CS260: Machine Learning Algorithms
Lecture 14: Semi-supervised learning, graph convolution network

Cho-Jui Hsieh
UCLA

Mar 6, 2019
Outline

- Label Propagation
- Manifold regularization
- Graph convolutional neural network
- Other graph-based algorithms
Label Propagation
Semi-supervised Learning

- Given both labeled and unlabeled data
- Is unlabeled data useful?

Figure from Wikipedia
Semi-supervised Learning

- Graph-based semi-supervised learning:
  Encode the unlabeled feature information into graph

- Two classical approaches:
  - Graph-based algorithm (label propagation)
  - Graph-based regularization (manifold regularization)
## Inductive vs Transductive

<table>
<thead>
<tr>
<th></th>
<th>Inductive (Generalize to unseen data)</th>
<th>Transductive (Doesn’t generalize to unseen data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>SVM, ...</td>
<td>X</td>
</tr>
<tr>
<td>(labeled data)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semi-supervised</td>
<td>Manifold Regularization</td>
<td>Label propagation</td>
</tr>
<tr>
<td>(labeled + unlabeled)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Main Idea

- **Smoothness Assumption:**
  
  If two data points are close to each other, their predictions should be similar.

Main Idea

- Smoothness Assumption:
  - If two data points are close to each other, their predictions should be similar
- Measure the similarity between data points (similarity graph)
- Enforce the predictions to be similar based on similarity graph
Similarity Graph

- Assume we have $n$ data points $x_1, \ldots, x_n$
- Define the similarity matrix $S \in \mathbb{R}^{n \times n}$

\[ S_{ij} = \text{similarity}(x_i, x_j) \]

- The similarity function can be defined by many ways, for example,

\[ \text{similarity}(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2) \]

- $S$ is a dense $n \times n$ matrix $\Rightarrow$ High computational cost
  
  Usually, a sparse similarity graph is preferred

- Usually, a k-nearest neighbors graph is used:
  
  $S_{ij} \neq 0$ only when $j$ is in $i$’s k-nearest neighbors
Transductive setting using label propagation

- **Input:**
  - $\ell$ labeled training samples $x_1, \ldots, x_\ell$ with labels $y_1, \ldots, y_\ell$ (each $y_i$ is a $k$ dimensional label vector for multiclass/multilabel problems)
  - $u$ unlabeled training samples $x_{\ell+1}, \ldots, x_{\ell+u}$

- **Output:** labels $y_{\ell+1}, \ldots, y_{\ell+u}$ for all unlabeled samples

- **Main idea:** propagate labels through the similarity matrix
Label Propagation

- $\mathbf{T} \in \mathbb{R}^{(\ell+u) \times (\ell+u)}$: transition matrix such that
  \[
  T_{ij} = P(j \rightarrow i) = \frac{S_{ij}}{\sum_{k=1}^{\ell+u} S_{kj}}
  \]
- $\mathbf{Y} \in \mathbb{R}^{(\ell+u) \times k}$: the label matrix. The first $\ell$ rows are labels $y_1, \ldots, y_\ell$. The rest $u$ rows are initialized by 0 (will not affect the results).
- The Algorithm:
  Repeat the following steps until convergence:
  Step 1: Propagate labels by the transition matrix: $\mathbf{Y} \leftarrow \mathbf{T} \mathbf{Y}$
  Step 2: Normalize each row of $\mathbf{Y}$
  Step 3: Reset the first $\ell$ rows of $\mathbf{Y}$ to be $y_1, \ldots, y_\ell$
The algorithm will converge to a simple solution!

Step 1 and step 2 can be combined into

\[ Y \leftarrow \bar{T}Y, \]

where \( \bar{T} \) is the row-normalized matrix of \( T \)

We focus on row \( \ell + 1 \) to \( \ell + u \) (defined by \( Y_U \))

\[
\begin{pmatrix}
Y_L \\
Y_U
\end{pmatrix}
\leftarrow
\begin{pmatrix}
\bar{T}_{\ell\ell} & \bar{T}_{\ell u} \\
\bar{T}_{ul} & \bar{T}_{uu}
\end{pmatrix}
\begin{pmatrix}
Y_L \\
Y_U
\end{pmatrix}
\]
Label Propagation (convergence)

