagonal matrix with diagonal entries in \( [\frac{1}{1-\alpha},\frac{1}{1-\alpha}] \). To introduce a compressed representation, we only keep the \( k \) principle components of \( (I - \alpha S)^{-1} \). Actually, the \( k \) principle components of \( (I - \alpha S)^{-1} \) are the same \( k \) components of \( S \) corresponding to the \( k \) largest eigenvalues. Given the eigen decomposition in Eqn. (3.5), keeping the first \( k \) principle components of \( (I - \alpha S)^{-1} \) is equivalent to keeping
the top $k$ entries only on the diagonal of $(I - \alpha \Lambda)^{-1}$, denoted by a $k \times k$ diagonal matrix $(I - \alpha \Lambda)^{-1}_{(k)}$. Let $F_{(k)} = \sqrt{1 - \alpha} Q_{(k)} (I - \alpha \Lambda)^{-1/2}_{(k)}$, where $Q_{(k)}$ are the corresponding $k$ columns for the $k$ principle components in $Q$, and $F_{(k)}$ is a $n \times k$ matrix. Since $(I - \alpha S)^{-1} = Q(I - \alpha \Lambda)^{-1} Q^T \approx Q_{(k)} (I - \alpha \Lambda)^{-1}_{(k)} Q_{(k)}^T$, Eqn. (3.4) can be approximated by
\[
\|X - (1 - \alpha)(I - \alpha S)^{-1} X\|^2 \approx \|X - F_{(k)} F_{(k)^T} X\|^2,
\]
which is similar to Eqn. (3.3). Thus, although it is not feasible to directly solve the non-convex Eqn. (3.4) for $S$, $F$ learned by Eqn. (3.3) can be considered as a low-rank nonnegative factorization of $(1 - \alpha)(I - \alpha S)^{-1}$ in Eqn. (3.4).

Mathematically, $FF^T$ keeps most of the properties of $(1 - \alpha)(I - \alpha S)^{-1}$. Specifically, both matrices contain only nonnegative entries and both are positive semi-definite and symmetric. In general it is not guaranteed that a learned $F$ can be used to derive a valid $S$ as a solution for Eqn. (3.4). The following derivations show the connection between $S$ and $F$:

$$S \approx Q_{(k)} \Lambda_{(k)} Q_{(k)}^T = \frac{F(I - \alpha \Lambda)^{1/2}_{(k)} \Lambda_{(k)} (I - \alpha \Lambda)^{1/2}_{(k)} F^T}{\sqrt{1 - \alpha}} = \frac{F(I - \alpha \Lambda)_{(k)}^T}{1 - \alpha} F^T.$$

Clearly $F$ can be normalized by the unknown variable $\Lambda_{(k)}$, which are the $k$ largest eigenvalues of $S$, to derive an approximated $S$. A strategy to estimate $\Lambda_{(k)}$ is to derive the top $k$ eigenvalues of $FF^T$ by eigen decomposition. However, this requires a good estimation that satisfies two conditions that $\Lambda_{(k)}$ is between $[-1, 1]$ and $S$ is nonnegative, which is not always feasible.

Note that, to make the problem more general, we remove the requirement of orthogonal columns in $F$ in Eqn. (3.3). After $F$ is learned, we can compute a new similarity matrix $W = FF^T$. Being symmetric and positive, $W$ can be directly used as the similarity matrix for label propagation.

