Assignment 3: Decision Trees and Naive Bayes Model

CS486/686 – Fall 2009

Out: October 27, 2009
Due: November 17, 2009

Be sure to include your name and student number with your assignment.

Text categorization

Text categorization is an important task in natural language processing and information retrieval. For instance, news articles, emails or blogs are often classified by topics. In this assignment, you will implement (in the language of your choice) a decision tree algorithm and a naive Bayes model to learn a classifier that can assign a newsgroup topic to any article. On the course webpage, a training set and test set of articles with their correct newsgroup label will be posted within 5 days. To simplify your implementation, these articles have been pre-processed and converted to the *bag of words* model. More precisely, each article is converted to a vector of binary values such that each entry indicates whether the document contains a specific word or not.

1. **[60 pts]** Decision Tree Learning

   Implement a decision tree learning algorithm. Here, each decision node corresponds to a word feature. Use a priority queue to decide which leaf to split at each step with a new word feature. Experiment with priority queues that rank leaves by

   (a) information gain and

   (b) information gain times number of training documents at the leaves

   achieved by the best splitting word feature. Grow the tree one node at a time, up to 100 nodes, by training with the training set only.

   Report the training and testing accuracy (i.e., percentage of correctly classified articles) of each tree (from 1 to 100 nodes) by producing two graphs (one for each type of queue) with two curves each (one curve for training accuracy and one curve for testing accuracy).

   What to hand in:

   - A printout of your code.
   - A printout (or hand drawing) showing two decision trees (one tree for each type of queue) with the first 10 word features selected and their information gain measure (times the number of documents for the second tree).
   - Two graphs (one for each type of queue) showing the training and testing accuracy as the number of nodes increases.
   - Does overfitting occur? If yes, after how many nodes does overfitting occur?
   - A brief discussion regarding the word features selected by the decision tree learning algorithm for each type of queue. In your opinion, did all the word features selected make sense? Which queue is better and why?

2. **[40 pts]** Naive Bayes Model

   Learn a naive Bayes model by maximum likelihood. More precisely, learn a Bayesian network where the root node is the label/category variable with one child variable per word feature. Learn the parameters of the model by maximizing the likelihood of the training set only. Classify documents by computing the label/category with the highest posterior probability $\Pr(label|\text{words in document})$. Report the training and testing accuracy (i.e., percentage of correctly classified articles).

   What to hand in:

   - A printout of your code.
   - A printout listing the 10 most discriminative word features measured by

     $$\max_{word} |\log \Pr(word|label_1) - \log \Pr(word|label_2)|$$

     Since the posterior of each label is multiplied by the conditional probability $\Pr(word|label_i)$, a word feature should be more discriminative when the ratio $\Pr(word|label_1)/\Pr(word|label_2)$ is large and therefore when the difference between $\log \Pr(word|label_1)$ and $\log \Pr(word|label_2)$ is large. In your opinion, are these good word features?
   - Training and testing accuracy (i.e., two numbers indicating the percentage of correctly classified articles for the training and testing set).
   - The naive Bayes model assumes that all word features are independent. Is this a reasonable assumption? Explain briefly.
   - What could you do to extend the Naive Bayes model to take into account dependencies between words?
   - Which approach performs best among decision trees and the naive Bayes model? Explain briefly why.