10-601 Machine Learning: Assignment 2

- The assignment is due at 3:00pm (beginning of class) on **Monday, February 11, 2008**.
- Since this assignment has an extra problem from last week it will be worth **125 points**
- Write your name at the top right-hand corner of each page submitted.
- Each student must hand in a writeup and their code, where asked. See the course webpage for the code submission and collaboration policies.

Q1: Maximum likelihood estimators [25 pts]

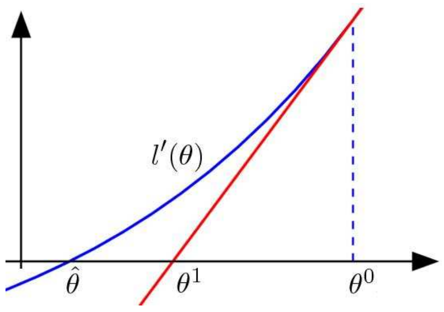
(same as Q4 from HW 1)

1. Let $X_1, X_2, ..., X_n \backsim Uniform(a, b)$ where a and b are parameters, $a < b$, with pdf:

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{if } a \le x \le b \\ 0 & \text{otherwise} \end{cases}$$

   Find the MLE $\hat{a}$ and $\hat{b}$.

2. While MLE's can sometimes be found analytically, for complicated likelihood functions it may need to be computed using numerical methods. One method is the Newton-Raphson algorithm, which iteratively finds a sequence of $\theta^0, \theta^1, ...$ that (under ideal conditions) converges to the MLE $\hat{\theta}$. The idea here is that we are trying to find the value of θ that maximizes the likelihood function. Newton's method allows us to do this by efficiently finding a root for the likelihood function's first derivative by following successively closer tangents of the first derivative function:



   Note that one may expand the derivative of the log-likelihood function around θ^j:

$$0 = l'(\hat{\theta}) \approx l'(\theta^j) + (\hat{\theta} - \theta^j)l''(\theta^j)$$

Solving for $\hat{\theta}$ gives

$$\hat{\theta} \approx \theta^j - \frac{l'(\theta^j)}{l''(\theta^j)}$$

This leads to an iterative scheme where

$$\hat{\theta^{j+1}} = \theta^j - \frac{l'(\theta^j)}{l''(\theta^j)}$$

For the binomial sampling function, pdf $f(x) = \binom{n}{x} p^x (1-p)^{n-x}$, find the MLE using Newton-Raphson, starting with an estimate $\theta^0 = 0.1$, $n = 100$, $x = 8$. Show the resulting θ^j until it reaches convergence ($\theta^j - \theta^j < .01$). (Note that the binomial pdf may be calculated analytically– you may use this to check your answer.) Submit your code.

Q2: Naïve Bayes [50 pts]

For this exercise you will be implementing a multinomial Naïve Bayes classifier to classify various text articles downloaded from the Internet.

1. Download the data and code (`HW2-data.zip`) from the class webpage. This archive contains three folders:

   - `economist`, containing articles from the serious European magazine Economist.com
   - `onion`, containing articles from the not-so-serious American magazine TheOnion.com
   - `code`, containing code, documentation and examples for parsing articles into word-count vectors

   The `economist` articles are further divided into seven subfolders, each containing articles concerning a particular geo-political region: `africa, asia, britain, europe, latin_america, north_america, international`.