- So \( Y_U \leftarrow \mathbf{T}_{ul} Y_L + \mathbf{T}_{uu} Y_U \)
- Run for infinite number of iteration, we have

\[
Y_U^* = \lim_{t \to \infty} \left( \mathbf{T}_{uu} Y_U^0 \underbrace{+ \left( \sum_{i=1}^{t} \mathbf{T}_{uu}^{-1} \right) \mathbf{T}_{ul} Y_L}_{\text{vanish when } t \to \infty} \right)
\]

\[
= \lim_{t \to \infty} \left( \sum_{i=1}^{t} \mathbf{T}_{uu}^{-1} \right) \mathbf{T}_{ul} Y_L
\]

\[
= (I - \mathbf{T}_{uu})^{-1} \mathbf{T}_{ul} Y_L
\]

- \( O(\text{nnz}(S)k) \) per power iteration
- In the following, we show another interpretation.
Graph Laplacian

- Graph Laplacian:

\[ L = D - S, \quad \text{where } D \text{ is a diagonal matrix with } D_{ii} = \sum_j S_{ij} \]

- \( L \) is positive semi-definite (if \( S \) is nonnegative)

- Main property: for any vector \( z \),

\[ z^T L z = \sum_{i,j} S_{ij} (z_i - z_j)^2 \]

- Measure the non-smoothness of \( z \) according to the similarity matrix \( S \)
Another form of label propagation

- Assume we only have one label (binary classification)
- Another equivalent form of label propagation:

\[
\begin{align*}
\arg\min_{\hat{y}} \sum_{i,j} S_{ij}(\hat{y}_i - \hat{y}_j)^2 &= \hat{y}^T L \hat{y} := f(\hat{y}) \\
\text{s.t.} \ & \hat{y}_{1:l} = y
\end{align*}
\]

where \( \hat{y} \in \mathbb{R}^{l+u} \) is the estimation of the labels.

- Optimal solution: \( \nabla_{\hat{y}_{l+1:u}} f(\hat{y}) = 0 \)

\[
\begin{align*}
(D_{ll} &- S_{ll} - S_{lu}) \hat{y}_{l} \\
&- (D_{ul} - S_{uu}) \hat{y}_{u} = 0 \\

\Rightarrow \quad \hat{y}_{u} &= (D_{uu} - S_{uu})^{-1} S_{ul} \hat{y}_{l} \\
\Rightarrow \quad \hat{y}_{u} &= (I - D_{uu}^{-1} S_{uu})^{-1} D_{uu}^{-1} S_{ul} \hat{y}_{l} \\
\Rightarrow \quad \hat{y}_{u} &= (I - \bar{T}_{uu})^{-1} \bar{T}_{ul} \hat{y}_{l}
\end{align*}
\]
Experimental Results (Zhu et al., 2003)

Figure 3. Harmonic energy minimization on digits “1” vs. “2” (left) and on all 10 digits (middle) and combining voted-perceptron with harmonic energy minimization on odd vs. even digits (right)

Figure 4. Harmonic energy minimization on PC vs. MAC (left), baseball vs. hockey (middle), and MS-Windows vs. MAC (right)
Manifold Regularization
Empirical Risk Minimization with Manifold Regularization

- Setting: given labeled training data \( x_1, \ldots, x_\ell \) with labels \( y_1, \ldots, y_\ell \) and unlabeled training data \( x_{\ell+1}, \ldots, x_{\ell+u} \), obtain a function \( f \) to predict the label of new instances.
- We assume \( f \) is a linear function \( (f(x) = w^T x) \)
- Empirical risk minimization:

\[
\min_w \sum_{i=1}^{\ell} \ell(y_i, w^T x_i) + \lambda R(w),
\]

where \( \ell(\cdot) \) is the loss function and \( R(\cdot) \) is the regularization.

- Only use labeled data.
We assume $f$ is a linear function ($f(x) = w^T x$)

Empirical risk minimization with Manifold Regularization:

$$\min_w \sum_{i=1}^{l} \ell(y_i, w^T x_i) + \lambda R(w) + \beta \sum_{i,j} S_{ij} (w^T x_i - w^T x_j)^2$$

$$= \sum_{i=1}^{l} \ell(y_i, w^T x_i) + \lambda R(w) + \beta \hat{y}^T L \hat{y},$$

where $\hat{y} = [w^T x_1, w^T x_2, \ldots, w^T x_{\ell+u}]^T$

Use manifold regularization to ensure that similar data points have the similar output labels
Manifold Regularization

- Train an “inductive” classifier: can generalize to unseen data
- Can be extended to other function classes (e.g., kernel methods)
- Can be used in other ML problems
  - Ranking, Matrix Completion, ...
Experimental Comparisons

Performance of RLS, LapRLS

Performance of SVM, LapSVM

PRBEP

rls (U)  rls (T)  laprls (U)  laprls (T)

svm (U)  svm (T)  lapsvm (U)  lapsvm (T)
Graph Convolutional Neural Network
Node classification problem

