Neighborhood-Based Classification

Exploiting Manifold Information in Statistical Learning

Xuejun Liao, Qiuhua Liu, Chunping Wang, and Lawrence Carin

{xjliao,ql1,cw36,lcarin}@ee.duke.edu

Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708-0291, USA
Outline

- Problem Statement
- Markov random walk on a data manifold
- Neighborhood-Based Classification (NeBC)
  - The nonparametric case
  - The parametric case
  - The semi-parametric case (Dependent Dirichlet distribution)
- Experimental Results
- Summary
Problem Statement

- Given: a discrete data manifold $\mathcal{X} = \{x_i : i = 1, 2, \ldots, N\}$, of which small subset indexed by $\mathcal{L}$ is labeled

- Objective:
  - complete the labels for $\mathcal{X}$ (transduction)
  - predict the labels for data outside of $\mathcal{X}$ (induction)

- Problems
  - Scarce labels, insufficient supervision, over-fitting
  - Data manifold unexploited
  - Lack of prior in choosing classifiers

- Basic approach
  - Modeling $\mathcal{X}$ as a graph, data connected by Markov random walk
  - Labeling a data point based on its neighborhood
  - Using Dirichlet distributions (DD) as nonparametric classifiers
Markov Random Walk

- Graph model of $\mathcal{X}$
  - $G = (\mathcal{X}, W)$
  - $W = [w_{ij}]_{N \times N}$ is the affinity matrix, with $w_{ij}$ the strength of immediate connection between $x_i$ and $x_j$

$$w_{ij} = \exp\left(\frac{|x_i - x_j|^2}{2d_i^2}\right)$$ (1)

- Markov random walk with one-step transitions

$$T_{ij} = p(x_j | x_i) = \begin{cases} \frac{W_{ij}}{\sum_{k \neq i} W_{ik}}, & j \neq i \\ 0, & j = i \end{cases}$$ (2)

$d_i$ is roughly the stepsize at $x_i$
Nonparametric NeBC (1/3)

Let \( g_{ik} = p(y_i = k | x_i) \) and \( g_i = [g_{i1}, g_{i2}, \cdots, g_{iK}] \), then

\[
y_i | g_i \sim \text{Mult}(y_i; g_i) = \prod_{k=1}^{K} (g_{ik}) \delta(y_i - k)
\] (3)

Finite-steps random walks:

Let \( 0 < \gamma < 1 \) be a discount factor of the neighbors. Define

\[
g_i^{(n)} = (1 - \gamma)g_i^* + \gamma \sum_{j=1}^{N} T_{ij} g_j^{(n-1)}, \quad i = 1, 2, \cdots, N
\] (4)

\[
G^{(n)} = (1 - \gamma)G^* + \gamma TG^{(n-1)}
\] (5)

where \( G = \begin{bmatrix} g_1 \\ \vdots \\ g_N \end{bmatrix} \) and \( G^* = \begin{bmatrix} g_1^* \\ \vdots \\ g_N^* \end{bmatrix} \)
Table 1: The Label Iteration Algorithm

<table>
<thead>
<tr>
<th>Input:</th>
<th>(\mathbf{T}, \gamma, n^*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>(G^{(n^*)})</td>
</tr>
</tbody>
</table>

**Initialization:**

\[
g_{ik}^{(0)} = \begin{cases} 
1, & \text{if } x_i \text{ labeled, and } y_i = k \\
0, & \text{if } x_i \text{ labeled, and } y_i \neq k \\
1/K, & \text{if } x_i \text{ unlabeled}
\end{cases}
\]

**Label Iteration:** Repeat until \(n = n^*\),

\[
G^{(n)} = (1 - \gamma)G^* + \gamma T G^{(n-1)}
\]
Nonparametric NeBC (3/3)

Convergence of Label Iteration:

\[
\| G^{(n+1)} - G^{(n)} \| = \| \gamma T G^{(n)} - \gamma T G^{(n-1)} \| \\
\leq \gamma \| T \| \| G^{(n)} - G^{(n-1)} \| \\
\leq \| G^{(n)} - G^{(n-1)} \|
\]

(6)

since \( \gamma < 1 \) and \( \| T \| \leq 1 \).

