Large Graph Construction for Scalable Semi-Supervised Learning

Wei Liu WLIU@EE.COLUMBIA.EDU
Junfeng He JH2700@COLUMBIA.EDU
Shih-Fu Chang SFCHANG@EE.COLUMBIA.EDU
Department of Electrical Engineering, Columbia University, New York, NY 10027, USA

Abstract

In this paper, we address the scalability issue plaguing graph-based semi-supervised learning via a small number of anchor points which adequately cover the entire point cloud. Critically, these anchor points enable nonparametric regression that predicts the label for each data point as a locally weighted average of the labels on anchor points. Because conventional graph construction is inefficient in large scale, we propose to construct a tractable large graph by coupling anchor-based label prediction and adjacency matrix design. Contrary to the Nyström approximation of adjacency matrices which results in indefinite graph Laplacians and in turn leads to potential non-convex optimization over graphs, the proposed graph construction approach based on a unique idea called AnchorGraph provides nonnegative adjacency matrices to guarantee positive semidefinite graph Laplacians. Our approach scales linearly with the data size and in practice usually produces a large sparse graph. Experiments on large datasets demonstrate the significant accuracy improvement and scalability of the proposed approach.

1. Introduction

In pervasive applications of machine learning, one frequently encounters situations where only a few labeled data are available and large amounts of data remain unlabeled. The labeled data often suffer from difficult and expensive acquisition whereas the unlabeled data can be cheaply and automatically gathered. Semi-supervised learning (SSL) (Chapelle et al., 2006)(Zhu, 2008) has been recommended to cope with the very situations of limited labeled data and abundant unlabeled data.

With rapid development of the Internet, now we can collect massive (up to hundreds of millions) unlabeled data such as images and videos, and then the need for large scale SSL arises. Unfortunately, most SSL methods scale badly with the data size n. For instance, the classical TSVM (Joachims, 1999) is computationally challenging, scaling exponentially with n. Among various versions of TSVM, CCCP-TSVM (Collobert et al., 2006) has the lowest complexity, but it scales as at least $O(n^2)$ so it is still difficult to scale up.

Graph-based SSL (Zhu et al., 2003)(Zhou et al., 2004) (Belkin et al., 2006) is appealing recently because it is easy to implement and gives rise to closed-form solutions. However, graph-based SSL usually has a cubic time complexity $O(n^3)$ since the inverse of the $n \times n$ graph Laplacian is needed[1], thus blocking widespread applicability to real-life problems that encounter growing amounts of unlabeled data. To temper the cubic time complexity, recent studies seek to reduce the intensive computation upon the graph Laplacian manipulation. (Delalleu et al., 2005) proposed a nonparametric inductive function which makes label prediction based on a subset of samples and then truncates the graph Laplacian with the selected subset and its connection to the rest samples. Clearly, such a truncation ignores the topology structure within the majority part of input data and thereby loses considerable data information. (Zhu & Lafferty, 2005) fitted a generative mixture model to the raw data and proposed the harmonic mixtures to span the label prediction function, but it did not explain how to construct a large sparse graph such that the proposed harmonic mixtures method can be scalable. (Tsang & Kwok, 2007) scaled up the manifold regularization technology first proposed in (Belkin et al., 2006) through solving

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

[1] It is not easy yet to exactly solve the equivalent large-scale linear systems.

the dual optimization problem of manifold regularization subject to a sparsity constraint. (Karlen et al., 2008) trained large scale TSVMs by means of stochastic gradient descent and a multi-layer architecture. (Zhang et al., 2009) applied the Nyström approximation to the huge graph adjacency (or affinity) matrix, but there is no guarantee for the graph Laplacian matrix computed from the Nyström-approximated adjacency matrix to be positive semidefinite, which leads to non-convex optimization. (Fergus et al., 2010) specified the label prediction function using smooth eigenvectors of the graph Laplacian which are calculated by a numerical method. However, this method relies on the dimension-separable data density assumption which is not always true.

In this paper, we propose a large graph construction approach to efficiently exploit all data points. This approach is simple and scalable, enjoying linear space and time complexities with respect to the data size.

2. Overview

We try to address the scalability issue pertaining to SSL from two perspectives: anchor-based label prediction and adjacency matrix design.

2.1. Anchor-Based Label Prediction

Our key observation is that the computational intensiveness of graph-based SSL stems from the full-size label prediction models. Since the number of unlabeled samples is huge in large scale applications, learning full-size prediction models is inefficient.

Suppose a soft label prediction function $f : \mathbb{R}^d \mapsto \mathbb{R}$ defined on the input samples $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n$. Without loss of generality, we assume that the first l samples are labeled and the rest remain unlabeled. To work under large scale, (Delalleu et al., 2005)(Zhu & Lafferty, 2005) made the label prediction function be a weighted average of the labels on a subset of anchor (landmark) samples. As such, if one can infer the labels associated with the much smaller subset, the labels of other unlabeled samples will be easily obtained by a simple linear combination.

