CS 194-10, Fall 2011
Assignment 4

1. *Linear neural networks* (20)
   Suppose you had a neural network with linear activation functions. That is, for each unit the output is some constant c times the weighted sum of the inputs.

   (a) Assume that the network has one hidden layer. For a given assignment to the weights $\mathbf{w}$, write down equations for the value of the units in the output layer as a function of $\mathbf{w}$ and the input layer $\mathbf{x}$, without any explicit mention of the output of the hidden layer. Show that there is a network with no hidden units that computes the same function.

   (b) Repeat the calculation in part (a), but this time do it for a network with any number of hidden layers.

   (c) Suppose a network with one hidden layer and linear activation functions has n input and output nodes and h hidden nodes. What effect does the transformation in part (a) to a network with no hidden layers have on the total number of weights? Discuss in particular the case $h \ll n$.

2. *ML estimation of exponential model* (10)
   A Gaussian distribution is often used to model data on the real line, but is sometimes inappropriate when the data are often close to zero but constrained to be nonnegative. In such cases one can fit an *exponential distribution*, whose probability density function is given by

   $$P(x) = \frac{1}{b} e^{-\frac{x}{b}} \ .$$

   Given N observations x_i drawn from such a distribution:

   (a) Write down the likelihood as a function of the scale parameter b.

   (b) Write down the derivative of the log likelihood.

   (c) Give a simple expression for the ML estimate for b.

3. *ML estimation of noisy-OR model* (extra credit)
   The noisy-OR model is defined as follows (Russell and Norvig, Section 14.3): For a Boolean variable X with Boolean parents U_j,

   $$P(X = true \mid u_1, \ldots, u_D) = 1 - \prod_{\{j : U_j = true\}} q_j \ ,$$

   where the product is taken over the parents that are set to true for that row of the CPT. Assuming a data set of N examples, each with values for X and its parents, compute the conditional maximum likelihood solution for the parameters q_j. (I.e., consider only the contribution to the overall likelihood from the noisy-OR model, not the prior on the U_js.) Hint: this may not have a simple answer; you may want to start with $D = 2$.

4. *Naive Bayes models for spam filtering* (70)

   In this assignment you will write your own Naive Bayes spam filter, i.e., a classifier that takes as input an incoming email message and classifies it as positive (**spam**) or negative (not-spam, also referred to as ham). The data you will use for training your classifier (and which we shall use to test it) is a mix of publicly available Enron emails and some authentic spam emails collected by the authors of **this paper**. The emails have been preprocessed by removing some http headers and other standard fields and converting into a sequence of tokens separated by spaces. Each email is in a separate file.

In order to apply a Naive Bayes classifier to an email, the email must be converted into a vector with a fixed set of attributes. Each attribute corresponds to a word or other token; we will consider two commonly used attribute types:

- *Boolean*: 1 if the email contains the token, 0 otherwise.
- *Normalized term frequency* or NTF: the number of times the token occurs in the email, divided by the length (total number of token occurrences) in the email.

Not all tokens need be used to form attributes. For example, if the training set contains 10,000 distinct tokens, one might select a smaller subset of, say, 500 tokens (we'll call these *features*) to define the attributes. Information gain is one possible selection criterion, but others could be used. The selection might vary depending on the attribute type to be used in the classifier.

The basic setup for your code is given in `NBmodel.py`; this includes some stubs to be completed, class definitions for `NaiveBayesModel` and the two subclasses, and a `test` method that scores a trained model against specified test directories of spam and ham.

(a) Implement the munging functions

- `munge_Boolean(email_file,features_file)` and
- `munge_NTF(email_file,features_file)`

that take as input the name of a preprocessed email file and a features file containing a (pickled) list of tokens and return a vector representation of the email. (Although the actual feature selection process will come later, you may want to create a dummy features file now just to test your code.)

(b) Determine appropriate data structures to represent Naive Bayes classifiers of the Boolean and NTF types, where the conditional distribution for an NTF attribute given the class is an exponential distribution as defined in Q.2 above.

(c) Suppose that classifying ham as spam is c times worse than classifying spam as ham, and that a Naive Bayes model returns the probability p that a given message is spam. Explain how to make the ham/spam decision for this message.

(d) Implement Naive Bayes classifier functions

- `NBclassify_Boolean(model,example,cost_ratio)` and
- `NBclassify_NTF(model,example,cost_ratio)`

that take as input a Naive Bayes model, a vector representation of an email, and a value for the cost ratio c, and return 1 if the email should be classified as spam and 0 otherwise. The classifier functions may assume that the model and the example have been created using the same features.

(e) Using a method of your choice, create and submit features files called `Boolean.features` and `NTF.features`, each containing pickled lists of tokens.

(f) Now train your models: write a function for each model type that reads in emails from two given directories (one spam, one ham) and a feature list from a file and outputs a trained model to a specified pickle file. You should submit two model files named `Boolean.model` and `NTF.model`; in your writeup include an estimate of the per-sample cost incurred for incorrect classifications on unseen data. Make sure your code is compatible with the `test` method we have provided; as usual, we will run the classifiers on unseen test data as part of the evaluation.

Turn in all your code, organized into clearly marked sections according to the parts of the assignment. Supply documentation and explanations where appropriate; describe any methods (cross-validation etc.) you used to get good results and estimate costs.

Submit your files collected together as `a4.tar.gz` using `submit a4` as described **here**