Large Graph Construction for Scalable Semi-Supervised Learning

Wei Liu    Junfeng He    Shih-Fu Chang

Department of Electrical Engineering
Columbia University
New York, NY, USA

International Conference on Machine Learning, 2010
Outline

1. Motivation
2. Related Work
   - Mid-Scale SSL
   - Large-Scale SSL
3. Anchor-Based Label Prediction
   - Anchors
   - Anchor-to-Data Nonparametric Regression
4. AnchorGraph
   - Coupling Label Prediction and Graph Construction
   - AnchorGraph Regularization
5. Experiments
Why Semi-Supervised Learning (SSL)?

- In practice, labeling data is expensive.
- Using labeled data and unlabeled data, SSL generalizes and improves supervised learning.
- Most SSL methods **cannot** handle a large amount of unlabeled data because they scale poorly with the data size.
- Large-scale SSL is critical nowadays as we can easily collect massive (up to hundreds of millions) unlabeled data from the Internet (Facebook, Flickr).
SSL on Large Social Networks

http://www.cmu.edu/joss/content/articles/volume1/Freeman.html
Manifold and Graph

Manifold Assumption
Close samples in some similarity measure have similar class labels. Graph-based SSL methods including LGC, GFHF, LapSVM, etc. adopt this assumption.

Neighborhood Graph
A sparse graph can reveal data manifold, where adjacent graph nodes represent nearby data points. The commonly used graph is $k$NN graph (linking graph edges between any data point and its $k$ nearest neighbors) whose construction takes $O(kn^2)$. Inefficient in large scale!
Dense graphs hardly reveal manifolds.

Sparse graphs are preferred.
Existing Large-Scale SSL Methods

- **Nonparametric Function Induction, Delalleu et al., 2005**
  Use the truncated graph on a sample subset and its connection to the rest samples.

- **Harmonic Mixtures, Zhu & Lafferty, 2005**
  Need a large sparse graph, but do not explain how to construct.

- **Prototype Vector Machine (PVM), Zhang et al., 2009**
  Use the Nyström approximation of the fully-connected graph adjacency matrix. There is no guarantee for the graph Laplacian matrix computed from this approximation to be positive semidefinite.

- **Eigenfunction, Fergus et al., 2010**
  Rely on the dimension-independent data density assumption which is not always true.
Our Aim

- Develop a scalable SSL approach that is easy to implement and has closed-form solutions.
- Address the critical issue on how to build an entire sparse graph over a large dataset.
- Low space and time complexities with respect to the data size. **Our Goal: achieve linear complexity!**
Anchors

How to scale up?

• Graph-based SSL methods usually need a cubic time complexity $O(n^3)$ with respect to the data size $n$ because $n \times n$ matrix inversion is needed in inferring full-size label prediction.

• In large scale, full-size label prediction is infeasible, which has motivated the idea of landmark samples, i.e., anchors.

• If one inferred the labels on a much smaller landmark set with size $m$ ($\ll n$), the time complexity could be reduced to $O(m^3)$. 
Anchors

Anchor Points

Example

K-Means Clustering

- data
- cluster center
Anchor-to-Data Relationship

- Input data points $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \subseteq \mathbb{R}^d$ and introduced anchor points $\mathcal{U} = \{\mathbf{u}_1, \cdots, \mathbf{u}_m\}$.
- Suppose a soft label prediction function $f : \mathbb{R}^d \rightarrow \mathbb{R}$. Predict the label for each data point as a weighted average of the labels on anchors:

$$f(\mathbf{x}_i) = \sum_{k=1}^{m} Z_{ik} f(\mathbf{u}_k), \quad i = 1, \cdots, n$$  \hspace{1cm} (1)

where $Z_{ik}$'s denote the combination weights.
- The matrix form

$$\mathbf{f} = \mathbf{Za}, \quad \mathbf{Z} \in \mathbb{R}^{n \times m}, \quad m \ll n$$  \hspace{1cm} (2)

where $\mathbf{a} = [f(\mathbf{u}_1), \cdots, f(\mathbf{u}_m)]^\top$ is the label vector that we will solve.
How to obtain $Z$?