2. Convert each article from a long string of text into a vector of word counts. Specifically, for each article, count the number of times each word occurs in that article. For instance, given the string S = `Will tomorrow be as sunny as today?`, and a vocabulary V = {0:will, 1:tomorrow, 2:be, 3:as, 4:sunny, 5:today, 6:pajamas}, each word in our vocabulary occurs in S once, except `3:as`, which occurs twice, and `6:pajamas`, which occurs zero times. Thus, by using our vocabulary of words, V of length w, which assigns an integer index to each word, we can represent our original string S as an length-w vector of integer counts: {1, 1, 1, 2, 1, 1, 0}. (*Code has been provided in the archive to help you parse the articles and write out their word-count vectors to files.*) Notice also that as the size of your vocabulary grows, the length of the feature vector will also increase, independent of the length of the articles. That is, there will be more and more zero entries in the word-count vector. To avoid memory issues, you would be advised to use a sparse representation of these word-count vectors, perhaps by intelligently omitting/compressing the zero-count entries. Also, calculating these counts over all the articles may take a long time (over 20 minutes on high powered machines), so we recommend you do your testing and development over a smaller subset of the data, and only run the full experiment once you have worked out your bugs.

3. Now, having successfully represented each article as a single integer vector of word counts, you will perform two classification experiments:

(a) **Onion vs. Economist**: Assign the class label `econ` to all the Economist articles (regardless of region/subfolder), and assign the class label `onion` to all the Onion articles. Your task is to implement a Multinomial Naïve Bayes classifier to distinguish between these two classes of articles.

(b) **Economist regions**: Now, for each Economist article, assign as a class label the name of the folder it is in. Thus, all articles in the `asia/` folder would be labeled as class `asia`, those in `africa/` as class `africa`, etc. Given this set of labellings, your task is to implement a Multinomial Naïve Bayes classifier to distinguish between these seven classes of articles.

For each experiment above:

i. Evaluate your learned classifier using 10-fold cross validation and report your overall prediction accuracy, along with the precision, recall and F1 of each class. Also, summarize your errors using a confusion matrix.

ii. Analyze your results: Are you surprised (positively or negatively) by the results? What did you expect and why? Examine any systematic errors your classifier makes. Can you come up with a plausible explanation?

iii. Analyze your model: Train on the full set of data and look at the weights assigned to the words in your model. For each class, list the five words the model says one is most likely to observe, given that class. Do you think these words are most representative of the class? If not, list five words you feel are more representative of that class, and explain your choice. Why might these lists of words be different? Are some *types* of words more useful for distinguishing between classes than others? Do the same analysis for the *least* probable words, given the class.

iv. Submit your code and any processed data or models, along with your results.

Notes: To avoid possible computational underflow issues, you may want to do your likelihood calculations in log space, e.g. since $ln(p^n) = n \ \ln(p)$, $p^n = e^{n \ \ln(p)}$. Recall also that $\log(\prod_i p_i) = \sum_i \log(p_i)$. You may also find it helpful to smooth your zero-counts.

Q3: Conditional independence and parameter estimation [25 pts]

Note: Throughout this problem, when asked about estimates, we are concerned with MLE's and not MAP estimates. Assume you are given a training dataset comprised of m = 1,000 binary classed examples (500 in the positive $y = 1$ class, 500 in the negative $y = 0$ class), each consisting of n = 10 binary valued attributes, generated from the following model, M_{indp}, which assumes conditional independence between attributes, given their class:

$$\mathrm{M_{indp}} : \forall i : 1 \le i \le n, \forall x, y \in \{0, 1\} \quad Pr(X_i = x|Y = y) = p^{indp}_{i,x,y}$$

In other words, each example $\langle\langle x_1, x_2, \ldots, x_n\rangle, y\rangle$ is generated by first picking a value y for the class Y, then picking the value x_i of each attribute X_i with probability $p^{indp}_{i,x_i,y}$. Each attribute x_i is thus determined independently of the other attributes. We will also assume that the probability of picking class $Y = 1$ is 0.5, i.e. $P(Y = 1) = P(Y = 0) = 0.5$.

1. How many free parameters, $p^{indp}_{i,x,y}$, does this model have?

2. Now assume you are given a particular instance of such a model, where the parameters are set as follows: $\forall i : p^{indp}_{i,1,1} = .8$ and $p^{indp}_{i,1,0} = .6$ (i.e., the probability of any attribute being set to 1 is 0.8 for a positive example, and 0.6 for a negative example). Assume you are also given a single test example from the **positive** class, $\bar{x}_* = \langle\langle 1,1,0,0,1,1,0,1,1,1\rangle, 1\rangle$.

