Reading Review

Shawn Tice
shawn.cameron.bird@gmail.com

22 October 2012
Class**ification** is the task of assigning an **observation** to one of two or more **categories** in a manner consistent with a **training set** of observations whose categories are known.

An observation is characterized by a collection of **features**, which may be **categorical**, **integer-valued**, or **real-valued**.
For Example

<table>
<thead>
<tr>
<th>Cap Shape</th>
<th>Cap Color</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>Brown</td>
<td>Edible</td>
</tr>
<tr>
<td>Convex</td>
<td>Yellow</td>
<td>Poisonous</td>
</tr>
<tr>
<td>Bell</td>
<td>White</td>
<td>Edible</td>
</tr>
</tbody>
</table>

Classification is a **supervised learning** problem. The **unsupervised** equivalent is clustering.
Applications

- Optical Character Recognition
- Speech Recognition
- Diagnosis
- Document Classification
- ...
Features of Classification Algorithms

Expressive
What kinds of clusters can the classifier separate? For example, linear versus polynomial.

Robust to Noise
Does noise in the training data significantly degrade performance?

Robust to Feature Choice
Does choosing irrelevant features degrade performance?

High Dimensionality
Does a large number of dimensions degrade performance?
Features of Classification Algorithms

Globally Optimal
Does the training method result in an optimal classifier for the training set?

Efficient Training
Is training the classifier on a large training set efficient?

Efficient Classification
Can the classifier efficiently classify an observation?
Induce a tree of predicates to iteratively divide the feature space until a single class is reached.

<table>
<thead>
<tr>
<th>Expressive</th>
<th>Robust to Noise</th>
<th>Robust to Feature Choice</th>
<th>Many Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectilinear</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Globally Optimal</td>
<td>Efficient Training</td>
<td>Efficient Classification</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
**Nearest-Neighbor**

Assign the majority class of the $k$ nearest neighbors in the training set according to some distance measure.

<table>
<thead>
<tr>
<th>Expressive</th>
<th>Robust to Noise</th>
<th>Robust to Feature Choice</th>
<th>Many Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Globally Optimal</td>
<td>Efficient Training</td>
<td>Efficient Classification</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Bayesian (Naïve)

Assign the class \( y \) to an observation \( x \) that maximizes the posterior probability \( P(y|x) \). Assume the features are conditionally independent given \( y \).

<table>
<thead>
<tr>
<th>Expressive</th>
<th>Robust to Noise</th>
<th>Robust to Feature Choice</th>
<th>Many Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Partially</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Globally</td>
<td>Efficient</td>
<td>Efficient</td>
<td></td>
</tr>
<tr>
<td>Optimal</td>
<td>Training</td>
<td>Classification</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Artificial Neural Network

Construct a network of input, hidden, and output nodes, where hidden nodes weight, aggregate, and filter features from input nodes, and output nodes assign a class.

<table>
<thead>
<tr>
<th>Expressive</th>
<th>Robust to Noise</th>
<th>Robust to Feature Choice</th>
<th>Many Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Globally Optimal</td>
<td>Efficient</td>
<td>Efficient</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Training</td>
<td>Classification</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Support Vector Machine

Induce a maximal margin hyperplane that divides observations in high-dimensional space. Observations on one side belong to one class, and observations on the other side belong to another class.

<table>
<thead>
<tr>
<th>Expressive</th>
<th>Robust to Noise</th>
<th>Robust to Feature Choice</th>
<th>Many Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Globally Optimal</td>
<td>Efficient Training</td>
<td>Efficient Classification</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Multiple Classes

Some classifiers (e.g. SVMs) only handle binary classification.

One-vs-Rest

Train one classifier per potential class that decides between that class and all other classes. Run all classifiers and pick the one that gets the most votes.

One-vs-One

Train one classifier per combination of two classes.
Multiple Classes

Error Correcting Codes allow for binary classification errors. Assign a unique bit string of length $n$ to each possible class, and train $n$ classifiers to predict each bit.

The class is the codeword whose Hamming Distance is closest to the constructed bit string.

If the Hamming distance between any pair of codewords is $d$ then any $⌊(d − 1)/2⌋$ errors in the output code can be corrected.
Ensemble Methods

Improve classification accuracy by training multiple classifiers and aggregating their outputs.