- Given a graph of $N$ nodes, with adjacency matrix $A \in \mathbb{R}^{N \times N}$
- Each node is associated with a $D$-dimensional feature vector.
- $X \in \mathbb{R}^{N \times D}$: each row corresponds to the feature vector of a node
- Observe labels for a subset of nodes: $Y \in \mathbb{R}^{N \times L}$, only observe a subset of rows, denoted by $Y_S$
- Goal: Predict labels for unlabeled nodes (transductive setting) or test nodes (inductive setting) or test graphs (inductive setting)
Graph Convolution Layer

- GCN: multiple graph convolution layers
- $\hat{A}$: normalized version of $A$:
  $$\tilde{A} = A + I, \quad \tilde{D}_{uv} = \sum_v \tilde{A}_{uv}, \quad P = \tilde{D}^{-1} \tilde{A}$$

- Graph convolution:
  - Input: features for each node $H^{(l)} \in \mathbb{R}^{n \times D}$
  - Output: features for each node $H^{(l+1)}$ after gathering neighborhood information
  - Convolution: $PH^{(l)}$: Aggregate features from neighbors
  - Convolution + fully-connected layer + nonlinear activation:
    $$H^{(l+1)} = \sigma(PH^{(l)}W^{(l)})$$

$W^{(l)}$ is the weights for the linear layer
$\sigma(\cdot)$: usually ReLU function
Graph convolutional network

(from Hamilton et al., “Inductive Representation Learning on Large Graphs”)

Graph convolutional network

- Initial features $H^{(0)} := X$
- For layer $l = 0, \ldots, L$
  \[ Z^{(l+1)} = PH^{(l)} W^{(l)}, \quad H^{(l+1)} = \sigma(Z^{(l+1)}), \]
- Use final layer feature $H^{(L)} \in \mathbb{R}^{N \times K}$ for classification:
  \[ \text{Loss} = \frac{1}{|S|} \sum_{s \in S} \text{loss}(y_s, Z_s^{(L)}) \]

  - Each row of $Z_s^{(L)}$ corresponds to the output score for each label.
  - Cross-entropy loss for classification.
Graph convolutional network

- Model parameters: $W^{(1)}, \ldots, W^{(L)}$
- Can be used to
  - Predict unlabeled nodes in the training set
  - Predict testing nodes (not in the training set)
  - Predict labels for a new graph
- Also, features extracted by GCN $H^{(L)}$ is usually very useful for other tasks
GCN training

- Full Gradient descent in the original paper (Kipf & Welling, 2017):
  - Need many iterations (epochs)
  - Large memory requirement: $O(NDL)$ for storing all the intermediate embeddings

- GraphSAGE (Hamilton et al., 2017):
  - Stochastic gradient descent
  - Can use several different aggregation operations
  - Neighborhood explosion problem and the need of neighborhood sampling.
PageRank/Hubs and Authorities
Text based ranking systems (a dominated approach in the early 90s)
- Compute the similarity between query and websites (documents)
- Keywords are a very limited way to express a complex information
- Need to rank websites by “popularity”, “authority”, . . .

PageRank:
- Developed by Brin and Page (1999)
- Determine the authority and popularity by hyperlinks
PageRank

Main idea: estimate the ranking of websites by the link structure.

Topology of Websites

- Transform the hyperlinks to a directed graph:
- The adjacency matrix $A$ such that $A_{ij} = 1$ if page $j$ points to page $i$
Transition Matrix

- Normalize the adjacency matrix so that the matrix is a stochastic matrix (each column sum up to 1)
- $P_{ij}$: probability that arriving at page $i$ from page $j$
- $P$: a stochastic matrix or a transition matrix

```
0  0  1  1/2
1/3  0  0  0
1/3  1/2  0  1/2
1/3  1/2  0  0
```
Random walk: step 1

- Random walk through the transition matrix
- Start from $[1, 0, 0, 0]$ (can use any initialization)
Random walk: step 1

- Random walk through the transition matrix
- $x^{t+1} = Px^t$
Random walk: step 2

- Random walk through the transition matrix
Random walk: step 2

- Random walk through the transition matrix
- $x^{t+2} = Px^{t+1}$
PageRank (convergence)

PageRank Algorithm

- Start from an initial vector $x$ with $\sum_i x_i = 1$ (the initial distribution)
- For $t = 1, 2, \ldots$
  \[ x^{t+1} = Px^t \]

- Each $x^t$ is a probability distribution (sums up to 1)
PageRank (convergence)