The idea is to use a subset $\mathcal{U} = \{\boldsymbol{u}_k\}_{k=1}^m \subset \mathbb{R}^d$ in which each $\boldsymbol{u}_k$ acts as an *anchor* point since we represent f in terms of these points, i.e.,

$$f(\boldsymbol{x}_i) = \sum_{k=1}^{m} Z_{ik} f(\boldsymbol{u}_k), \tag{1}$$

where Z_{ik}'s are sample-adaptive weights. Such a label prediction essentially falls into nonparametric regression (Hastie et al., 2009). Let us define two vectors $\boldsymbol{f} = [f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_n)]^\top$ and $\boldsymbol{a} = [f(\boldsymbol{u}_1), \cdots, f(\boldsymbol{u}_m)]^\top$, and rewrite eq. (1) as

$$\boldsymbol{f} = Z\boldsymbol{a},\ Z \in \mathbb{R}^{n\times m},\ m \ll n. \tag{2}$$

This formula serves as a main disposal of scalable SSL because it reduces the solution space of unknown labels from large $\boldsymbol{f}$ to much smaller $\boldsymbol{a}$. This economical label prediction model eq. (2) surely mitigates the computational burden of the original full-size models.

Importantly, we take these anchor points $\{\boldsymbol{u}_k\}$ as `k-means` clustering centers instead of randomly sampled exemplars because it turns out that `k-means` clustering centers have a stronger representation power to adequately cover the vast point cloud $\mathcal{X}$.

2.2. Adjacency Matrix Design

Recall that in the literature an undirected weighted graph $G(V, E, W)$ is built on n data points. V is a set of nodes with each v_i representing a data point $\boldsymbol{x}_i$, $E \subseteq V \times V$ is a set of edges connecting adjacent nodes, and $W \in \mathbb{R}^{n\times n}$ is a weighted adjacency matrix which measures the strength of edges. Obviously, edge connections in graphs are crucial to the outcome. One broadly used connecting strategy is the kNN graph which creates an edge between v_i and v_j if $\boldsymbol{x}_i$ is among k nearest neighbors of $\boldsymbol{x}_j$ or vice versa. The time cost of kNN graph construction is $O(kn^2)$, so even this conventional graph construction approach is infeasible in large scale. Although we may employ approximate kNN graph construction to save the time cost, the large matrix inversion or large-scale linear system solving involved in manipulating large graphs remains a big hurdle.

On the other hand, it is unrealistic to save in memory a matrix W as large as $n \times n$. Hence, designing memory and computationally tractable W constitutes a major bottleneck of large scale graph-based SSL. We should find an approach to parsimoniously represent W for large graphs.

2.3. Design Principles

Now we investigate some principles for designing Z and W tailored to large scale problems.

Principle (1) We impose the nonnegative normalization constraints $\sum_{k=1}^m Z_{ik} = 1$ and $Z_{ik} \geq 0$ to maintain the unified range of values for all predicted soft labels via regression. The *manifold assumption* implies that contiguous data points should have similar labels and distant data points are very unlikely to take similar labels. This motivates us to also impose $Z_{ik} = 0$

when anchor $\boldsymbol{u}_k$ is far away from $\boldsymbol{x}_i$ so that the regression on $\boldsymbol{x}_i$ is a locally weighted average in spirit. As a result, $Z \in \mathbb{R}^{n\times m}$ is nonnegative as well as sparse.

Principle (2) We require $W \geq 0$. The nonnegative adjacency matrix is sufficient to make the resulting graph Laplacian $L = D - W$ ($D \in \mathbb{R}^{n\times n}$ is a diagonal matrix with entries being $D_{ii} = \sum_{j=1}^{n} W_{ij}$) positive semidefinite (Chung, 1997). This nonnegative property is important to guarantee global optimum of many graph-based SSL methods.

Principle (3) We prefer sparse W because sparse graphs have much less spurious connections between dissimilar points and tend to exhibit high quality. (Zhu, 2008) has pointed out that fully-connected dense graphs perform worse than sparse graphs empirically.

Intuitively, we would use the nonnegative sparse matrix Z to design the nonnegative sparse matrix W. Actually, in the next section, we are able to design Z and W jointly and generate empirically sparse large graphs. On the contrary, the recently proposed Prototype Vector Machine (PVM) (Zhang et al., 2009) designed Z and W separately, producing improper dense graphs. In addition, when using the Nyström method to approximate a predefined W like a kernel matrix, PVM fails to preserve the nonnegative property of graph adjacency matrices. Therefore, PVM cannot guarantee that the graph Laplacian regularization term appearing in its cost functions is convex, so PVM suffers heavily from local minima. Crucially, we are not trying to approximate any predefined W; instead, we design it directly to cater for the nonnegative and sparse properties.