- Use different parameterizations of the base classifier
- Manipulate the training set (bagging and boosting)
- Manipulate the class labels (error-correcting output coding)
Ensemble Methods

The most intuitive (to me) approach is to train different parameterizations of a base classifier on the entire training set, then run each classifier on an observation and take the weighted vote of all classifiers.

A classifier’s weight is determined by its predicted accuracy.
Ensemble Methods

**Bagging** divides the training set into smaller samples, and trains a new classifier on each sample.

**Boosting** is similar, but observations are assigned a sampling weight which increases or decreases in successive rounds, depending on how a classifier trained on that sample classifies each observation.
Fusion and Metalearning

Summary: Büttcher et al. show that when combining the results of several different learners it’s usually possible to marginally improve the performance of the best single learner.

Two approaches: **Fusion** and **Stacking**.
**Fusion**

Fusion combines multiple lists of results into a single list according to a scoring formula.

Reported improvements (p381) to P@10 and MAP are around ±0.03 (but more often +0.03).

Fusion is designed for improving search results, but could perhaps be applied to soft classification.
Stacking combines the results of several classifiers by transforming their scores for training documents into log-odds, then applying learning (e.g. regression) to those transformed scores.

Reported improvements (p383) are substantial—as much as a +0.13 improvement over the best individual classifier on average precision when using cross validation.
Relevance Feedback

Relevance feedback attempts to improve an initial query by using documents marked by the user as relevant to expand the query.

We can use relevance feedback term selection techniques to build a query from an initial document in order to find other similar documents in an inverted index.

The two basic models are vector-based and probability-based.
Vector-Based Relevance Feedback

The method proposed by Ide (1971), called the Ide-dec-hi formula for modifying a query based on relevance information:

\[ Q_1 = Q_0 + \sum_{i}^{n_r} r_i - s_i \]

where \( s_i \) is the vector for “the first non-relevant document”.

\[ \text{where } s_i \text{ is the vector for “the first non-relevant document”} \]
Büttcher et al. develop a term weight related to IDF and the $F_4$ measure:

$$w_t = \log \frac{(n_{t,r} + 0.5)(N - N_t - n_r + n_{t,r} + 0.5)}{(n_r - n_{t,r} + 0.5)(N_t - n_{t,r} + 0.5)}$$

and use that to derive a term selection value:

$$n_{t,r} \cdot w_t$$
Relevance Feedback

Ruthven and Lalmas report that vector-based RF methods perform better, but don’t actually provide numbers; Bütcher et al. only present the probabilistic approach.

Both parties suggest selecting around 10 terms, but don’t provide much explanation.

Many machine learning algorithms have been successfully applied to text categorization.

In the literature SVMs have consistently outperformed other techniques, but logistic regression works nearly as well and has much more efficient implementations.

Stacking and metalearning techniques can be used to combine several classifiers and have been shown to improve on the best performance of any single constituent classifier.
Plans

Since we’ll be implementing our algorithms in PHP without new C extensions, performance will be more important than achieving maximum accuracy.

So I’ll experiment with algorithms that trade off some potential accuracy for more efficient implementations: Naive Bayes, Logistic Regression, and perhaps one other algorithm if time permits (e.g. a neural network with one or more hidden layers or decision trees).

I’ll also experiment with boosting.
Plans

Because we’ll be bootstrapping the training set, I suspect our learning algorithms may initially perform very poorly.

Thus I may need to investigate feature selection techniques for reducing the size of the feature set (terms).
For relevance feedback, both vector-based and probability-based techniques appear relatively easy to implement.

So I’ll implement both and see which returns more relevant documents on average.
References

Büttcher, S., Clarke, C., and Cormack, G.  
*Information Retrieval: Implementing and Evaluating Search Engines*  

Joachims, T.  
*Text categorization with support vector machines*  
Technical report, LS VIII Number 23  
University of Dortmund, 1997.
Ruthven, I., and Lalmas, M.  
*A survey on the use of relevance feedback for information access systems*  

Tan, Pang-Ning, Steinbach, M., Kumar, V.  
*Introduction to Data Mining*  
Addison Wesley, 2006.