   What is the probability of the instance $\bar{x}_* = \langle 1,1,0,0,1,1,0,1,1,1\rangle$ being generated given that the class is positive—in other words, what is $Pr(\bar{x}_* | y = 1, M_{indp})$? (*see note in Q2 on avoiding underflow issues*)

3. What is the $Pr(y = 1|\bar{x}_*, M_{indp})$? (I.e., what is the predicted probability that the class is 1 **under the model** defined in 2?)

4. Based on the training data, what is the maximum likelihood estimator $\hat{p}^{indp}_{i,x,1}$ for the model parameter $p^{indp}_{i,x,1}$? What is the MLE $\hat{p}^{indp}_{i,x,0}$ for the model parameter $p^{indp}_{i,x,0}$? Express your answer in terms of *properties of the training data*, not the instantiated model parameters given in (2) above).

5. Now consider a new model, M_{dep}, where *no* assumptions are made regarding the possible dependencies between attributes:

$$\mathrm{M_{dep}} : \forall \bar{x} : \bar{x} \in \{0,1\}^n, \forall y \in \{0,1\} \quad Pr(\bar{X} = \bar{x}|Y = y) = p^{dep}_{\bar{x},y}$$

   In other words, each example $\langle\langle x_1, x_2, \ldots, x_n\rangle, y\rangle$ is generated by first picking a value y for the class Y, then picking an entire vector $\bar{x} = \langle x_1, x_2, \ldots, x_n\rangle$, with the probability of picking that vector given by the parameter $p^{dep}_{\bar{x},y}$.

   How many free parameters, $p^{dep}_{\bar{x},y}$, does this model have? How does this compare to M_{indp}?

6. Let $\bar{x}_*$ refer to the single test example of part (2). Under this new M_{dep}, to find $Pr(Y|\bar{x}_*)$ you first need to estimate $\hat{p}^{dep}_{\bar{x}_*,1}$ and $\hat{p}^{dep}_{\bar{x}_*,0}$. Given that you have 500 training examples of each class generated from $M_{\mathbf{indp}}$, but learned your estimates $\hat{p}^{dep}_{\bar{x}_*,1}$ and $\hat{p}^{dep}_{\bar{x}_*,0}$ over this training data assuming $M_{\mathbf{dep}}$:

   (a) What is the MLE for the parameter $\hat{p}^{dep}_{\bar{x}_*,1}$? (again, express this in terms of properties of the training data.)

   (b) What is the probability that this MLE will be zero? I.e., what is $Pr(\hat{p}^{dep}_{\bar{x}_*,1} = 0)$, where the probability here is taken over different outcomes of the "experiment" of generating the training data from M_{indp}.

   (c) What is $Pr(\hat{p}^{dep}_{\bar{x}_*,0} = 0)$?

   (d) Can $\hat{p}^{dep}_{\bar{x}_*,1} = \hat{p}^{dep}_{\bar{x}_*,0} = 0$? When might this occur and what would it mean? How might you assign a class to $\bar{x}$ in this situation?

Q4: Voted perceptrons [25 pts]

Recall the proof of the voted perceptron's mistake bounds over separable data as demonstrated in class and in the notes. We saw that the number of mistakes, k, was bounded by the radius the examples fall into, R, and the margin between classes, γ, such that $k < \frac{R}{\gamma^2}$. To prove this, we had to make two assumptions:

1. The examples can be separated by some vector u with margin $\gamma > 0$
2. The examples all lie within some radius R of the origin

Describe an algorithm that will produce a sequence of examples that force the perceptron algorithm to make a sequence of mistakes of arbitrary length, m, if either of these assumptions (your choice) is removed (but the data still remain separable).