Assignment 3: Decision Trees and Naive Bayes Model

CS486/686 – Spring 2017

Out: June 19, 2017
Due: June 30 (11:59 pm), 2017.

Submit an electronic copy of your assignment via LEARN. Late submissions incur a 2% penalty for every rounded up hour past the deadline. For example, an assignment submitted 5 hours and 15 min late will receive a penalty of ceiling(5.25) * 2% = 12%.

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

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 a decision tree algorithm and a naive Bayes model in Python to learn classifiers that can assign a newsgroup topic to any article. Download a training set and test set of articles with their correct newsgroup label from the course website. 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. Once your implementation is complete verify its correctness by comparing your results to those obtained by the sklearn.tree.DecisionTreeClassifier and sklearn.naive_bayes.BernoulliNB classifiers available in the Python scikit-learn library (www.scikit-learn.org) for machine learning.

1. **[50 pts]** Decision Tree Learning

   Implement a decision tree learning algorithm. Here, each decision node corresponds to a word feature, which is selected by maximizing the information gain. Instead of designing a decision tree learning algorithm that builds a full tree, which may overfit, design your algorithm to take as input a maximum depth. Experiment with your algorithm by building trees with increasing maximum depth until a full tree is obtained. Report the training and testing accuracy (i.e., percentage of correctly classified articles) of each tree by producing a graph with two curves (one curve for training accuracy and one curve for testing accuracy as a function of the maximum depth). Report also the tree that achieved the highest testing accuracy. Verify the correctness of your decision tree by comparing your results with those obtained by sklearn.tree.DecisionTreeClassifier(criterion='entropy'). If your code is incomplete or it is not working well, answer the questions below based on the results obtained with sklearn.tree.DecisionTreeClassifier(criterion='entropy') to earn up to two thirds of the marks.

   What to hand in:

   - A printout of your code.
   - A graph showing the training and testing accuracy as the maximum depth increases.
   - Does overfitting occur? If yes, after what maximum depth does overfitting occur?
   - A printout showing the decision tree that achieved the highest testing accuracy. At each leaf, show the class and at each internal node, show the word feature with its information gain.
   - A brief discussion of the word features selected by the decision tree that achieved the highest testing accuracy. In your opinion, did all the word features selected make sense?

2. **[50 pts]** Naive Bayes Model

Learn a naive Bayes model by maximum likelihood learning with Laplace smoothing (also known as add-one smoothing). 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 (with Laplace smoothing) 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). Verify the correctness of your naive Bayes model by comparing your results with those obtained by sklearn.naive_bayes.BernoulliNB(alpha=1,binarize=None). If your code is incomplete or it is not working well, answer the questions below based on the results obtained with sklearn.naive_bayes.BernoulliNB(alpha=1,binarize=None) to earn up to two thirds of the marks.

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. For each word feature, report $|\log \Pr(word|label_1) - \log \Pr(word|label_2)|$. In your opinion, are these good word features?
- A printout of the 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 the decision tree and the naive Bayes model? Explain briefly why.