Infinite-steps random walks (0 < \( \gamma < 1 \)):

\[
G^{(\infty)} = (1 - \gamma)(I - \gamma T)^{-1} G^*  
\]

(7)

Let \( [b_{ij}]_{N \times N} = B = (1 - \gamma)(I - \gamma T)^{-1} \), then \( b_{ij} \geq 0 \) and \( \sum_{j=1}^{N} b_{ij} = 1 \).
Let $\sigma_{ik} = p(y_i = k | x_i, \theta)$ be a classifier parameterized by $\theta$, $\sigma_i = [\sigma_{i1}, \cdots, \sigma_{iK}]$, and

$$y_i | g_i \sim \text{Mult}(y_i; g_i) = \prod_{k=1}^{K} (g_{ik})^{\delta(y_i - k)}$$

(8)

$$g_i = \sum_{j=1}^{N} b_{ij} \sigma_j$$

(9)

where $[b_{ij}]_{N \times N} = B = (1 - \gamma)(I - \gamma T)^{-1}$.

In the case the classifier is a multinomial logistic regression,

$$\sigma_{ik} = \frac{\exp(\theta_k^T x_i)}{\sum_{j=1}^{K} \exp(\theta_j^T x_i)}.$$
Estimating $\theta$ by maximizing the likelihood of observed labels

$$p(\{y_i : i \in \mathcal{L}\}|\mathcal{X}, \theta) = \prod_{i \in \mathcal{L}} \prod_{k=1}^{K} \left( \sum_{j=1}^{N} \psi_{ij} \right)^{\delta(y_i - k)}$$

$$= \prod_{i \in \mathcal{L}} \sum_{j=1}^{N} \psi_{ij} \ln p(y_j = y_i|\mathbf{x}_j, \theta) \quad (10)$$

The Expectation-Maximization (EM) algorithm:

$$E - \text{step} : \left\{ \begin{array}{l}
\psi_{ij} = \frac{b_{ij}p(y_j = y_i|\mathbf{x}_j, \theta^n)}{\sum_{j=1}^{N} b_{ij}p(y_j = y_i|\mathbf{x}_j, \theta^n)} \\
Q(\theta|\theta^n) = \sum_{i \in \mathcal{L}} \sum_{j=1}^{N} \psi_{ij} \ln p(y_j = y_i|\mathbf{x}_j, \theta) \end{array} \right. \quad (11)$$

$$M - \text{step} : \quad \theta^{n+1} = \theta^n + c \left[ \nabla^2 Q(\theta|\theta^n) \right]^{-1} \nabla Q(\theta|\theta^n)$$
Semi-parametric NeBC (1/6)

Let \( g_{ik} = p(y_i = k|x_i) \), \( g_i = [g_{i1}, g_{i2}, \cdots, g_{iK}] \), then

\[
y_i|g_i \sim \text{Mult}(y_i; g_i) = \prod_{k=1}^{K} (g_{ik})^{\delta(y_i=k)}
\]  

(12)

\( g_i \) is governed by a dependent Dirichlet distribution (DDD) defined by

\[
g_i = \sum_{j=1}^{N} b_{ij} g_j^* \tag{13}
\]

\[
[b_{ij}]_{N \times N} = B = (1 - \gamma)(I - \gamma T)^{-1}
\]  

(14)

\[
g_i^* \sim \text{Dir}(g_i^*; \alpha \sigma_i) = \frac{\Gamma(\sum_{k=1}^{K} \alpha \sigma_{ik})}{\prod_{k=1}^{K} \Gamma(\alpha \sigma_{ik})} \prod_{k=1}^{K} (g_{ik}^*)^{\alpha \sigma_{ik} - 1}
\]  

(15)

where \( \alpha > 0, \sigma_i = [\sigma_{i1}, \cdots, \sigma_{iK}] \), and \( \sigma_{ik} = p(y_i = k|x_i) \) is the base classifier at location \( x_i \); the base classifier can be parameterized.
Semi-parametric NeBC (2/6)

Let \( z_i = j \) indicates \( g_i = g^*_j \), then

\[
y_i | z_i, g^* \sim \text{Mult}(y_i; g^*_z) = \prod_{k=1}^{K} (g^*_{z_k}) \delta(y_i - k)
\]

(16)

Conditional on \( z \), we can obtain the full conditionals

\[
y_n | \{y_i\}_{i \neq n}, z \sim \frac{\alpha \sigma_{zn}y_n + \sum_{i \neq n}^{N} \delta(z_i - z_n) \delta(y_i - y_n)}{\alpha + \sum_{i \neq n}^{N} \delta(z_i - z_n)}
\]

(17)
Semi-parametric NeBC (3/6)

Marginalizing $z$, using $p(z_i = j) = b_{ij}$, we get

$$ y_n \mid \{ y_i \}_{i \neq n}, b \sim \sum_{j=1}^{N} \rho_{nj} \sigma_j y_n + \sum_{l \neq n}^{N} \lambda_{nl} \delta(y_n - y_l) $$