- We impose the nonnegative normalization constraints $\sum_{k=1}^{m} Z_{ik} = 1$ and $Z_{ik} \geq 0$ in order to maintain the unified range of values for all predicted soft labels via anchors.

- The manifold assumption implies that nearby points should have similar labels and distant points are very unlikely to take similar labels.

- So, we further impose $Z_{ik} = 0$ when anchor $u_k$ is far away from $x_i$. Specifically, $Z_{ik} \neq 0$ only for $s$ closest anchor points (the $s$ indexes $\langle i \rangle \subset [1 : m]$) of $x_i$. 
The weights $Z_{ik}$'s are sample-adaptive, and thus the anchor-based label prediction scheme falls into **nonparametric regression**: the label of $x_i$ is a locally weighted average of the labels on anchors.

The regression matrix $Z \in \mathbb{R}^{n \times m}$ is nonnegative and sparse.
The Nadaraya-Watson Kernel Regression

\[ Z_{ik} = \frac{K_h(x_i, u_k)}{\sum_{k' \in \langle i \rangle} K_h(x_i, u_{k'})} \quad \forall k \in \langle i \rangle \]  

- Typically, we may adopt the Gaussian kernel
  \[ K_h(x_i, u_k) = \exp(-\|x_i - u_k\|^2/2h^2). \]
- The above predefined weights are sensitive to the hyperparameter \( h \) and lack a meaningful interpretation.
Optimized Weights

Local Anchor Embedding (LAE)

\[
\min_{\mathbf{z}_i \in \mathbb{R}^s} \quad g(\mathbf{z}_i) = \frac{1}{2} \| \mathbf{x}_i - \mathbf{U}_{\langle i \rangle} \mathbf{z}_i \|^2 \\
\text{s.t.} \quad \mathbf{1}^\top \mathbf{z}_i = 1, \quad \mathbf{z}_i \geq 0
\] (4)

- \( \mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_m] \) and \( \mathbf{U}_{\langle i \rangle} \in \mathbb{R}^{d \times s} \) is a sub-matrix composed of \( s \) nearest anchors of \( \mathbf{x}_i \).
- Similar to Locally Linear Embedding (LLE), LAE reconstructs any \( \mathbf{x}_i \) as a convex combination of its nearest anchors. The combination coefficients are preserved for the weights \( Z_{ik} \)'s used in nonparametric regression:

\[
Z_{i,\langle i \rangle} = \mathbf{z}_i^\top, \quad Z_{i,\overline{\langle i \rangle}} = 0, \quad |\langle i \rangle| = s.
\] (5)
We propose the **Nesterov-accelerated projected gradient algorithm**, a first-order optimization procedure, to solve the optimization problem.

In contrast to the predefined kernel-based weights, LAE is more advantageous because it provides optimized regression weights that are also sparser.

**Example**

Project \( x_i \) onto the convex hull of its closest anchors \( U_{\langle i \rangle} \)

\[ Z_{42} = 0, \ Z_{43} = 1 \]

\[ Z_{73} = 0.5, \ Z_{74} = 0.5 \]
Construct a large graph $G(\mathcal{X}, \mathcal{W})$ over the dataset $\mathcal{X}$. $\mathcal{W} \in \mathbb{R}^{n \times n}$ denotes the adjacency matrix.

The exact $k$NN graph construction (square time complexity) is infeasible in large scale; it is unrealistic to save a full matrix $\mathcal{W}$ in memory.

Our idea is low-rank matrix factorization: representing $\mathcal{W}$ as a Gram matrix.
Principles for Designing $W$

- $W \geq 0$, which is sufficient to make the resulting graph Laplacian $L = D - W$ ($D = \text{diag}(W1)$) positive semidefinite. Keeping p.s.d. graph Laplacians is important to guarantee global optimum of graph-based SSL (ensuring that the graph regularizer $f^T L f$ is convex).

- **Sparse** $W$. Sparse graphs have much less spurious connections between dissimilar points and tend to exhibit high quality.
Nyström Approximation

- When using the Nyström method to design the graph adjacency matrix $W = \hat{K}$, produce an improper dense graph.
- $\hat{K} = K_u K_u^{-1} K_u$ approximates a predefined kernel matrix $K \in \mathbb{R}^{n \times n}$.
- Cannot guarantee $\hat{K} \geq 0$; $\hat{K}$ is dense.
In anchor-based label prediction, we have obtained the nonnegative and sparse $Z \in \mathbb{R}^{n \times m}$. Don’t waste it.