3. AnchorGraph: Large Graph Construction

3.1. Design of Z

We aim at designing a regression matrix Z that measures the underlying relationship between raw samples $\mathcal{X}$ and anchors $\mathcal{U}$ (note that $\mathcal{U}$ is outside $\mathcal{X}$). Following *Principle* (1) in the last section, we desire to keep nonzero Z_{ik} for s ($< m$) closest anchors to $\boldsymbol{x}_i$. Exactly, the Nadaraya-Watson kernel regression (Hastie et al., 2009) defines such Z_{ik} based on a kernel function $K_h()$ with a bandwidth h:

$$Z_{ik} = \frac{K_h(\boldsymbol{x}_i, \boldsymbol{u}_k)}{\sum_{k'\in\langle i\rangle} K_h(\boldsymbol{x}_i, \boldsymbol{u}_{k'})} \ \forall k \in \langle i\rangle, \tag{3}$$

where the notation $\langle i\rangle \subset [1:m]$ is the set saving the indexes of s nearest anchors of $\boldsymbol{x}_i$. Typically, we may adopt the Gaussian kernel $K_h(\boldsymbol{x}_i, \boldsymbol{u}_k) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{u}_k\|^2/2h^2)$ for the kernel regression.

With the consideration that the kernel-defined weights are sensitive to the hyperparameter h and lack a meaningful interpretation, we obtain them from another perspective: geometric reconstruction similar to LLE (Roweis & Saul, 2000). Concretely, we reconstruct any data point $\boldsymbol{x}_i$ as a convex combination of its closest anchors, and the combination coefficients are preserved for the weights in nonparametric regression. Let us define a matrix $U = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m]$ and denote by $U_{\langle i\rangle} \in \mathbb{R}^{d\times s}$ a sub-matrix composed of s nearest anchors of $\boldsymbol{x}_i$. Then, we propose *Local Anchor Embedding* (LAE) to optimize the convex combination coefficients:

$$\begin{aligned} \min_{\boldsymbol{z}_i\in\mathbb{R}^s} \quad & g(\boldsymbol{z}_i) = \frac{1}{2}\|\boldsymbol{x}_i - U_{\langle i\rangle}\boldsymbol{z}_i\|^2 \\ \text{s.t.} \quad & \mathbf{1}^\top \boldsymbol{z}_i = 1, \quad \boldsymbol{z}_i \geq 0 \end{aligned} \tag{4}$$

where s entries of the vector $\boldsymbol{z}_i$ correspond to s combination coefficients contributed by s closest anchors. Beyond LLE, LAE imposes the nonnegative constraint and then the convex solution set to eq. (4) constitutes a *multinomial simplex*

$$\mathbb{S} = \left\{\boldsymbol{z} \in \mathbb{R}^s : \mathbf{1}^\top \boldsymbol{z} = 1, \boldsymbol{z} \geq 0\right\}. \tag{5}$$

In contrast to the regression weights as predefined in eq. (3), LAE is more advantageous because it provides optimized regression weights that are also sparser than the predefined ones.

Standard quadratic programming (QP) solvers can be used to solve eq. (4) but most of them need to compute some approximation of the Hessian and are thus relatively expensive. We apply the projected gradient method, a first-order optimization procedure, to solve eq. (4) instead. The updating rule in the projected gradient method is expressed as the following iterative formula

$$\boldsymbol{z}_i^{(t+1)} = \Pi_{\mathbb{S}}(\boldsymbol{z}_i^{(t)} - \eta_t \nabla g(\boldsymbol{z}_i^{(t)})), \tag{6}$$

where t denotes the time stamp, $\eta_t > 0$ denotes the appropriate step size, $\nabla g(\boldsymbol{z})$ denotes the gradient of g at $\boldsymbol{z}$, and $\Pi_{\mathbb{S}}(\boldsymbol{z})$ denotes the simplex projection operator on any $\boldsymbol{z} \in \mathbb{R}^s$. Mathematically, the projection operator is formulated as

$$\Pi_{\mathbb{S}}(\boldsymbol{z}) = \arg\min_{\boldsymbol{z}'\in\mathbb{S}} \|\boldsymbol{z}' - \boldsymbol{z}\|. \tag{7}$$

Such a projection operator has been implemented efficiently in $O(s \log s)$ (Duchi et al., 2008), which is described in Algorithm 1.

To achieve faster optimization, we employ *Nesterov's method* (Nesterov, 2003) to accelerate the gradient decent in eq. (6). As a brilliant achievement in the optimization field, Nesterov's method has a much faster

Algorithm 1 Simplex Projection

Input: A vector $\boldsymbol{z} \in \mathbb{R}^s$.
sort $\boldsymbol{z}$ into $\boldsymbol{v}$ such that $v_1 \geq v_2 \geq \cdots \geq v_s$
find $\rho = \max\{j \in [1:s] : v_j - \frac{1}{j}(\sum_{r=1}^{j} v_r - 1) > 0\}$
compute $\theta = \frac{1}{\rho}(\sum_{j=1}^{\rho} v_j - 1)$
Output: A vector $\boldsymbol{z}' = [z'_1, \cdots, z'_s]^\top$ such that $z'_j = \max\{z_j - \theta, 0\}$.

