Semi-supervised learning and active learning

Le Song

Machine Learning II: Advanced Topics
CSE 8803ML, Spring 2012
Combining classifiers

- Ensemble learning: a machine learning paradigm where multiple learners are used to solve the problem

- The generalization ability of the ensemble is usually significantly better than that of an individual learner
Bagging

- Bagging: Bootstrap aggregating
  - Generate B bootstrap samples of the training data: uniformly random sampling with replacement
  - Train a classifier or a regression function using each bootstrap sample
  - For classification: majority vote on the classification results
  - For regression: average on the predicted values

<table>
<thead>
<tr>
<th>Original</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set 1</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Training set 2</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Training set 3</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Training set 4</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>
Stacking classifiers

- Level-0 models are based on different learning models and use original data (level-0 data)

- Level-1 models are based on results of level-0 models (level-1 data are outputs of level-0 models) -- also called “generalizer”

- If you have lots of models, you can stacking into deeper hierarchies
Adaboost flow chart

Original training set

Data set \(_1\)  \rightarrow  Data set \(_2\)  \rightarrow  \ldots \ldots  \rightarrow  Data set \(_T\)

Learner\(_1\)  \rightarrow  Learner\(_2\)  \rightarrow  \ldots \ldots  \rightarrow  Learner\(_T\)

Weighted combination

Training instances that are wrongly predicted by Learner\(_1\) will play more important roles in the training of Learner\(_2\).
AdaBoost

- constructing $D_t$:
  - $D_1(i) = 1/m$
  - given $D_t$ and $h_t$:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \left\{ \begin{array}{ll}
e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\
e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{array} \right.$$ 

$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i h_t(x_i))$$

where $Z_t = \text{normalization constant}$

$$\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right) > 0$$

- final classifier:

$$H_{\text{final}}(x) = \text{sign} \left( \sum_t \alpha_t h_t(x) \right)$$
Boosting round 3

- Obtain weak learning, and then reweight data
- Now we have 3 classifiers

\[ h_3 \]

\[ \varepsilon_3 = 0.14 \]
\[ \alpha_3 = 0.92 \]
Boosting aggregate classifier

- Final classifier is weighted combination of weak classifiers

\[ H_{\text{final}} = \text{sign} \left( 0.42 \times 0.65 + 0.92 \right) \]
How will train/test error behave?

- Expect: training error to continue to drop (or reach zero)
- First guess: test error to increase when the continued classifier becomes too complex
  - Overfitting: hard to know when to stop training
Actual experimental observation

- Test error does not increase, even after 1000 rounds

- Test error continues to drop even after training error is zeros!

<table>
<thead>
<tr>
<th># rounds</th>
<th>5</th>
<th>100</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>train error</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>test error</td>
<td>8.4</td>
<td>3.3</td>
<td>3.1</td>
</tr>
</tbody>
</table>
Labeled data can be rare or expensive

- Need to pay someone to do it, requires special testing ...

- Unlabeled data is much cheaper

Can we make use of cheap unlabeled data?
- Unlabeled data is missing the most important information
- But maybe still has useful regularities that we can use.

Three semi-supervised method
- Co-training
- Semi-Supervised (Transductive) SVM [Joachims98]
- Graph-based methods
Many problems have two different sources of info you can use to determine label.

E.g., classifying webpages: can use words on page or words on links pointing to the page.
Co-training

- Then look for unlabeled examples where one rule is confident and the other is not. Have it label the example for the other.

- Training 2 classifiers, one on each type of info. Using each to help train the other.
Semi-Supervised SVM (S\(^3\)VM)

- Suppose we believe decision boundary goes through low density regions of the space/large margin.

- Aim for classifiers with large margin wrt labeled and unlabeled data. (L+U)
Unfortunately, optimization problem is now NP-hard. Algorithm instead does local optimization.

- Start with large margin over labeled data. Induces labels on U.
- Then try flipping labels in greedy fashion.
- Or, branch-and-bound, other methods (Chapelle et al. 06)

Quite successful on text data.
Graph-based methods

- Suppose that very similar examples probably have the same label

- If you have a lot of labeled data, this suggests a Nearest-Neighbor type of algorithm

- If you have a lot of unlabeled data, perhaps can use them as “stepping stones”

E.g., handwritten digits [Zhu07]:

<table>
<thead>
<tr>
<th>2</th>
<th>2 2 2 2 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>not similar</td>
<td>‘indirectly’ similar with stepping stones</td>
</tr>
</tbody>
</table>
Graph-based methods

- Idea: construct a graph with edges between very similar examples.

