BE 562: Problem Set 2

Due: Monday, September 30, 2019 at 8 PM

Problem 1. Bayes Rule and Naive Bayes Classifier. (20 pts)

In this problem we will familiarize ourselves with Bayes Rule and the use of Naive Bayes for classification. Assume that the probability of there being rain on any given day is 0.2, and that the probability of getting in a car accident is 0.05.

a) Assume these two probabilities are independent. What is the probability of a rainy day where you get into a car accident? What is the conditional probability of getting into a car accident today, given that you already know today is rainy?

b) From vehicle accident records, assume that we've determined that the probability of it being a rainy day, given that a car accident was observed is 0.33. Using Bayes' Rule and the prior probabilities, what is the conditional probability of getting into a car accident today, given that you already know today is rainy?

c) Why are the conditional probabilities you computed in parts (a) and (b) different?

d) The following table describes features for DNA sequences and the class of corresponding DNA sequence. GC content describes the fraction of Gs and Cs (as opposed to As and Ts) in the DNA sequence. Complexity describes the degree of randomness of the sequence. Repeats are stretches of DNA that are highly repetitive and occur multiple times in a genome. Motifs are sites where transcription factors bind.

GC-Content	Length	Complexity	Class
High	Short	Low	Repeat
High	Short	High	Gene
Medium	Short	Low	Motif
Low	Long	Low	Gene
Low	Short	High	Gene
Medium	Short	High	Motif
Medium	Long	Low	Gene
Low	Short	Low	Repeat
Medium	Short	High	Motif
Medium	Long	Low	Gene

Use the Naive Bayes classifier to predict this gene's class (Show your work!):

GC-Content	Length	Complexity	Class
Medium	Long	Low	?

Problem 2. Bayesian Decision Theory. (20 pts)

For many classification problems, our objective will be more complex than simply trying to minimize the number of misclassifications. Frequently different mistakes carry different costs. For example, if we are trying to classify a patient as having cancer or not, it can be argued that it is far more harmful to misclassify a patient as being healthy if they have cancer than to misclassify a patient as having cancer if they are healthy. In the first case, the patient will not be treated and would be more likely to die, whereas the second mistake involves emotional grief but no greater chance of loss of life. To formalize such issues, we introduce a loss function L_{kj}. This function assigns a loss to the misclassification of an object as class j when the true class is class k. For instance, in the case of cancer classification L_{kj} might look like (where the true class k is along the x axis):

	True Cancer	**True Normal**
Predict Cancer	0	10
Predict Normal	1000	0

Which says that there is a loss of 10 if we misdiagnose a healthy patient as having cancer, but a much bigger loss of 1000 if misdiagnose a patient with cancer as normal.

a) In classification, we do not know the actual class k of a data sample d (a patient in our example). Thus, when computing the loss of predicting class j for sample d, we can only calculate expected loss. Knowing the class priors $P(k)$ and likelihoods $P(d|k)$, derive the equation for the expected loss $E_k[L]$ of assigning sample d to class j.

b) Say, for every data point we wish to choose the class which minimizes the quantity you derived in part (a). For the case where the loss matrix is $L_{kj} = 1 - I_{kj}$ (I_{kj} are the elements of the identity matrix), determine what such a decision rule would do.

c) What is the interpretation of the loss matrix in part (b)?

Problem 3. K-means clustering. (30 pts)
In this problem you will implement the K-means clustering algorithm. You may use any programming language. Include the source code and instructions for how to compile/run your code.

Program Requirements:

- **i.** Take a tab-delimited file with an N × M matrix of N data points with M features each. In other words, each row is a data point and each column is a feature value associated with that data point.
- **ii.** Accept K, the number of clusters, as a parameter.
- **iii.** Output the M-dimensional mean vectors for each of the clusters, one per line.
- **iv.** After the mean vectors, output the assignment for each of the data points in the input file, one per line. The first cluster should have index 0, up to index K – 1. These assignments should match up to the order of the vectors in the input file.

You may make your stopping/convergence condition as simple or complex as you want to make it. For example, your condition could be to stop whenever cluster assignments do not change between iterations, or to stop when the cluster centers don't move more than some threshold amount. It may be prudent to set a maximum number of iterations no matter what your stopping condition is. Note: Use **generate-clusters.py** to generate test data. You will find that the easiest data to cluster will be spherical (unit variance in both dimensions) and well separated (cluster means farther than 2 standard deviations apart). Run the script with no arguments to display its help information. While **generate-clusters.py** will create only two dimensional data points, your algorithm should accept an arbitrary number of dimensions (e.g. up to 22-dimensions in part (d)).