Algorithm 2 Local Anchor Embedding (LAE)

Input: data points $\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, anchor point matrix $U \in \mathbb{R}^{d\times m}$, integer s.
for $i = 1$ **to** n **do**
  for $\boldsymbol{x}_i$ find s nearest neighbors in U, saving the index set $\langle i\rangle$;
  define functions $g(\boldsymbol{z}) = \|\boldsymbol{x}_i - U_{\langle i\rangle}\boldsymbol{z}\|^2/2$, $\nabla g(\boldsymbol{z}) = U_{\langle i\rangle}^\top U_{\langle i\rangle}\boldsymbol{z} - U_{\langle i\rangle}^\top \boldsymbol{x}_i$, and $\tilde{g}_{\beta,\boldsymbol{v}}(\boldsymbol{z}) = g(\boldsymbol{v}) + \nabla g(\boldsymbol{v})^\top(\boldsymbol{z} - \boldsymbol{v}) + \beta\|\boldsymbol{z} - \boldsymbol{v}\|^2/2$;
  initialize $\boldsymbol{z}^{(1)} = \boldsymbol{z}^{(0)} = \mathbf{1}/s$, $\delta_{-1} = 0$, $\delta_0 = 1$, $\beta_0 = 1$, $t = 0$;
  repeat
    $t = t + 1$, $\alpha_t = \frac{\delta_{t-2}-1}{\delta_{t-1}}$
    set $\boldsymbol{v}^{(t)} = \boldsymbol{z}^{(t)} + \alpha_t(\boldsymbol{z}^{(t)} - \boldsymbol{z}^{(t-1)})$
    for $j = 0, 1, \cdots$ **do**
      $\beta = 2^j\beta_{t-1}$, $\boldsymbol{z} = \Pi_{\mathbb{S}}(\boldsymbol{v}^{(t)} - \frac{1}{\beta}\nabla g(\boldsymbol{v}^{(t)}))$
      if $g(\boldsymbol{z}) \leq \tilde{g}_{\beta,\boldsymbol{v}^{(t)}}(\boldsymbol{z})$ **then**
        update $\beta_t = \beta$ and $\boldsymbol{z}^{(t+1)} = \boldsymbol{z}$
        break
      end if
    end for
    update $\delta_t = \frac{1+\sqrt{1+4\delta_{t-1}^2}}{2}$
  until $\boldsymbol{z}^{(t)}$ converges;
  $\boldsymbol{z}_i = \boldsymbol{z}^{(t)}$.
end for
Output: LAE vectors $\{\boldsymbol{z}_i\}_{i=1}^n$.

convergence rate than the traditional methods such as gradient descent and subgradient descent. We describe LAE accelerated by Nesterov's method in Algorithm 2. After solving the optimal weight vector $\boldsymbol{z}_i$, we set

$$Z_{i,\langle i\rangle} = \boldsymbol{z}_i^\top,\ |\langle i\rangle| = s,\ \boldsymbol{z}_i \in \mathbb{R}^s \tag{8}$$

and $Z_{i,\overline{\langle i\rangle}} = 0$ for the rest entries of Z. To summarize, we optimize the weights used for anchor-based nonparametric regression by means of data reconstruction with contiguous anchors. For each data point, the LAE algorithm converges within a few iterations T in practice. Ultimately, LAE outputs a highly sparse Z (a memory space of $O(sn)$) with a total time complexity $O(smn + s^2Tn)$.

3.2. Design of W

So far, we have set up m anchors (cluster centers) to cover a point cloud of n data points, and also designed a nonnegative matrix Z that supports the economical label prediction model shown in eq. (2). Intuitively, we may design the adjacency matrix W using Z as follows

$$W = Z\Lambda^{-1}Z^\top, \tag{9}$$

in which the diagonal matrix $\Lambda \in \mathbb{R}^{m\times m}$ is defined as $\Lambda_{kk} = \sum_{i=1}^n Z_{ik}$. Immediately, such a defined adjacency matrix W satisfies *Principle* (2) since Z is nonnegative. Further, we find out that nonnegative sparse Z leads to empirically sparse W when the anchor points are set to cluster centers such that most data point pairs across different clusters do not share the same set of closest cluster centers. Accordingly, W satisfies *Principle* (3) in most cases[2].

We term the large graph G described by the adjacency matrix W in eq. (9) `AnchorGraph`. Eq. (9) is the core finding of this paper, which constructs a nonnegative and empirically sparse graph adjacency matrix W via crafty matrix factorization. Furthermore, it couples anchor-based label prediction and adjacency matrix design via the common matrix Z. Hence, we only need to save Z, linear in the data size n, in memory as it not only contributes to the final label prediction but also skillfully constructs the `AnchorGraph`. The resultant graph Laplacian of the `AnchorGraph` is derived by $L = D - W = I - Z\Lambda^{-1}Z^\top$ where the diagonal degree matrix D equals the identity matrix.

