# Assignment 3: Decision Trees and Naive Bayes Model

CS486/686 – Winter 2015

Out: November 4, 2015
Due: November 16, 2015 at 12pm

**Submit your assignment via LEARN (CS486 site) in the Assignment 3 Dropbox folder.**
**No late assignments will be accepted**

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.

Train and test your algorithms with a subset of the 20 newsgroup dataset (see the course webpage for the data)[1]. More precisely, you will use the documents posted on the `alt.atheism` and `comp.graphics` newsgroup. Your goal is to design a classifier that can assign a newsgroup (`alt.atheism` or `comp.graphics`) to a given article. 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.

Each line of the files trainData.txt and testData.txt are formatted ”docId wordId” which indicates that word wordId is present in document docId. The files trainLabel.txt and testLabel.txt indicate the label/category (1=`alt.atheism` or 2=`comp.graphics`) for each document (docId = line#). The file words.txt indicates which word corresponds to each wordId (denoted by the line#). If you are using Matlab, the file loadScript.m provides a simple script to load the files into appropriate matrices. At the Matlab prompt, just type ”loadScript” to execute the script. Feel free to use any other language and to build your own loading script for the data if you prefer.

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

   Implement a decision tree learning algorithm. Here, each decision node corresponds to a word feature, and the leaf nodes correspond to a prediction of what newgroup the article belongs in. You will experiment with two feature selection mechanisms as follows:

   (a) average information gain (evenly weighted across the leaves)

   $$I_G = I(E) - [\frac{1}{2} * I(E_1) + \frac{1}{2} * I(E_2)]$$

   (b) information gain weighted by the fraction of documents on each side of the split, as we discussed in class

   $$I_G = I(E) - [\frac{N_1}{N} I(E_1) + \frac{N_2}{N} I(E_2)]$$

   where $I(E)$ is the information content in the $N$ examples before the split, and $I(E_i)$ is the information content of the $N_i$ examples in the $i^{th}$ leaf after the split. The first method does not deal well with splits that put a lot

[1] you can also find out more about the dataset at `http://people.csail.mit.edu/jrennie/20Newsgroups/`, but **use the files provided on the course webpage for the assignment**

of examples on one side, and very few on the other. For example, suppose you have evenly distributed data across the target class (so $I(E) = 1$), and $N$ examples, and 2 features. The first feature splits one example off from the other $N - 1$, so has an information gain using method (b) of $I(E_1) = 0$ and $I(E_2) \approx 1$ so $I_G \approx 1 - [\frac{1}{N} * 0 + \frac{N-1}{N} * 1] = \frac{1}{N}$. Method (a), on the other hand, will make $I_G$ look better than it should be, since it will compute $I_g \approx 1 - [\frac{1}{2} * 0 + \frac{1}{2} * 1] = \frac{1}{2}$. Use $I(\{\}) = 1$ (the entropy of the empty set is maximal).

As discussed in class, build each decision tree by maintaining a priority queue of leaves, sorted by the maximum value of the feature (the information gain of the best performing next feature to split on). At each iteration, split the leaf at which the split will give the largest information gain (according to each of the measures above), and do the split on the word feature that gives that highest gain. Grow the tree one node at a time, up to 100 nodes, by training with the training set only. The point estimate should be the majority class at the leaf.

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 feature selection method) 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 feature selection method) with the first 10 word features selected and their information gain measure. Show the prediction at each leaf.

  Each printed or drawn tree should have 10 nodes and 11 leaves.
- Two graphs (one for each type of queue) showing the training and testing accuracy as the number of nodes increases.

2. **[40 pts]** Naïve Bayes Model

Learn a naïve 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. The word variables should be binary and represent whether that word is present or absent in the document. Learn the parameters of the model by maximizing the likelihood of the training set only. This will set the class probability to the fraction of documents in the training set from each category, and the probability of a word given a document category as the fraction of documents in that category that contain that word. You should use a *Laplace correction* by adding 1 to numerator and 2 to the denominator, in order to avoid situations where both classes have probability of 0. 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 or small and therefore when the absolute 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 naïve Bayes model assumes that all word features are independent. Is this a reasonable assumption? Explain briefly.
- What could you do to extend the Naïve Bayes model to take into account dependencies between words?