(18)

$$ \rho_{nj} = b_{nj} \sum_{m=0}^{N-1} \frac{\alpha p(\sum_{i \neq n}^{N} \delta(z_i - j) = m)}{\alpha + m} $$

(19)

$$ \lambda_{nl} = \sum_{j=1}^{N} b_{nj} b_{lj} \sum_{m=0}^{N-2} \frac{p(\sum_{i \neq n,l}^{N} \delta(z_i - j) = m)}{\alpha + m + 1} $$

(20)

with $\sum_{j=1}^{N} \rho_{nj} + \sum_{l \neq n}^{N} \lambda_{nl} = 1$.

$$ \lim_{\alpha \to \infty} \rho_{nj} = b_{nj}, \lim_{\alpha \to \infty} \lambda_{nl} = 0. \text{ However } \lim_{\alpha \to 0} \rho_{nj} > 0. $$
Any parameterized classifier can be used as the base classifier.

For example, the base can be the parametric classifier discussed earlier.

With an inaccurate base classifier, the DDD can give improved prediction of the missing labels by exploiting the data manifold.

Open questions

- How to learn the base classifier using the exact formulation of the DDD?
- Why does the base not vanish when $\alpha$ goes to zero?
Let us state the problem in a general context. Assume \( \mathbf{p} = [p_1, p_2, \cdots, p_n] \) and \( \mathbf{q} = [q_1, q_2, \cdots, q_n] \) are two vectors. We want to decompose \((p_1 + q_1)(p_2 + q_2) \cdots (p_n + q_n)\) into a sum of \( n + 1 \) terms such that the \( m \)-th term involves \( m \) and only \( m \) \( p \)'s, for \( m = 0, 1, 2, \cdots, n \).

We call this problem the *non-stationary* triangle problem, since it specializes to the classic Pascal triangle when \( \mathbf{p} = \mathbf{q} \) are vectors of all 1’s. We give a fast algorithm in Table 2, to solve the *non-stationary* triangle problem.
Table 2: The Non-stationary Triangle Algorithm

Input: \( p = [p_1, p_2, \cdots, p_n] \)
Output: \( f = [f_0, f_2, \cdots, f_n] \)

\[
q = 1 - p;
\]
\[
f = [q_1, p_1];
\]
\[
\text{for } i = 2 \text{ to } n
\]
\[
f_{\text{new}} = [q_i f, 0] + [0, p_i f];
\]
\[
f = f_{\text{new}};
\]
\[
\text{endfor}
\]
Results on Two Helixes

- Algorithm tested: Nonparametric NeBC (Label Iteration)
- movie of label-iteration
Results on Benchmark Data

- Data sets: Pima, Ionosphere, and WDBC
- Training on both labeled and unlabeled data;
- Transduction: testing on unlabeled data seen in training
- Induction: testing on unseen data
- Algorithms being compared:
  - The proposed parametric NeBC
  - The transductive SVM (Joachims, 1999)
  - The algorithm of Szummer & Jaakkola (Szummer & Jaakkola, 2002)
  - GRF (Zhu et. al, 2003)
  - Logistic GRF (Krishnapuram et. al, 2005)
Transduction with parametric NeBC: Benchmark Data

Figure 1: Transductive results. Each curve is an average from 20 independent trials. The horizontal axis is the size of $\mathcal{X}_L$. The algorithms are tested on $\mathcal{X}_U$. The algorithm of Szummer & Jaakkola (Szummer & Jaakkola, 2002) and ours use $\sigma_i = \min_j \|x_i - x_j\|/3$ and $t = 100$. 

Liao, Liu, Wang, and Carin, Neighborhood-Based Classification, Talk at Yale University, October 16, 2006 – p. 18/20
Figure 2: Inductive results. Each curve is an average from 50 independent trials. The horizontal axis is the size of $\mathcal{X}_U$. From left to right in the sub-figures, the size of $\mathcal{X}_L$ is 10, 20, 30, 40. The algorithms are tested on 200 data randomly sampled from $\mathcal{X} \setminus (\mathcal{X}_L \cup \mathcal{X}_U)$. Error bars are shown for the proposed algorithm, which uses $\sigma_i = \min_j \|x_i - x_j\|/3$ and $t = 100$. 

Liao, Liu, Wang, and Carin, *Neighborhood-Based Classification*, Talk at Yale University, October 16, 2006

— p. 19/20
Summary

- The nonparametric NeBC works well when the data of the same class are well connected.

- The parametric NeBC facilitate inductive classification and improves generalization when the parametric form is chosen right.

- The semi-parametric NeBC (DDD) is expected to combine the features of the two.

- Undergoing work on how to learn the base classifier with the exact formulation of the DDD.