3.2 Multiplicative Update Rule. To solve the optimization problem in Eqn. (3.3), we propose an algorithm similar to nonnegative matrix factorization [7]. We first calculate the gradient of $F$ by
\[
\frac{\partial Q}{\partial F} = -2(X - FF^T X)X^T F - 2X(X^T - X^T F F^T) F.
\]
The additive update rule from the gradient descent algorithm is
\[
F_{ij} \leftarrow F_{ij} + \eta_{ij} [(X - FF^T X)X^T F + X(X^T - X^T F F^T) F]_{ij}
\]
where $\eta_{ij} > 0$ is the learning rate. If we further assume that $X$ contains only nonnegative values, the same technique used in [7] and [13] can be used to learn a nonnegative $F$ by the following multiplicative update rule:
\[
(3.6) \quad F_{ij} \leftarrow F_{ij} \frac{X_{ij}}{\sqrt{(FF^T F)^{ij} X_{ij} + X_{ij} F F^T F_{ij}}},
\]
where $\times$ represents element-wise multiplication. The algorithm randomly initializes $F$ to be a nonnegative matrix and then iteratively updates $F$ with Eqn. (3.6) until convergence. Empirically, each update reduces the cost function in Eqn. (3.3) as shown in our experiments. Thus, the algorithm learns a local optimal solution. Let $l$ be the number of iterations needed for convergence. By calculating the matrix multiplication in a proper order, the time complexity of the algorithm is $\Theta(l(m + k)n^2)$, which is also $\Theta(lmnk)$ since $k \ll m$. Empirically the algorithm needs hundreds of iterations to converge. Assuming the number of iterations is bounded by a constant, the total time complexity is linear with the number of samples or features.

3.3 A Clustering View. An intuitive interpretation of $F$ by Eqn. (3.3) is that $F$ reveals $k$-way clustering of dataset $X$ and each $F_{ij}$ tells the membership of the $i$th data point to the $j$th cluster. If $F$ is the ideal binary membership indicator with $F_{ij} = 1$ if the $i$th data point belongs to the $j$th cluster and otherwise $F_{ij} = 0$, the columns in $F$ are naturally orthogonal. Thus, $F$ is not required to contain orthogonal columns in (3.3) since the cluster components in the optimal $F$ are often almost orthogonal (as shown in our experiments). Since $F$ can contain any nonnegative value instead of just binary values, it can be explained as soft cluster memberships as in EM clustering. It is also natural to define the pairwise similarities as $W = FF^T$ since $W_{ij} = \sum_{l=1}^{k} F_{il} F_{lj}$ implies how likely the $i$th and $j$th data points are in the same cluster.

Our algorithm is different from spectral clustering [13] or Laplacian embedding [12] because both the approaches require a predefined similarity measure to start with. Since we still assume each data point can be reconstructed linearly from its neighborhood and we explicitly minimize the reconstruction error $\|X - FF^T X\|^2$, $F$ is more appropriate for label propagation than the membership matrix learned by clustering techniques.

3.4 Outline of GLNP Algorithm. The complete Global Linear Neighborhood Propagation (GLNP) algorithm is described in Alg. 1. GLNP first learns the $F$ matrix from data $X$ using the multiplicative update rule
\begin{algorithm}

\textbf{Algorithm 1 GLNP}

\textbf{INPUT:} $\mathcal{X} = \{x_1, \ldots, x_l, \ldots, x_n\}$ is a set of $n$ data points in $\mathbb{R}^m$; $\{x_1, \ldots, x_l\}$ are labeled by $\{y_1, \ldots, y_l\}$ and $\{x_{l+1}, \ldots, x_n\}$ are unlabeled; rank $k$ and balancing parameter $\alpha \in (0, 1)$.

\textbf{OUTPUT:} The labels of the data points $\{x_{l+1}, \ldots, x_n\}$.

1. Construct a $n \times m$ matrix $X$ from $\mathcal{X}$ where $X_{ij}$ is the value of the data point $x_i$ at the $j$th dimension and rescale $X$ if necessary. Construct a $n \times c$ matrix $Y$ where $Y_{ij} = 1$ if $1 \leq i \leq l$ and $y_i = j$, and 0 otherwise.

2. Randomly initialize a nonnegative $n \times k$ matrix $F$.

\hspace{6mm} Use the multiplicative update rule

\begin{equation}
F_{ij}^{t+1} \leftarrow F_{ij}^t \sqrt{\frac{(2XX^TF)^t_{ij}}{(F^tX^TXF^t + X^TXF^tF^tF)^t_{ij}}} \end{equation}

\hspace{6mm} to update $F$ until convergence.

3. Construct a $n \times n$ matrix $S = D^{-1/2}F^TFD^{-1/2}$ where $D$ is a diagonal matrix with its $(i, i)$-element equal to the sum of the $i$th row of $F^TF$.

