CS630 Representing and Accessing Digital Information
Fall 2004
Assignment 2

Due at the beginning of class, on Wednesday, October 6

Part 1: Naïve Bayes is a Linear Classifier (20 Points)

Prove that the decision boundary of Naïve Bayes (as derived in class) is a hyperplane. This means that the decision boundary takes the form $w^T x + b = 0$ so that all examples that lie on one side of the hyperplane are classified as positive, and all examples on the other side are classified as negative. Derive how w and b are computed in Naïve Bayes.

Part 2: Design Choices in Text Classification (60 Points)

The goal of this part is to get experience with machine learning methods for text classification. You will use two machine learning methods to learn text classification rules. The following is provided:

- Dataset of abstracts from the arXiv pre-print database (see www.arxiv.org). It is available from the WWW page. The arXiv is on online database containing scientific articles from physics and computers science. The data is split into a training set of 29890 training documents (file arxiv_doc.train) and a test set (file arxiv_doc.test) of 32487 documents. Each document has a unique document ID. The data is classified into 15 classes (e.g. computer science, high-energy physics, quantum physics, etc.). The classes are the top-level of the hierarchy on the page www.arxiv.org. The class assignments are given in the files arxiv_classes.train and arxiv_classes.test. Each line in the class files starts with the document ID and is followed by all classes the document is in. Note that a document can be in multiple classes (i.e. multi-label).
- SVM-light training algorithm for support vector machines. It is available from the WWW page. It compiles on pretty much all platforms that support gcc and binaries are available for most platforms. How to use SVM-light is explained on the WWW page. SVM-light consists of two executables. "svm_learn" trains the SVM for a training set and outputs a model. "svm_classify" reads the model and classifies a test set (i.e. it outputs the signed distance from the hyperplane, which can be used as a measure of confidence). You can specify the value of C using the option –c <value> in "svm_learn". If you do not specify C, the default value is 1/(average Euclidian length of the training vectors). For this project you can probably ignore all other options of "svm_learn".
- You can make use of any other software tools you can find. In particular, using the parsing and the evaluation part of Lemur should make things easier than programming everything from scratch. Report in your write-up which software tools you used.

Learn a binary text classification rules for each of the 15 arXiv classes. As learning methods, use

- SVM-light

- One other learning method (e.g. naïve Bayes, Rocchio, K-NN, Decision Tree, or any other method you think might work well).

You will have to make design choices, for example:

- How to tokenize
- Use of term weighting
- Use of document length normalization
- Use of feature selection
- Use of stemming
- Use of stopword removal
- Choice of algorithm dependent parameters (e.g. for SVM the parameter C)
- Etc., feel free to be creative

What to turn in:

- Since it is impossible to explore the whole space of all different design choices, you will have to keep most of them fixed. However, pick at least two design parameters for each learning method and vary them. Explain, why you picked these two parameters to vary.
- Report the performance on the test set. Use the precision/recall-break-even point as a performance measure and report separate scores for each of the 15 classes.
- Analyze and discuss how the parameter variations influence the results.
- Analyze and discuss the differences between the methods.
- Explain how you built your system and analyze it (e.g. where are bottlenecks, what takes most of the runtime). If you had to do it again, what would you do differently and where do you expect improvements?

Part 3: Human Intelligence vs. Stupid Statistics (20 Points + 5 Bonus)

One of the classes in the arXiv data is "computer science" (indicated by "cs"). Let's see how well you, as human experts for computer science, can write a classification rule that finds all computer science documents (i.e. a binary classification task of cs vs. non-cs).

What to turn in:

- Setting 1: Write the classification rule without looking at any data. Essentially, anything goes that does not involve verifying your classification rule against data (i.e. the training set, the test set, or other data). Evaluate your rule on the test set (using precision and recall) and send it to me via email (tj@cs.cornell.edu).
- Setting 2: You are allowed to tune your classification rule by verifying its performance against the training set. Again, everything goes, except using machine learning and except looking at the test set. After you are satisfied with your rule (or you got too tired of fiddling around with it), evaluate your rule on the test set (using precision and recall) and send it to me via email. Who has the rule with the highest F1-score gets a bonus in the assignment grade and a super-cool Google t-shirt!
- For both settings, document what you did and how you proceeded. Briefly explain your reasoning when making each design decision. Discuss the results in comparison to the machine learning methods. Include your rules in the write-up.