- **a)** Implement the K-means algorithm, following the program requirements listed above.
- **b)** Implement the fuzzy K-means algorithm, following the same program requirements. This can either be a completely separate program, or it can be an option in your original program. Include source code in your write-up.
- **c)** Apply your programs to some synthetic data generated by **generate-clusters.py**. Generate 50 points from 3 Gaussian distributions for a total of 150 data points. To make these distributions "difficult" to cluster, the cluster centers should be within 2 standard deviations of each other, and at least one of the distributions should be elongated, not spherical.
  - **i.** Run both algorithms (K-means and fuzzy K-means) on this data with K = 2, 3 and 4 cluster centers. Discuss what happens when the number of cluster centers is not 3. Include the parameters chosen for data generation.
  - **ii.** To get a visual sense of how well the algorithms did, generate at least one clustering plot for each algorithm. The clustering plot should show a scatter of the data points, and indicate BOTH the "correct" and "predicted" cluster for each point. For example, you might draw each point as a *, +, or according to the correct cluster, and a different color for the predicted clusters. You may also want to plot where the predicted cluster centers are.

d) The K-means algorithm tries to find the cluster centers μ that minimize the objective function

$$\sum_{i=1}^{K} \sum_{x_j \in C_i} \Big(x_j - \mu_i \Big)^2$$

where C_i is the set of points assigned to cluster i. K-means is a greedy algorithm and is not guaranteed to find the global minimum of this objective. In fact, the quality of the resulting solution often depends significantly on your choice of initial cluster centers. For K = 3, repeat your K-means program 100 times with randomized initial cluster centers, and report the best cluster centers. How much variation did you find?

e) Now you will apply your K-means program to some real-world data, described in the paper *The Transcriptional Responses of* Mycobacterium tuberculosis *to Inhibitors of Metabolism* (Boshoff *et al.*, 2004). We provide you with a subset of their data (see **description.txt** file). Each of the 3 data files contain expression data for 3924 genes of the TB bacterium, *Mycobacterium tuberculosis*. The different data files represent responses to different classes of drugs. One class inhibits cell wall synthesis, one blocks translation, and the last blocks iron uptake. For each gene, the authors measured expression for specific drugs within their class. We are interested in clustering the genes to find groups of genes that are coordinately regulated in their response to each drug. This may happen, for example, if there is one gene which acts as a "master switch", regulating various other downstream genes.

- **i.** For each of the 3 classes of drugs, use either of your programs to cluster the genes into K = 150 clusters. Feel free to use a different K if you feel it produces better results. Since plotting high dimensional data is difficult, report some other statistics about your clusters (i.e. such as a bar chart of their sizes or statistics on the variability of the clusters upon rerunning your program with different initial clusters). Since this is real data, there is no "right" answer, so don't worry if your clusters are not as clearly defined.
- **ii.** For each class of drugs, the response of each gene is represented as a length 18-22 vector, called an expression profile. The profile contains the data for the different experimental conditions. The length (l_2 norm) of each expression vector indicates how much the drug changes the gene's expression relative to the control (no drug present). Find the gene that is most sensitive to each drug (what is it?), and then look at what cluster it belongs to. Do a search of http://TBDB.org for some of the genes in each cluster (see **rowNames.txt**). List a few functions that these genes are associated with.
- **iii.** Plot the results of your clustering so you can visualize your cluster centers and cluster assignments. For even more extra credit, see what commonalities you can find between the genes in each cluster (why might they be co-regulated?). Are there conserved promoter motifs? Do they belong to the same operon? This is exploratory work – we do not know the answers!

Problem 4. Expectation Maximization in Gaussian Mixtures. (30 pts)
Recall from lecture that fuzzy K-means is a form of expectation maximization using a mixture of Gaussians. We are going to show that in this problem.