PageRank Algorithm

- Start from an initial vector $\mathbf{x}$ with $\sum_i x_i = 1$ (the initial distribution)
- For $t = 1, 2, \ldots$
  \[ \mathbf{x}^{t+1} = P \mathbf{x}^t \]

- Each $\mathbf{x}^t$ is a probability distribution (sums up to 1)
- Will converge to a stationary distribution $\pi$ such that
  \[ \pi = P\pi \]

if $P$ satisfies the following two conditions:

1. $P$ is irreducible: for all $i, j$, there exists some $t$ such that $(P^t)_{i,j} > 0$
2. $P$ is aperiodic: for all $i, j$, we have gcd\{ $t : (P^t)_{i,j} > 0$ \} = 1
PageRank (convergence)

**PageRank Algorithm**

- Start from an initial vector $\mathbf{x}$ with $\sum_i x_i = 1$ (the initial distribution)
- For $t = 1, 2, \ldots$
  $$\mathbf{x}^{t+1} = P\mathbf{x}^t$$

- Each $\mathbf{x}^t$ is a probability distribution (sums up to 1)
- Will converge to a stationary distribution $\pi$ such that
  $$\pi = P\pi$$

if $P$ satisfies the following two conditions:

1. $P$ is irreducible: for all $i, j$, there exists some $t$ such that $(P^t)_{i,j} > 0$
2. $P$ is aperiodic: for all $i, j$, we have $\gcd\{t : (P^t)_{i,j} > 0\} = 1$

- $\pi$ is the unique right eigenvector of $P$ with eigenvalue 1.
How to guarantee convergence?

Add the possibility of jumping to a random node with small probability $\alpha$, we get the commonly used PageRank

$$\pi = \left( \alpha P + (1 - \alpha)ve^T \right) \pi$$

- $v = \frac{1}{n}e = [\frac{1}{n} \quad \frac{1}{n} \quad \ldots \quad \frac{1}{n}]^T$ is commonly used
- Personalized PageRank: $v = e_i$
PageRank: Summary

**Input:** Transition matrix \( P \) and personalization vector \( \mathbf{v} \)

1. Initial \( x_i^{(0)} = \frac{1}{n} \) for \( i = 1, 2, \ldots, n \)

2. for \( t = 1, 2, \ldots \) do
   - \( x^{(t+1)} \leftarrow \alpha P x^{(t)} + (1 - \alpha) \mathbf{v} \)

3. end for
Hubs and Authorities

- Proposed by (Kleinberg, 1999)
- Also, identify important websites on the Internet
- Main idea: two types of scores
  - “authority”: a web page with authoritative content ⇒ it is pointed to by many hub pages
  - “hub”: a web page pointing to many authoritative web pages.
Hubs and Authorities

- Proposed by (Kleinberg, 1999)
- Also, identify important websites on the Internet
- Main idea: two types of scores
  - “authority”: a web page with authoritative content $\Rightarrow$ it is pointed to by many hub pages
  - “hub”: a web page pointing to many authoritative web pages.
- Let $h \in \mathbb{R}^n$ be the hubs score and $a \in \mathbb{R}^n$ be the authority score for $n$ web pages.
- Initialize $h = [1, 1, \cdots, 1]$, $a = [1, 1, \cdots, 1]$
- $M \in \mathbb{R}^{n \times n}$ is the network graph:

  $$M_{ij} = 1 \text{ means page } i \text{ links to page } j$$

  $$M_{ij} = 0 \text{ otherwise.}$$
Hubs and Authorities

- "Authority" of a page: Counting how many in-links to each page.

\[
a_i = \sum_{j=1}^{n} M_{ji} h_j \quad a = M^T h
\]
A hub page should link to many pages with high authority: Page’s hub value is the sum of the authority scores of all the pages it links to

\[ h_i = \sum_{j=1}^{n} M_{ij} a_j \quad h = Ma \]
Hubs and Authority

- Re-compute authority: each page’s new authority score is equal to the sum of the “hub” scores that point to it.

\[ a = M^T h \]
Hubs and Authorities

- Normalize $a, h$ after each iteration.
- After infinite number of iterations:

$$a \propto (M^T M)^\infty M^T 1$$
$$h \propto (MM^T)^\infty 1$$

- Therefore,
  - Authority score $a$ is the leading eigenvector of $M^T M$
    $\Rightarrow$ the leading right singularvector of $M$
  - Hub score $h$ is the leading eigenvector of $MM^T$
    $\Rightarrow$ the leading left singularvector of $M$
Conclusions

- Semi-supervised learning
- Other graph-based algorithms

Questions?