Theoretically, we can derive eq. (9) by a probabilistic means. As the presented LAE algorithm derives Z from a geometrical reconstruction view, this matrix Z actually unveils a tight affinity measure between data points and anchor points. That is sound in the sense that the more an anchor $\boldsymbol{u}_k$ contributes to the reconstruction of a data point $\boldsymbol{x}_i$, the larger the affinity between them. To explicitly capture the data-to-anchor relationship, we introduce a *bipartite graph* (Chung, 1997) $\mathcal{B}(V, \mathcal{U}, \mathcal{E})$. The new node set $\mathcal{U}$ includes nodes $\{u_k\}_{k=1}^m$ representing the anchor points and $\mathcal{E}$ contains edges connecting V and $\mathcal{U}$. We connect an undirected edge between v_i and u_k if and only if $Z_{ik} > 0$ and designate the edge weight as Z_{ik}. Then the cross adjacency matrix between $\{v_i\}_{i=1}^n$ and $\{u_k\}_{k=1}^m$ is Z and the full adjacency matrix for the bipartite graph $\mathcal{B}$ is thus $B = \begin{bmatrix} 0 & Z \\ Z^\top & 0 \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$ where $Z\mathbf{1} = \mathbf{1}$. A toy example for $\mathcal{B}$ is visualized in Fig. 1.

Over the bipartite graph $\mathcal{B}$, we establish stationary Markov random walks through defining the one-step transition probability matrix as $P = (D^B)^{-1}B$ in

[2] In an extreme case, if a *hub* anchor point exists such that a large number of data points are connected to it then W may be dense.

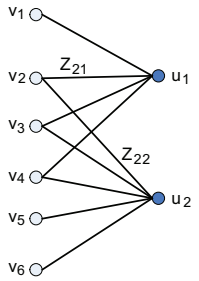

Figure 1. A bipartite graph representation of data points $v_1, \cdots, v_6$ and anchor points u_1, u_2. Z_{ik} captures the data-to-anchor relationship ($Z_{21} + Z_{22} = 1$).

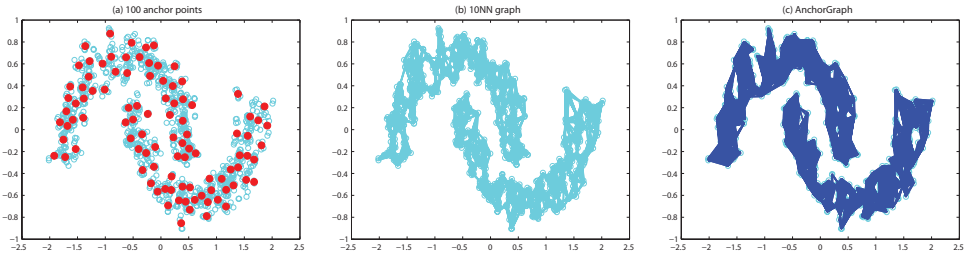

Figure 2. The two-moon problem of 1200 2D points. (a) 100 anchor points by `k-means` clustering ($m = 100$); (b) 10NN graph built on original points; (c) the proposed `AnchorGraph` with $s = 2$ built on original points.

which $D^B \in \mathbb{R}^{(n+m)\times(n+m)}$ is a diagonal matrix with entries being $D^B_{ii} = \sum_{j=1}^{n+m} B_{ij}$. By doing so, we obtain the transition probabilities in one time step as follows

$$p^{(1)}(u_k|v_i) = \frac{Z_{ik}}{\sum_{k'=1}^{m} Z_{ik'}} = Z_{ik},\ p^{(1)}(v_i|u_k) = \frac{Z_{ik}}{\sum_{j=1}^{n} Z_{jk}} \quad \forall i \in [1:n],\ \forall k \in [1:m]. \tag{10}$$

Obviously, $p^{(1)}(v_j|v_i) = 0$ and $p^{(1)}(u_r|u_k) = 0$ since there are no direct edges connecting them. Let us contemplate the two-step transition probabilities $p^{(2)}(v_j|v_i)$ and have the following proposition.

Proposition 1. *Given one-step transition probabilities defined in eq. (10), the transition probabilities in two time steps are*

$$p^{(2)}(v_j|v_i) = p^{(2)}(v_i|v_j) = \sum_{k=1}^{m} \frac{Z_{ik}Z_{jk}}{\Lambda_{kk}}. \tag{11}$$

Proof. We exploit the chain rule of Markov random walks to deduce

$$p^{(2)}(v_j|v_i) = \sum_{k=1}^{m} p^{(1)}(v_j|u_k)p^{(1)}(u_k|v_i) = \sum_{k=1}^{m} \frac{Z_{jk}}{\sum_{j'=1}^{n} Z_{j'k}} Z_{ik} = \sum_{k=1}^{m} \frac{Z_{ik}Z_{jk}}{\Lambda_{kk}}$$

which does not depend on the order between i and j, so we complete the proof. □

Proposition 1 indicates

$$W_{ij} = p^{(2)}(v_j|v_i) = p^{(2)}(v_i|v_j) \tag{12}$$

which interprets the designed adjacency matrix in a probabilistic measure and thereby testifies the correctness of our design. It is noticeable that we may also define graph adjacency matrices using the higher-order transition probabilities such as $W'_{ij} = p^{(4)}(v_j|v_i)$, but this leads to a denser adjacency matrix $W' = Z\Lambda^{-1}Z^\top Z\Lambda^{-1}Z^\top$ and increases the computational cost as well.