For this problem, we are going to assume that we are given an independent, identically distributed sample of N points $x_1, ..., x_N$. We are told that each x_i is drawn from one of K Gaussian distributions with unit variance but with unknown means μ_k. Consequently each x_i can be associated with a label Y_i from the set $1, ..., K$. But we are not told what these labels are. The Gaussians are equally probable, i.e., the prior probability that a data point is drawn from one of the Gaussians is $1/K$. Our goal is to estimate (1) the means μ_k for each of the K Gaussians, and (2) the probability that each x_i belongs to each Gaussian. To do so, we will use expectation maximization (EM). As you learned in lecture, EM is a re-estimation procedure that – loosely speaking – iterates between estimating the missing labels (Y_i in our case) given the current best parameter values (μ_k in our case, this is the E-step) and then re-estimating the parameters given these estimates by choosing the parameters to maximize the expected log-likelihood of the data and the labels (this is the M-step).

a) Suppose that X_i is sampled from a Gaussian distribution with unit variance: $Xi\ N(\mu_k, 1)$. Write out the equation for $P(X_i = x_i; \mu_k)$. The notation μ_k just means that the probability distribution for X_i uses the parameter μ_k for the mean.

b) Now let's estimate the probability of the unobserved labels. Given a set of K means $\mu = \mu_1, ..., \mu_K$ and a data point x_i, write the equation for $P(Y_i = k|X_i = x_i; \mu)$, using the result from part(a). Hint: Use Bayes Rule.

c) Assume for a moment we knew the labels associated with each point. Write out the equation for $\log P(x_1, ..., x_N,\ y_1, ..., y_N; \mu)$, the log-likelihood of the N points and their labels. We denote the log-likelihood as $L(\mu)$, making clear its dependence on the parameters μ. Notice that $L(\mu)$ is a function of $x_1, ..., x_N,\ y_1, ..., y_N$.

d) However, we do not know the labels. But part (b) did provide us with $P(Y_i = k|X_i = x_i; \mu)$. We can use this to find the expected log-likelihood, $Q(\mu)$. The expected value of a general function $f(x)$ is just $\sum_x f(x)P(x)$. Using your answer from (b) and(c), write out the equation for the expected value of the log-likelihood, $Q(\mu) = E[L(\mu)]$, where the expectation is of the Y variables, using the conditional the probability distribution $P(Y_i = k|X_i = x_i; \mu^{(t)})$. The notation $\mu^{(t)}$ refers to the current best estimate of the parameters. You should consider $\mu^{(t)}$ a constant quantity, provided to you (from the previous iteration). The quantities computed here are the output of the **E-step** of EM. Show your derivation!

Note: Feel free to replace constants that will not influence subsequent parts of this problem with a symbol (e.g. C). Hint: You may want to use the linearity of expectation.

e) During the **M-step** of EM, we wish to choose $\mu = \mu_1, ..., \mu_K$ to maximize the expected log-likelihood generated during the previous E-step. To do so, we need to calculate the derivatives $\partial Q(\mu)/\partial \mu_k$. Derive the equation for these derivatives.

f) To find the μ which maximize the expected log-likelihood, take your answer from part (e), set the derivative $\partial Q(\mu)/\partial \mu_k = 0$, and solve for μ_k. We denote this value of μ as $\mu^{(t+1)}$: these would then be used in the next iteration of EM.

Problem 5. Conjugate Priors. (20 pts)

a) Binomial. Assume a binomial distribution with parameter p:

$$p(N_1) = \binom{N_1 + N_2}{N_1} p^{N_1}(1-p)^{N_2}$$

We want to estimate the parameter p which is $P(outcome\ 1)$. Assume you have a training set of N data points of which N_1 are outcome 1 and N_2 are outcome 2. Derive the maximum likelihood estimator for p given these data.

b) Recall that the beta distribution is the conjugate prior for the binomial is

$$Beta(p|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1}$$

Show that if the prior probability of p follows a beta with parameters (α,β) and the likelihood of p given the data set from (a) follows a binomial, the posterior probability of p given the data set is a beta distribution with parameters ($N_1 + \alpha$, $N_2 + \beta$).

c) Given a posterior distribution of p that follows $Beta(\alpha, \beta)$, derive the expected value of p.

d) Multinomial. Repeat part (a), but this time assume a multinomial distribution instead of a binomial.

$$P(x_1, x_2, ..., x_N|p_1, p_2, ..., p_N) = \frac{N!}{\prod_{i=1}^{N} x_i!} \prod_{i=1}^{N} p_i^{x_i}$$

e) The conjugate prior for a multinomial is the Dirichlet. Derive the form of the posterior probability if the likelihood is a multinomial and the prior is a Dirichlet.

$$Dir(\alpha_1, \alpha_2, ..., \alpha_N)|x_1, x_2, ..., x_N) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{N} p_i^{\alpha_i - 1}$$

f) Derive the expected value for the parameters of a multinomial given a training set and a Dirichlet prior.