Kernel Learning Using Neural Networks

Renqiang Min
Machine Learning Group
University of Toronto
Adviser: Tony Bonner and Zhaolei Zhang

Aug 11, 2007
CIAR Summer School
Outline

Previous Kernel Learning Methods

Kernel Learning Using Neural Networks

Ongoing Work
Training part and test part of $K$

$$K = \begin{bmatrix} \text{TrainingPart}_{N \times N} & [\text{TestPart}^T]_{N \times T} \\ \text{TestPart}_{T \times N} & \text{unused} \end{bmatrix}$$

$T$ is the size of the test set and $N$ is the size of the training set. $K$ is a $(N + T) \times (N + T)$ matrix.
Existing kernel learning methods

- diffusion kernels
- linear combinations of kernels based on Kernel Alignment with SDP
- hyperkernels
- convex combinations of kernels via semi-infinite linear programming
Kernel Alignment

Kernel Alignment aligns a linear combination of kernels, $K_1, K_2, \cdots, K_m$, to an optimal kernel computed using class information of the training data.

A column vector $y$ contains the binary class membership of all training data points, $K_{\text{opt}} = yy^T$, where $y \in \{-1, +1\}^N$ and $N$ is the size of the training set.

The objective function of Kernel Alignment is

$$\ell = \frac{\text{Tr}(K_{\text{tr}} K_{\text{opt}}^T)}{\sqrt{\text{Tr}(K_{\text{tr}} K_{\text{tr}}^T) \text{Tr}(K_{\text{opt}} K_{\text{opt}}^T)}} = \frac{\text{Tr}(K_{\text{tr}} K_{\text{opt}}^T)}{N \sqrt{\text{Tr}(K_{\text{tr}} K_{\text{tr}}^T)}}$$

(1)

where $K = \theta_1 K_1 + \theta_2 K_2 + \cdots + \theta_m K_m$, $K \succeq 0$, and $\text{tr}$ denotes the training part of $K$. 
Limitations of Existing Kernel Learning Methods

- Use blackbox packages to optimize
- Computationally Expensive
- Impractical for problems with fair-size datasets
Outline

Previous Kernel Learning Methods

Kernel Learning Using Neural Networks

Ongoing Work
Why Neural Nets

- We want to have a powerful non-linear feature mapping
- We want to make use of the rich structure information existing in the dataset not just labels
- We want an efficient learning approach applicable to large datasets
Learn the Desired Feature Directly

$$\max_K \ell = \frac{\text{Tr}(K_{tr}K_{opt}^T)}{N \sqrt{\text{Tr}(K_{tr}K_{tr}^T)}}$$

subject to $\text{Tr}(K) = 1, K \succeq 0$.

- $K_{tr} = F_{tr}^T F_{tr}$, $F_{tr}$: the feature vectors learned from neural networks for the training data.
- $f$, a column of $F_{tr}$, represents the feature vector learned for one data point.
- Learn the weights $\rightarrow$ Learn the mapping $\rightarrow$ Learn the kernel.
the constraint $Tr(K) = 1$

- To enforce the constraint, we make $f = \frac{z}{\|z\|}$, where $z$ is the linear output vector of an encoder with one logistic hidden layer.
- All the feature vectors lie on the surface of a unit sphere.
- Relaxing this constraint so that some points can lie inside the sphere, we use a logistic unit $r$ to represent the norm of a feature vector.
- Then $f = r \frac{z}{\|z\|}$. 
The Structure of the Encoder

input vector x

r

z
Learn the Weights in the Network

\[
\frac{\partial \ell}{\partial K_{tr}} = \frac{K_{opt} \text{Tr}(K_{tr}K_{tr}^T)^{\frac{1}{2}} - K_{tr} \text{Tr}(K_{tr}K_{opt}^T) \text{Tr}(K_{tr}K_{tr}^T)^{-\frac{1}{2}}}{\text{Tr}(K_{tr}K_{tr}^T)}
\]

\[
\frac{\partial \ell}{\partial f(j)} = \sum_k \frac{\partial \ell}{\partial K_{tr,kj}} f(k) + \sum_k \frac{\partial \ell}{\partial K_{tr,jk}} f(k);
\]

Back Propagation using Stochastic Gradient Descent with adapted learning rates invented by Geoff.
Combined with Unsupervised Learning

- The class information is limited. Might overfit.
- The structure in the original data is rich: put a lot of constraints on the weights.
- Maximizing the Kernel Alignment objective + Reconstructing the original data vectors.
- Autoencoder!
- As in [Hinton and Salakhutdinov, 2006] and its following work, make some components in the code (feature) vector ONLY participate in reconstruction.
The Structure of the autoencoder

reconstructed
x

z
z'

used for reconstruction only

input vector x
Old Results on Handwritten Digit Classification

- Dataset 1: 1100 8s (600 for training, 500 for testing) and 1100 9s (600 for training, 500 for testing)
- Dataset 2: 1100 4s (600 for training, 500 for testing) and 1100 6s (600 for training, 500 for testing)
- Old Results:

<table>
<thead>
<tr>
<th>Kernels</th>
<th>Gaussian Kernel</th>
<th>NN Ball Surface</th>
<th>NN Sphere</th>
<th>Auto</th>
<th>Auto-RBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataset1(1000)</td>
<td>11</td>
<td>9</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>dataset2(1000)</td>
<td>13</td>
<td>12</td>
<td>7</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

The number of errors is out of 1000. Here, in the final 50 iterations of the training, we only minimize the kernel alignment cost.
Extensions to Multi-Class Classification

Define the optimal kernel as follows:

\[
K_{opt}(i, j) = \begin{cases} 
  +1 & \text{if } i \text{ and } j \text{ are in the same class or } i = j \\
  -1 & \text{otherwise}; 
\end{cases}
\]  

(2)

Still maximize the Kernel Alignment Objective.

Use one-vs-the-rest SVM \(k\) times or use multi-class SVM. \(k\): the number of classes.
Outline

Previous Kernel Learning Methods

Kernel Learning Using Neural Networks

Ongoing Work
Train the model on MNIST to do multi-class classification (the binary classification task is too easy).

Learn an Autoencoder with 4 hidden layers using stacked RBM stead of only using RBM to learn the first hidden layer.

Relax the $Tr(K) = 1$ constraint by using logistic units for the feature vector.
Work in progress

- deal with the dual of SVM directly without minimizing kernel alignment cost
- coordinate optimization: iterate between optimizing the dual parameters and the weights in the neural networks
Optimization in the dual

\[ \min_w \max_\alpha \sum_i \alpha_i - \sum_{ij} \frac{1}{2} \alpha_i \alpha_j f_i^T f_j \]

s.t. \( 0 \leq \alpha_i \leq C, i, j = 1, \ldots, n. \)

- Use log-barrier method to change the constrained optimization to an unconstrained optimization
- annealing the log-barrier coefficient.
- coordinate optimization (current implementation is stochastic gradient-based. Conjugate-Gradient and SMO can be used here.).
The End

Thank you!