4. AnchorGraph Regularization

As the major contribution of this paper, the proposed `AnchorGraph` resembles the classical kNN graph in the connection structure. On the two-moon toy data, the `AnchorGraph`, which is really sparse and shown in Fig. 2(c), is close to the kNN graph shown in Fig. 2(b). Hence, we are able to establish a graph-regularized framework upon this `AnchorGraph` as it comprises all data and exhibits high fidelity to the kNN graph.

We turn our attention to a standard multi-class SSL setting where each labeled sample $\boldsymbol{x}_i$ ($i = 1 \cdots, l$) carries a discrete label $y_i \in \{1, \cdots, c\}$ from c distinct classes. We denote by $Y = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_c] \in \mathbb{R}^{l\times c}$ a class indicator matrix on labeled samples with $Y_{ij} = 1$ if $y_i = j$ and $Y_{ij} = 0$ otherwise. Amenable to the aforementioned anchor-based label prediction model, we only need to solve the soft labels associated with anchors which are put in the label matrix $A = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_c] \in \mathbb{R}^{m\times c}$ in which each column vector accounts for a class. We introduce the graph Laplacian regularization norm $\Omega_G(\boldsymbol{f}) = \frac{1}{2}\boldsymbol{f}^\top L\boldsymbol{f}$ that has been widely exploited in the recent papers. Tailored to each class, we have a label prediction function $\boldsymbol{f}_j = Z\boldsymbol{a}_j$. Then we formulate a SSL framework as follows

$$\min_{A=[\boldsymbol{a}_1,\cdots,\boldsymbol{a}_c]} \mathcal{Q}(A) = \frac{1}{2}\sum_{j=1}^{c} \|Z_l\boldsymbol{a}_j - \boldsymbol{y}_j\|^2 + \gamma \sum_{j=1}^{c} \Omega_G(Z\boldsymbol{a}_j) = \frac{1}{2}\|Z_lA - Y\|_{\mathrm{F}}^2 + \frac{\gamma}{2}\mathrm{tr}(A^\top Z^\top LZA),$$

Table 1. Time complexity analysis of the proposed scalable SSL approach. n is the data size, m is # of anchor points, s is # of nearest anchors in LAE, and T is # of iterations in LAE ($n \gg m \gg s$).

Approach	find anchors	design Z	graph regularization	total time complexity
`AnchorGraphReg`	$O(mn)$	$O(smn)$ or $O(smn + s^2Tn)$	$O(m^3 + m^2n)$	$O(m^2n)$

where $Z_l \in \mathbb{R}^{l\times m}$ is the sub-matrix according to the labeled partition, $\|.\|_{\mathrm{F}}$ stands for the Frobenius norm, and $\gamma > 0$ is the regularization parameter.

Meanwhile, we compute a "reduced" Laplacian matrix:

$$\begin{aligned}\tilde{L} &= Z^\top LZ = Z^\top(I - Z\Lambda^{-1}Z^\top)Z \\ &= Z^\top Z - (Z^\top Z)\Lambda^{-1}(Z^\top Z),\end{aligned}$$

which is both memory-wise and computationally tractable, taking $O(m^2)$ space and $O(m^3 + m^2n)$ time. Subsequently, we could simplify the cost function $\mathcal{Q}(A)$ to follows

$$\mathcal{Q}(A) = \frac{1}{2}\|Z_lA - Y\|_{\mathrm{F}}^2 + \frac{\gamma}{2}\mathrm{tr}(A^\top \tilde{L}A). \tag{13}$$

With simple algebra, we can easily obtain the globally optimal solution to eq. (13):

$$A^* = (Z_l^\top Z_l + \gamma\tilde{L})^{-1}Z_l^\top Y. \tag{14}$$

As such, we yield a closed-form solution for addressing large scale SSL. In the sequel we employ the solved soft labels associated with anchors to predict the hard label for any unlabeled sample as

$$\hat{y}_i = \arg\max_{j\in\{1,\cdots,c\}} \frac{Z_{i\cdot}\boldsymbol{a}_j}{\lambda_j}, \quad i = l+1,\cdots,n \tag{15}$$

where $Z_{i\cdot} \in \mathbb{R}^{1\times m}$ denotes the ith row of Z, and the normalization factor $\lambda_j = \mathbf{1}^\top Z\boldsymbol{a}_j$, suggested as a useful *class mass normalization* in the classical SSL paper (Zhu et al., 2003), balances skewed class distributions.

4.1. Complexity Analysis

The proposed `AnchorGraph` regularization, abbreviated to `AnchorGraphReg`, consists of three stages: 1) find anchors by `k-means` clustering, 2) design Z, and 3) run graph regularization. In each stage the space complexity is bounded by $O(m + n)$. In the second stage, we may use either predefined Z in eq. (3) or optimized Z offered by LAE. The time complexity for each stage is listed in Table 1. We have used a fixed number m ($\ll n$) of anchor points for large scale SSL, which is independent of the data size n. Therefore, our `AnchorGraphReg` approach scales linearly with the data size n.