In depth, each row $Z_i$ in $Z$ is a new representation of raw sample $x_i$. $x_i \rightarrow Z_i^\top$ is reminiscent of sparse coding with the basis $U$ since $x_i \approx U_{\langle i \rangle} z_i = U Z_i^\top$.

Guess the Gram matrix $W = ZZ^\top$?
Design $W$ Using $Z$

Low-Rank Matrix Factorization

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

(6)

in which the diagonal matrix $\Lambda \in \mathbb{R}^{m \times m}$ contains inbound degrees of anchors $\Lambda_{kk} = \sum_{i=1}^{n} Z_{ik}$.

- $W$ balances the popularity of anchors.
- $W$ is nonnegative and empirically sparse.
- Theoretically, we can derive this equation by a probabilistic means (e.g., employ the bipartite graph on the next slide).
Bipartite Data-Anchor Graph

Figure: A bipartite graph representation $B(\mathcal{X}, \mathcal{U}, Z)$ of data $\mathcal{X}$ and anchors $\mathcal{U}$. $Z$ is defined as the cross adjacency matrix between $\mathcal{X}$ and $\mathcal{U}$ ($Z1 = 1$).
One-Step Transition in Markov Random Walks

\[ p^{(1)}(u_k|x_i) = \frac{Z_{ik}}{\sum_{k'=1}^{m} Z_{ik'}} = Z_{ik} \]
One-Step Transition in Markov Random Walks

\[
p^{(1)}(x_i | u_k) = \frac{Z_{ik}}{\sum_{j=1}^{n} Z_{jk}} = \frac{Z_{ik}}{\Lambda_{kk}}
\]
Two-Step Transition in Markov Random Walks

\[ p^{(2)}(x_j|x_i) = p^{(2)}(x_i|x_j) = \sum_{k=1}^{m} p^{(1)}(x_j|u_k)p^{(1)}(u_k|x_i) = \sum_{k=1}^{m} \frac{Z_{ik}Z_{jk}}{\Lambda_{kk}} \]
Interpret $W$ as two-step transition probabilities in the bipartite data-anchor graph $\mathcal{B}$:

$$W_{ij} = p^{(2)}(x_j|x_i) = p^{(2)}(x_i|x_j)$$

$$\Rightarrow W = Z \Lambda^{-1} Z^\top$$

Term the large graph $G$ described by such a $W$

AnchorGraph.

Don’t need to compute $W$ and the graph Laplacian $L = D - W$ explicitly. Only need to save $Z$ ($O(sn)$) in memory.
With $f = Za$, the object function of SSL is

$$Q(a) = \|Za - y\|^2 + \gamma a^T Z^T L Z a$$

where we compute a "reduced" Laplacian matrix:

$$\tilde{L} = Z^T L Z = Z^T (D - Z \Lambda^{-1} Z^T) Z$$

$$= Z^T Z - (Z^T Z) \Lambda^{-1} (Z^T Z),$$

where $D = \text{diag}(W1) = \text{diag}(Z \Lambda^{-1} Z^T 1) = \text{diag}(Z \Lambda^{-1} \Lambda 1) = \text{diag}(Z1) = \text{diag}(1) = I$.

Computing $\tilde{L}$ is tractable, taking $O(m^3 + m^2 n)$ time.
We can easily obtain the globally optimal solution as follows

$$a^* = (Z_i^T Z_i + \gamma \tilde{L})^{-1} Z_i^T y_i. \quad (9)$$

As such, we yield a closed-form solution for large-scale SSL.

Final prediction of multiple classes ($Y = \{1, \cdots, c\}$):

$$\hat{y}_i = \arg \max_{j \in Y} \frac{Z_i a^*_j}{\lambda_j}, \quad i = l + 1, \cdots, n \quad (10)$$

where $a^*_j$ is solved corresponding to each class and the normalization factor $\lambda_j = 1^T Z a^*_j$ balances skewed class distributions.
Complexity Analysis

**Space Complexity**

\[ O(n) \] since we save \( Z \) and \( \tilde{L} \in \mathbb{R}^{m \times m} \) in memory.