4. Initialize a $n \times c$ matrix $Z = Y$, and iterate until convergence

\begin{equation}
Z^{t+1} = \alpha SZ^t + (1 - \alpha)Y.
\end{equation}

5. Output the labels of each data point $x_i$ by $y_i = \arg\max_j Z_{ij}$ for $i = l+1, \ldots, n$.

\end{algorithm}

in Eqn. (3.6). A normalized $S$ matrix is then computed from $F$. Finally, $S$ is used to run label propagation for semi-supervised learning.

3.5 Efficient Inductive Learning. Although GLNP is mainly proposed for transductive learning or semi-supervised learning, it can also be extended for efficient inductive learning. Given the learned optimal $n \times k$ matrix $F$ for $n$ samples $\{x_1, \ldots, x_n\}$ and the corresponding $Z$ matrix obtained after label propagation, let $f_i$ be a column vector obtained by the transpose of the $i$th row of $F$ and $z_i$ be the $i$th row of $Z$. For a new test point $u$, we first learn a $k \times 1$ column vector $f_u$ from the following optimization problem:

\begin{equation}
\min_{f_u} \left\| x_u^T - \sum_{i=1}^{n} (f_u^T f_i) x_i^T \right\|^2
\end{equation}

subject to $f_u \geq 0$. We can further derive Eqn. (3.7) as follows:

\begin{align*}
\left\| x_u^T - \sum_{i=1}^{n} (f_u^T f_i) x_i^T \right\|^2 &= \sum_{j=1}^{m} \left\| x_u - \sum_{i=1}^{n} (f_u^T f_i) x_{ij} \right\|^2 \\
&= \sum_{j=1}^{m} \left\| x_u - \sum_{i=1}^{n} f_u^T f_i x_{ij} \right\|^2 \\
&= \sum_{j=1}^{m} \left\| x_u - \sum_{i=1}^{n} f_u^T f_i \right\|^2 \\
&= \left\| Cf_u - x_u \right\|^2,
\end{align*}

where $C$ is a $m \times k$ matrix and the $j$th row $(j = 1, \ldots, m)$ of $C$ equals to $\sum_{i=1}^{n} (f_i^T X_{ij})$. $C$ can be interpreted as the similarity matrix between the $m$ features and the $k$ cluster components. Eqn. (3.7) seeks the best $f_u$—a $k$ dimensional embedding of $x_u$ such that $Cf_u$ can reconstruct $x_u$. Eqn. (3.7) is a standard nonnegative least-squares problem with $k$ variables. Since $k$ is usually very small, the problem can be solved efficiently. After $f_u$ is learned, by the inductive form of label propagation described in [4], predictions against each class can be calculated by

\begin{equation}
z_u = \sum_{i=1}^{n} f_u^T f_i z_i.
\end{equation}

The predicted label of $u$ is then $y_u = \arg\max_j z_{uj}$. In the inductive learning setting, we only need to precompute and save the $m \times k$ matrix $C$ instead of the $n \times m$ data matrix $X$. Given $k \ll n$ in most cases, this is a significant saving in space, while other label propagation algorithms such as [23] and [19] have to save the full data matrix $X$ for inductive learning.

4 Experiments and Analysis

We experimented with simulations, four UCI datasets and three high-dimensional microarray gene expression datasets.

4.1 Simulations. To compare global linear neighborhoods with local linear neighborhoods, we generated two toy datasets shown in Fig. [2]. In both datasets, data points in two classes (red squares vs. blue triangles) were generated from mixtures of different numbers of Gaussian distributions. 30 data points were sampled from each class. Both datasets are visualized with the mean and variance plotted by ellipses for each Gaussian distribution (Fig. [2](a) & (d)).

One data point from each cluster was randomly selected as the training set, denoted by solid squares or
trianes. For LNP, we set $\alpha = 0.99$ in label propagation since it generated the best results on the two simulated datasets. For GLNP, the results were similar as long as $\alpha$ was not too large. Thus, we set $\alpha = 0.1$. For both LNP and GLNP, we set $k = 3$ for the first dataset and $k = 4$ for the second dataset to match the number of Gaussian clusters. Euclidean distance was used to select the $k$-nearest neighbors for LNP.