5. Experiments

In this section, we evaluate the proposed scalable graph-based SSL approach `AnchorGraphReg` (AGR), which integrates anchor-based label prediction and adjacency matrix design, on three real-world datasets. We compare it with state-of-the-art SSL approaches and two recent scalable SSL approaches Eigenfunction (Fergus et al., 2010) and PVM (Zhang et al., 2009). We also report the performance of several baseline methods including 1NN, linear SVM and RBF SVM.

For fair comparisons, we designate the same clustering centers, that act as anchor points, for PVM with square loss, PVM with hinge loss, AGR with predefined Z (denoted by `AnchorGraphReg`^{0}), and AGR with LAE-optimized Z (denoted by `AnchorGraphReg`). For the two versions of AGR, we fix $s = 3$ to make constructed `AnchorGraph`s as sparse as possible. We use the same RBF kernel for SVM and PVM where the width of the RBF kernel is set by cross validation. All these compared methods are implemented in MATLAB 7.9 and run on a 2.53 GHz, 4GB RAM Core 2 Duo PC.

5.1. Mid-sized Dataset

To see if the proposed AGR can show good performance in mid-scale, we conduct experiments on the benchmark dataset **USPS** (the training part) in which each sample is a 16×16 digit image and there are ten types of digits 0, 1, 2, ..., 9 used as 10 classes, summing up to a total of 7,291. To make a SSL setting, we randomly choose $l = 100$ labeled samples such that they contain at least one sample from each class (note that this setting introduces the skewed class distribution in the labeled samples). We evaluate the baseline 1NN, two state-of-the-art SSL methods Local and Global Consistency (LGC) (Zhou et al., 2004) and Gaussian Fields and Harmonic Functions (GFHF) augmented by class mass normalization (Zhu et al., 2003), and four versions of AGR using randomly selected anchors and cluster center anchors.

Averaged over 20 trials, we calculate the classification error rates for the referred methods here. The results are displayed in Table 2 and Fig. 3. Table 2 lists a total running time including three stages `k-means` clustering, designing Z, and graph regularization for every version of AGR. The time cost of graph regulariza-

Table 2. Classification error rates (%) on **USPS-Train** (7,291 samples) with $l = 100$ labeled samples. $m = 1000$ for four versions of AGR. The running time of `k-means` clustering is 7.65 seconds.

Method	Error Rate (%)	Running Time (seconds)
1NN	20.15±1.80	0.12
LGC with 6NN graph	8.79±2.27	403.02
GFHF with 6NN graph	5.19±0.43	413.28
random `AnchorGraphReg`^{0}	11.15±0.77	2.55
random `AnchorGraphReg`	10.30±0.75	8.85
`AnchorGraphReg`^{0}	7.40±0.59	10.20
`AnchorGraphReg`	**6.56±0.55**	16.57

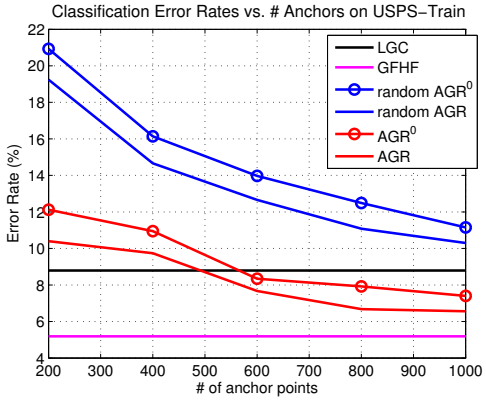

Figure 3. Average classification error rates vs. numbers of anchor points.

tion is quite small and can almost be ignored. From Table 2, we know that kNN graph construction and inverse of graph Laplacian in either LGC or GFHF are time-consuming, so LGC and GFHF are infeasible for larger datasets. It is pleasant to observe that the proposed AGR with $m = 1000$ cluster center anchors outperforms LGC and is comparable to GFHF, taking much shorter running time. Fig. 3 reveals that the cluster center anchors demonstrate a substantial advantage over the random anchors when using them for our approach AGR, and that the increasing anchor size m indeed leads to significant improvement of classification accuracy of AGR. In addition, AGR with LAE-optimized Z further improves the performance of AGR with predefined Z, so we can say that the geometrical strategy for designing Z makes sense.

5.2. Large Datasets

The **MNIST** dataset[3] contains handwritten digit images from '0' to '9'. It has a training set of 60,000 samples and a test set of 10,000 samples. We hold the training and test samples as a whole and randomly choose labeled samples from the whole set. The rest samples then remain as the unlabeled data. Similar to the **USPS** experiments, the SSL setting on **MNIST** also introduces the skewed class distribution in the labeled samples. In order to accelerate the running speed, we perform PCA to reduce the original $28 * 28$ image dimensions to 86 dimensions.

[3]http://yann.lecun.com/exdb/mnist/

Table 3. Classification error rates (%) on **MNIST** (70,000 samples). $m = 1000$ for two versions of AGR.