- Unlabeled data can help “glue” the objects of the same class together.
Graph-based methods

- Idea: construct a graph with edges between very similar examples
- Unlabeled data can help “glue” the objects of the same class together
- Suppose just two labels: 0 & 1. Solve for labels $f(x)$ for unlabeled examples $x$ to minimize:
  - Label propagation: average of neighbor labels
  - Minimum cut $\sum_{e=(u,v)} |f(u) - f(v)|$
  - Minimum “soft-cut” $\sum_{e=(u,v)} (f(u) - f(v))^2$
  - Spectral partitioning
Semi-Supervised Learning

Data

Supervised

Semi-Supervised
SSL using Graph Laplacian

- Want to find label function $f$ that minimizes:

$$J(f) = f^T L f + \sum_{i=1}^{l} \lambda(f(i) - y_i)^2$$

- Solution: $$(L + \Lambda)f = \Lambda y$$

\(n \times n\) system \((n = \#\text{ points})\)
Passive Learning (Non-sequential Design)

Learning Algorithm (estimator)

Data Source

Expert / Oracle

Raw Unlabeled Data

Labeled data points

Algorithm outputs a classifier
Active Learning (Sequential Design)

Learning Algorithm

Data Source

Expert / Oracle

Raw Unlabeled Data

Request for the label of a data point
The label of that point

Request for the label of another data point
The label of that point

Algorithm outputs a classifier
Active Learning (Sequential Design)

Learning Algorithm

Data Source

Raw Unlabeled Data

Request for the label of a data point

The label of that point

Request for the label of another data point

The label of that point

...

Algorithm outputs a classifier

Expert / Oracle

How many label requests are required to learn?
Label Complexity
Support Vector Machines (SVM)

\[
\begin{align*}
\text{min} & \quad \frac{1}{2} w^\top w + C \sum_j \xi_j \\
\text{s.t.} & \quad (w^\top x_j + b) y_j \geq 1 - \xi_j, \quad \xi_j \geq 0, \quad \forall j
\end{align*}
\]

\(\xi_j\): Slack variables
SVM dual problem

- Plug in $w$ and $b$ into the Lagrangian, and the dual problem
  \[
  \text{Max}_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j
  \]
  \[
  \text{s.t. } \sum_i \alpha_i y_i = 0
  \]
  \[
  0 \leq \alpha_i \leq C
  \]

- It is a quadratic programming; solve for $\alpha$, then we get
  \[
  w = \sum_j \alpha_j y_j x_j
  \]
  \[
  b = y_k - w^T x_k \text{ for any } k \text{ such that } 0 < \alpha_k < C
  \]
  \[
  x_i^T x_j = \sum_l x_{il} x_{jl}
  \]
  \[
  \phi(x_i)^T \phi(x_j) = \sum_l \phi(x_i)_l \phi(x_j)_l
  \]

- Data points corresponding to nonzeros $\alpha_i$ are called support vectors
Version space

- Given a set of labeled training data, there is a set of hyperplanes that separate the data.

- We call this set of consistent hypotheses the version space

\[ V = \{ \mathbf{w} \in \mathcal{W} \mid \|\mathbf{w}\| = 1, \quad y_i (\mathbf{w} \cdot \Phi(x_i)) > 0, \quad i = 1 \ldots n \} \]

\[ V_i^- = V_i \cap \{ \mathbf{w} \in \mathcal{W} \mid -(\mathbf{w} \cdot \Phi(x_{i+1})) > 0 \} \]

\[ V_i^+ = V_i \cap \{ \mathbf{w} \in \mathcal{W} \mid +(\mathbf{w} \cdot \Phi(x_{i+1})) > 0 \} \]
Active learning for SVM

- We wish to reduce the version space as fast as possible

- Intuitively, one good way of doing this is to choose a query that halves the version space

- Given an unlabeled instance $x$ from the pool, it is not practical to explicitly compute the sizes of the new version spaces $V_-$ and $V_+$ (i.e., the version spaces obtained when $x$ is labeled as $-1$ and $+1$ respectively). Three approximating scheme:
  - Simple margin
  - MaxMin margin
  - Ratio Margin
Each data point has a corresponding Hyperplane

How close this hyperplane is to will tell us how much it bisects the current version space

Choose x close to w
Simple margin

- If $V_i$ is highly non-symmetric and/or $w_i$ is not centrally placed, the result might be ugly.
MaxMin Margin

- Use the fact that an SVMs margin is proportional to the resulting version space’s area

- The algorithm: for each unlabeled point compute the two margins of the potential version spaces $V^+$ and $V^-$. Request the label for the point with the largest $\min(m^+, m^-)$
MaxMin Margin

- A better approximation of the resulting split

- Both MaxMin and Ratio (coming next) computationally more intensive than Simple

- But can still do slightly better, still without explicitly computing the areas
Ratio Margin

- Similar to MaxMin, but considers the fact that the shape of the version space might make the margins small even if they are a good choice.