The classification results by LNP and GLNP are shown in Fig. 2(b)&(e) and Fig. 2(c)&(f). LNP was unable to classify some data points after label propagation, denoted by black asterisks. In these examples, the graphs defined by the adjacency matrix $W$ learned by LNP are separated into several small disjoint components or not strongly connected. If there is no training data in a component, the data points in the component will remain unlabeled after label propagation. The graphs constructed by LNP for both datasets are visualized in Fig. 3(a)&(b). Although LNP picked the $k$-nearest neighbors to learn $W$, some data points are connected to less than $k$ neighbors due to the nature of the optimization problem defined in Eqn. (2.1). Even if some data points seem to be connected in Fig. 3(a), those links cannot be used for label propagation because the data points are not strongly connected in the graph. In both datasets, LNP failed to make predictions for around one third to a half of the data points (Fig. 2(b)&(e)). By increasing $k$, LNP could make perfect classification on the first simulated dataset. However, in the second dataset, LNP was never able to cor-
rectly classify around one fourth of the data points regardless of the choice of $k$. GLNP classified all data points perfectly in both datasets (Fig. 2(c)&(f)). The cluster components $F$ learned by GLNP are shown in Fig. 3(c)&(d). The internal cluster structures of the data were learned correctly by the components distinguishing the Gaussian clusters in the two datasets.

4.2 Experiments on UCI Datasets and Gene Expression Datasets. In the experiments on the seven real datasets, we compared GLNP with Linear Neighborhood Propagation (LNP) \cite{19}. Sparsity Induced Similarity measure (SIS) \cite{6} and the original label propagation with Gaussian Kernel (LP). As it was shown in \cite{19}, since the performance of LP is very sensitive to the variance parameter $\sigma^2$ and highly data-dependent, there is no good strategy to choose $\sigma^2$. Thus, in all experiments we report the results with both the best $\sigma$ and $\sigma^2 = \text{mean}\left\{ \frac{\|x_i - x_j\|^2}{2} \right\}$. The parameter $\alpha$ for label propagation was tuned to achieve optimal performance for all methods. We also tested different $k$ parameters for a comprehensive comparison of GLNP and LNP.

Four UCI datasets were tested. The first dataset is the Iris flower dataset. It contains 50 samples from each of three species of Iris flowers (setosa, virginica and versicolor) with four features. The second dataset is the Yale Face Database B from \cite{9}. We used the processed data from \cite{5}, which includes images of individuals 2, 5 and 10, and down-sampled each image to 30×40 pixels. The processed data contains 1755 images with 1200 features. The third dataset is Glass Identification data (Glass). It has 9 continuous numerical features describing each of 214 instances in two classes: Window vs non-Window glasses. The fourth dataset is the Wine dataset, which contains 178 samples from three types of wines.

We also tested three high-dimensional microarray gene expression datasets. The first dataset is the breast cancer gene expression profiles from \cite{18}. It contains the expression profiles of 24481 genes for 97 patient samples, among whom 51 patients had good prognosis and 46 patients had poor prognosis. The second dataset is the lung cancer gene expression data from \cite{10}. It contains the expression values of 22283 genes from 58 tumor and 49 non-tumor tissues (from 20 never smokers, 26 former smokers, and 28 current smokers). The third dataset is another lung cancer gene expression data from \cite{16}. It contains the expression profiles of 22283 genes for paired adjacent normal-tumor samples from 27 patients.
underwent surgery for lung cancer, which add up to 54 samples for classification.

We randomly selected 1% samples as the training set for Iris flower dataset and Yale Face dataset and run random trials of classification multiple times to report the average accuracy for each method. Since the other five datasets are harder to classify, 10% samples were selected as the training set in each trial. On each dataset, we run 50 trials with different randomly selected training sets.