Method	$l = 100$	$l = 1000$
1NN	27.86±1.25	10.96±0.30
Linear SVM	26.60±1.45	13.22±0.40
RBF SVM	22.70±1.35	7.58±0.29
Eigenfunction	21.35±2.08	11.91±0.62
PVM(square loss)	19.21±1.70	7.88±0.18
PVM(hinge loss)	18.55±1.59	7.21±0.19
`AnchorGraphReg`^{0}	11.11±1.14	6.35±0.16
`AnchorGraphReg`	**9.40±1.07**	**6.17±0.15**

Averaged over 20 trials, we calculate the error rates for eight mentioned methods with the number of labeled samples being 100 and 1000, respectively. The results are listed in Table 3. Again, we observe that AGR ($m = 1000$) with LAE-optimized Z is superior to the other methods, which demonstrates that the linear time large graph construction approach `AnchorGraph` exhibits high quality, thus enabling more accurate graph-based SSL in large scale. The two competing large scale SSL methods, Eigenfunction and PVM, perform worse because both of them fail to construct good large graphs. PVM produces dense graphs, while Eigenfunction seems to construct backbone graphs for approximate numerical computations of eigenvectors. As the key advantage, the proposed `AnchorGraph` efficiently yields an empirically sparse adjacency matrix in which dissimilar data points would have 0 adjacency weights. Another advantage is that AGR with optimized Z introduces three parameters m, s and γ of which we only need to tune the real-valued γ with the other two fixed.

In order to test the performance in larger scale, we construct extended **MNIST** by translating the original images by one pixel in each direction, and then obtain 630,000 images like (Karlen et al., 2008). By repeating the similar evaluation process as **MNIST**, we report average classification error rates of five methods in Table 4 given 100 labeled samples. The results including average error rates and average running times shown in Table 4 further confirm the superior performance of AGR ($m = 500$) which achieves a 1/2-fold error rate reduction compared to the baseline 1NN.

6. Conclusion and Discussion

Previous SSL methods scale badly with the data size, which prevents SSL from being widely applied. This

Table 4. Classification error rates (%) on **Extended MNIST** (630,000 samples) with $l = 100$ labeled samples. $m = 500$ for two versions of AGR. The running time of `k-means` clustering is 195.16 seconds.

Method	Error Rate (%)	Running Time (seconds)
1NN	39.65±1.86	5.46
Eigenfunction	36.94±2.67	44.08
PVM(square loss)	29.37±2.53	266.89
`AnchorGraphReg`^{0}	24.71±1.92	232.37
`AnchorGraphReg`	**19.75±1.83**	331.72

paper tries to make SSL practical on large scale data collections by skillfully constructing large graphs over all data. The proposed SSL approach AGR, successfully addressing scalable SSL, is simple to understand, easy to implement, yet excellent enough to be comparable with state-of-the-arts. Both time and memory needed by AGR grow only linearly with the data size, so it can enable us to apply SSL to even larger datasets with millions of samples.

In essence, AGR has a natural out-of-sample extension and can easily apply to novel samples once we compute the regression weights $Z_{.k}$ for any novel sample. For very large datasets (millions or more) `k-means` clustering may be expensive. To run AGR, we propose to adopt random anchors or try faster clustering algorithms such as random forest clustering. We invented an effective method for mid-scale SSL problems to learn uniform graph structures in our latest paper (Liu & Chang, 2009). However, the large scale challenge poses an obstacle to graph learning. What we can do in future is to further sparsify the proposed `AnchorGraph`.

References

Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: a geometric framework for learning from examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

Chapelle, O., Schölkopf, B., and Zien, A. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, USA, 2006.

Chung, F. *Spectral Graph Theory*. No. 92 in CBMS Regional Conference Series in Mathematics, American Mathematical Society, Providence, RI, 1997.

Collobert, R., Sinz, F., Weston, J., and Bottou, L. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

Delalleu, O., Bengio, Y., and Roux, N. Le. Non-parametric function induction in semi-supervised learning. In *Proc. Artificial Intelligence and Statistics*, 2005.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proc. ICML*, 2008.

Fergus, R., Weiss, Y., and Torralba, A. Semi-supervised learning in gigantic image collections. In *NIPS 22*, 2010.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition, Springer, 2009.

Joachims, T. Transductive inference for text classification using support vector machines. In *Proc. ICML*, 1999.

Karlen, M., Weston, J., Erkan, A., and Collobert, R. Large scale manifold transduction. In *Proc. ICML*, 2008.

Liu, W. and Chang, S.-F. Robust multi-class transductive learning with graphs. In *Proc. CVPR*, 2009.

Nesterov, Y. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2003.

Roweis, S. and Saul, L. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 (5500):2323–2326, 2000.

Tsang, I. W. and Kwok, J. T. Large-scale sparsified manifold regularization. In *NIPS 19*, 2007.

Zhang, K., Kwok, J. T., and Parvin, B. Prototype vector machine for large scale semi-supervised learning. In *Proc. ICML*, 2009.

Zhou, D., Bousquet, O., Lal, T., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *NIPS 16*, 2004.

Zhu, X. Semi-supervied learning literature survey. Technical report, University of Wisconsin Madison, 2008.

Zhu, X. and Lafferty, J. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *Proc. ICML*, 2005.

Zhu, X., Ghahramani, Z., and Lafferty, J. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. ICML*, 2003.