- Choose the point with the largest resulting

\[
\min\left(\frac{m^-}{m^+}, \frac{m^+}{m^-}\right)
\]

- Seems to be a good choice.
Experiments

- Text document classification
  - Reuters Data Set, around 13000 articles
  - Multi-class classification of articles by topics
  - Around 10000 dimensions (word vectors)
  - Sample 1000 unlabeled examples, randomly

- Choose two for a start
  - Polynomial kernel classification
  - Active Learning: Simple, MaxMin & Ratio
  - Articles transformed to vectors of word frequencies ("bag of words")
## Results after ten queries

- Binary classifier, one topic against the rest

<table>
<thead>
<tr>
<th>Topic</th>
<th>Simple</th>
<th>MaxMin</th>
<th>Ratio</th>
<th>Equivalent Random size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earn</td>
<td>86.39 ± 1.65</td>
<td>87.75 ± 1.40</td>
<td>90.24 ± 2.31</td>
<td>34</td>
</tr>
<tr>
<td>Acq</td>
<td>77.04 ± 1.17</td>
<td>77.08 ± 2.00</td>
<td>80.42 ± 1.50</td>
<td>&gt; 100</td>
</tr>
<tr>
<td>Money-fx</td>
<td>93.82 ± 0.35</td>
<td>94.80 ± 0.14</td>
<td>94.83 ± 0.13</td>
<td>50</td>
</tr>
<tr>
<td>Grain</td>
<td>95.53 ± 0.09</td>
<td>95.29 ± 0.38</td>
<td>95.55 ± 1.22</td>
<td>13</td>
</tr>
<tr>
<td>Crude</td>
<td>95.26 ± 0.38</td>
<td>95.26 ± 0.15</td>
<td>95.35 ± 0.21</td>
<td>&gt; 100</td>
</tr>
<tr>
<td>Trade</td>
<td>96.31 ± 0.28</td>
<td>96.64 ± 0.10</td>
<td>96.60 ± 0.15</td>
<td>&gt; 100</td>
</tr>
<tr>
<td>Interest</td>
<td>96.15 ± 0.21</td>
<td>96.55 ± 0.09</td>
<td>96.43 ± 0.09</td>
<td>&gt; 100</td>
</tr>
<tr>
<td>Ship</td>
<td>97.75 ± 0.11</td>
<td>97.81 ± 0.09</td>
<td>97.66 ± 0.12</td>
<td>&gt; 100</td>
</tr>
<tr>
<td>Wheat</td>
<td>98.10 ± 0.24</td>
<td>98.48 ± 0.09</td>
<td>98.13 ± 0.20</td>
<td>&gt; 100</td>
</tr>
<tr>
<td>Corn</td>
<td>98.31 ± 0.19</td>
<td>98.56 ± 0.05</td>
<td>98.30 ± 0.19</td>
<td>15</td>
</tr>
</tbody>
</table>
Performance improvement
Active learning with label propagation

(1) Build neighborhood graph

(2) Query some random points

(3) Propagate labels

(4) Make query and go to (3)
Exploiting cluster structure

- Find a clustering of the data
- Sample a few randomly-chosen points in each cluster
- Assign each cluster its majority label
- Now use this fully labeled data set to build a classifier
Finding the right granularity

Unlabeled data

Find a clustering

Ask for some labels

Refine the clustering

Now what?
Using a hierarchical clustering

- Always work with some pruning of the hierarchy: a clustering induced by the tree. Pick a cluster (intelligently) and query a random point in it.

For each tree node (i.e. cluster) v maintain: (i) majority label $L(v)$; (ii) empirical label frequencies $bp_{v,l}$; and (iii) confidence interval
Activized Learning

“Activizer” Meta-algorithm

Data Source

Raw Unlabeled Data

Request for the label of a data point

The label of that point

Request for the label of another data point

The label of that point

...Algorithm outputs a classifier

Expert / Oracle

Passive Learning

Algorithm

(Supervised / Semi-Supervised)
Activized Learning

“Activizer” Meta-algorithm

Data Source

Expert / Oracle

Raw Unlabeled Data

Request for the label of a data point
The label of that point
Request for the label of another data point
The label of that point

Algorithm outputs a classifier

Passive Learning Algorithm
(Supervised / Semi-Supervised)

Are there general-purpose activizers that strictly improve the label complexity of any passive algorithm?
A simple activizer for any threshold-learning algorithm.
An Example: Threshold Classifiers

A simple activizer for any threshold-learning algorithm.

Take \( n/2 \) unlabeled data points, request their labels
Locate the closest \(-/+\) points: \( a,b \)
Estimate \( P([a,b]) \), and sample \( \approx n/(4P([a,b])) \) unlabeled data points
Request the labels in \([a,b]\)
Label rest ourselves.

Train passive alg on all examples.

Used only \( n \) label requests,
but get a classifier trained on \( \Omega(n^2) \) data points!

Improvement in label complexity over passive.
(in this case, apply idea sequentially to get exponential improvement)