The classification results on the UCI datasets and the gene expression datasets are reported in Fig. 4 and Fig. 5, respectively. GLNP performed consistently better than LNP with almost any choice of $k$ in the experiments on all the datasets. The lower variance of GLNP compared to other methods suggest that the performance of GLNP is more stable. Another observation is that the performance of GLNP is not very sensitive to parameter $k$ as long as $k$ is not too small. As GLNP interprets $k$ as the number of clusters in the data, it is expected that when $k$ is smaller than the number of classes, the performance of GLNP might not be as competitive in several cases. LNP performed reasonably well on relatively noise-free datasets because the label of a sample can be reliably predicted by just checking its local neighbors. However, when the noise in the data leads to wrong neighborhood relations as shown in the simulations, the global information is necessary to achieve better performance. This explanation is clearly illustrated by the results on the three gene expression datasets, where the performance of GLNP is more significantly better than LNP. SIS performed best on Wine dataset. However, SIS could not scale to Yale Faces and other gene expression datasets due to the low scalability of its optimization. In addition, it is infeasible to run SIS on datasets with a larger number of features than samples. Although LP performed well with the optimal $\sigma$, it is not possible to estimate the optimal $\sigma$ from data. The default $\sigma$ generated much worse results compared to the optimal $\sigma$.

To analyze the cluster components $F$ learned by GLNP, we plot the components learned from the Iris, Yale Faces and Glass datasets in Fig. 6. The three datasets were selected since they are less noisy and contain clearer cluster structures. $k$ was chosen to be the same as the number of classes in the datasets. It is clear that each component represents one class in the three datasets. Fig. 6(a) shows the convergence property of GLNP. Since the convergence rates are...
similar in the datasets under different choices of $k$, we plot the cost function in each iteration with $k$ equal to the number of classes on the seven datasets. In all cases, GLNP converged well within less than 500 iterations, and the cost function monotonically decreases after each iteration. We also tested subsets of Yale Face dataset with different numbers of samples or features ranging from 100 to 1000 to report the average run time of GLNP in Fig. 7(b). The result shows that the run time of GLNP is approximately linear with the number of samples or features.

5 Discussions

Compared with local linear neighborhoods, global linear neighborhoods preserve the global structures among the data points, and thus, constitute more robust and reliable similarity graphs for label propagation. GLNP algorithm adopts a multiplicative update rule to find a nonnegative factorization matrix of the global similarity matrix. The strategy is similar to nonnegative matrix factorization and thus, scales reasonably well on large datasets. Empirically, GLNP converges fast in all the experiments reported in this study (Fig. 7(a)). GLNP was also able to scale up on datasets with more than 10,000 samples (results not shown). Thus, GLNP possesses better scalability compared with SIS.

Neighborhood learning for label propagation is different from general metric learning [20, 8] or covariance matrix learning by General Gaussian Models. For example, [12] proposed a semi-supervised sparse metric learning algorithm by first performing affinity propagation and then optimizing the sparse metric by alternating linearization. The sparsity in their paper is defined on the metric instead of the affinity matrix. However, the purpose of sparse graph representation learning is to learn the best sparse or compressed graph that preserves the graph structure for effective label propagation. Thus, the new graph needs to be non-negative, preferably highly connected and sparse or low-rank. These properties might not be easily achieved by extending Gaussian models because the learned covariance matrix cannot contain diagonal blocks and needs to be sparse or low-rank. GLNP is also fundamentally different from the graph compression approaches based on a preprocessing with clustering. For example, the authors in [11] proposed to first find $k$ cluster centers and then construct a local linear embedding of data points by closest cluster centers. In philosophy, GLNP is global embedding and the approach in [11] is still a “local” embedding but by cluster centers. The sparsity is obtained by the low-rank embedding for GLNP but by the nearest anchor points for the approach in [11]. Although the clustering step was introduced to compress the data, it is in general a difficult and unreliable preprocessing step, which requires another similarity measure between the data points.

Although the multiplicative update rule assumes that the data is nonnegative, GLNP can also handle data with both positive and negative values by using the additive update rule with nonnegative constraints. Furthermore, by rescaling the data to be in a positive interval, the faster multiplicative update rule can be applied and empirically, GLNP also produced significantly better results on the rescaled datasets. Finally, it is not always possible to find the corresponding similarity matrix ($S$) used to compute the global linear neighborhoods, with the cluster components $F$. However, it is less a concern since our focus is on finding a sparse global similarity matrix for label propagation to improve classification performance.
6 Acknowledgement

This work was supported in part by award III-1117153 from the National Science Foundation. We sincerely thank Eric P. Xing for his constructive comments and helpful suggestions on the experiment design and paper drafting.

References