**Time Complexity**

\[ O(m^2n) \] since \( n \gg m \).

**Table:** Time complexity analysis. \( n \) is the data size, \( m \) is \# of anchor points, \( s \) is \# of nearest anchors in designing \( Z \), and \( T \) is \# of iterations of LAE (\( n \gg m \gg s \)).

<table>
<thead>
<tr>
<th>Stage</th>
<th>AnchorGraphReg</th>
</tr>
</thead>
<tbody>
<tr>
<td>find anchors</td>
<td>( O(mn) )</td>
</tr>
<tr>
<td>design ( Z )</td>
<td>( O(smn) ) or ( O(smn + s^2 Tn) )</td>
</tr>
<tr>
<td>graph regularization</td>
<td>( O(m^3 + m^2 n) )</td>
</tr>
<tr>
<td>total time complexity</td>
<td>( O(m^2 n) )</td>
</tr>
</tbody>
</table>
Compared Methods

- LGC, GFHF, PVM, Eigenfunction, etc.
- random AGR$^0$: random anchors, predefined $Z$
- random AGR: random anchors, LAE-optimized $Z$
- AGR$^0$: \texttt{k-means} anchors, predefined $Z$
- AGR: \texttt{k-means} anchors, LAE-optimized $Z$
Mid-sized Dataset

**Table:** Classification error rates (%) on **USPS-Train** (7,291 samples) with 100 labeled samples. Take 1,000 anchors for four versions of AGR. The running time of k-means clustering is 7.65 seconds. Error rate $\text{GFHF} < \text{AGR} < \text{AGR}^0 < \text{LGC}$. AGR is much faster than GFHF.

<table>
<thead>
<tr>
<th>Method</th>
<th>Error Rate (%)</th>
<th>Running Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1NN</td>
<td>20.15±1.80</td>
<td>0.12</td>
</tr>
<tr>
<td>LGC with 6NN graph</td>
<td>8.79±2.27</td>
<td>403.02</td>
</tr>
<tr>
<td>GFHF with 6NN graph</td>
<td>5.19±0.43</td>
<td>413.28</td>
</tr>
<tr>
<td>random AGR$^0$</td>
<td>11.15±0.77</td>
<td>2.55</td>
</tr>
<tr>
<td>random AGR</td>
<td>10.30±0.75</td>
<td>8.85</td>
</tr>
<tr>
<td>AGR$^0$</td>
<td>7.40±0.59</td>
<td>10.20</td>
</tr>
<tr>
<td><strong>AGR</strong></td>
<td><strong>6.56±0.55</strong></td>
<td><strong>16.57</strong></td>
</tr>
</tbody>
</table>
Mid-sized Dataset

**Figure:** 100 labeled samples. When the anchor size reaches 600, both AGR\(^0\) and AGR begin to outperform LGC.
Table: Classification error rates (%) on Extended MNIST (630,000 samples) with 100 labeled samples. Take 500 anchors for two versions of AGR. The running time of $k$-means clustering is 195.16 seconds. AGR is the best in classification accuracy, achieving 1/2-fold improvement over the baseline 1NN.

<table>
<thead>
<tr>
<th>Method</th>
<th>Error Rate (%)</th>
<th>Running Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1NN</td>
<td>39.65 ± 1.86</td>
<td>5.46</td>
</tr>
<tr>
<td>Eigenfunction</td>
<td>36.94 ± 2.67</td>
<td>44.08</td>
</tr>
<tr>
<td>PVM(square loss)</td>
<td>29.37 ± 2.53</td>
<td>266.89</td>
</tr>
<tr>
<td>AGR$^0$</td>
<td>24.71 ± 1.92</td>
<td>232.37</td>
</tr>
<tr>
<td>AGR</td>
<td>19.75 ± 1.83</td>
<td>331.72</td>
</tr>
</tbody>
</table>
Conclusions

- Develop a scalable SSL approach using the proposed large graph construction, **AnchorGraph**, which is guaranteed to result in positive semidefinite graph Laplacians.

- 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.

- **AnchorGraph** has wider utility such as large-scale nonlinear dimensionality reduction, spectral clustering, regression, etc.
Thanks! Ask me via wliu@ee.columbia.edu.