10-701 Machine Learning, Spring 2011: Homework 1 Solution

February 1, 2011

Instructions There are 3 questions on this assignment. The last question involves coding. Attach your code to the writeup. Please submit your homework as 3 **separate** sets of pages according to TAs, with your name and userid on each set.

1 Information Gain, KL-divergence and Entropy [Xi Chen, 30 points]

1. When we construct a decision tree, the next attribute to split is the one with maximum mutual information (a.k.a. information gain), which is defined in terms of entropy. In this problem, we will explore its connection to *KL-divergence*. The KL-divergence from a distribution $p(x)$ to a distribution $q(x)$ can be thought of as a distance measure from p to q:

$$KL(p||q) = -\sum_x p(x) \log_2 \frac{q(x)}{p(x)}.$$

If $p(x) = q(x)$, then $KL(p||q) = 0$. Otherwise, $KL(p||q) > 0$.[1]

We can define mutual information as the KL-divergence from the observed joint distribution of X and Y to the product of their marginals:

$$I(X, Y) \equiv KL(p(x, y)||p(x)p(y))$$

(a) Show that this definition of mutual information is equivalent to the one given in class,. That is, show that $I(X, Y) = H(X) - H(X|Y)$ and $I(X, Y) = H(Y) - H(Y|X)$ from the definition in terms of KL-divergence. From this definition, we can easily see that mutual information is symmetric, i.e. $I(X, Y) = I(Y, X)$. [10pt]

(b) According to this definition, under what conditions do we have that $I(X, Y) = 0$. [5pt]

★ **SOLUTION:**

(a)

$$\begin{aligned} KL(p||q) &= -\sum_x \sum_y p(x, y) \log_2 \left(\frac{p(x)p(y)}{p(x, y)} \right) \\ &= -\sum_x \sum_y p(x, y)(\log_2 p(x) + \log_2 p(y) - \log_2 p(x, y)) \\ &= -\sum_y p(y) \log_2 p(y) + \sum_x p(x) \sum_y p(y|x) \log_2 p(y|x) \\ &= H(Y) - H(Y|X) \end{aligned}$$

Equivalence to $H(X) - H(X|Y)$ can be shown in a similar way.

[1] For more details on KL-divergence, refer to Section 1.6 in Bishop.

(b) When X and Y are statistically independent, i.e. $p(x,y) = p(x)p(y)$, $I(X,Y) = 0$.

2. In the class, we define the entropy based on a discrete random variable X. Now consider the case that X is a continuous random variable with the probability density function $p(x)$. The entropy is defined as:

$$H(X) = -\int p(x) \ln p(x) dx$$

Assume that X follows a Gaussian distribution with the mean μ and variance σ^2, i.e.

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp -\frac{(x-\mu)^2}{2\sigma^2}$$

(a) Please derive its entropy $H(X)$. [10pt]

(b) Give a careful observation for the entropy you derived and please indicate one property, which holds for the entropy for (any) discrete random variable, but does not hold here. [5pt]

★ **SOLUTION:**

$$\begin{aligned} 0H(X) &= -\int p(x) \ln p(x) dx \\ &= -\int p(x) \left(-\frac{1}{2} \ln(2\pi\sigma^2) - \frac{(x-\mu)^2}{2\sigma^2} \right) dx \\ &= \frac{1}{2} \left(\ln(2\pi\sigma^2) + \frac{1}{\sigma^2} \int p(x)(x-\mu)^2 dx \right) \\ &= \frac{1}{2} \left(\ln(2\pi\sigma^2) + 1 \right) \end{aligned}$$

The last inequality is according to the variance of a standard normal distribution:

$$\sigma^2 = \text{var}(X) = \mathbb{E}\left((X-\mu)^2\right) = \int p(x)(x-\mu)^2 dx$$

Note that unlike the entropy for discrete variable which is always non-negative, when $\sigma^2 < \frac{1}{2\pi e}$, $H(x) < 0$.

2 Bayes' Rule and Point Estimation [Xi Chen, 30 points]

1. Assume the probability of a certain disease is 0.01. The probability of testing positive given that a person is infected with the disease is 0.95 and the probability of testing positive given the person is not infected with the disease is 0.05.

(a) Calculate the probability of testing positive. [5pt]

(b) Use Bayes' Rule to calculate the probability of being infected with the disease given that the test is positive. [5pt]

★ **SOLUTION:**

(a) Given the information in the problem, we have $P(D) = 0.01$, $P(T|D) = 0.95$ and $P(T|\overline{D}) = 0.05$.

$$\begin{aligned} P(T) &= P(T \wedge D) + P(T \wedge \overline{D}) \\ &= P(T|D)P(D) + P(T|\overline{D})P(\overline{D}) \\ &= 0.95 \times 0.01 + 0.05 \times 0.99 \\ &= 0.059 \end{aligned}$$

(b)

$$P(D|T) = \frac{P(T|D)P(D)}{P(T)} = \frac{0.95 \times 0.01}{0.059} \approx 0.16$$

2. The Poisson distribution is a useful discrete distribution which can be used to model the number of occurrences of something per unit time. For example, in networking, the number of packets to arrive in a given time window is often assumed to follow a poisson distribution. If X is Poisson distributed, i.e. $X \sim Poisson(\lambda)$, its probability mass function takes the following form:

$$P(X|\lambda) = \frac{\lambda^X e^{-\lambda}}{X!},$$

It can be shown that if $\mathbb{E}(X) = \lambda$. Assume now we have n i.i.d. data points from $Poisson(\lambda)$: $\mathcal{D} = \{X_1, \ldots, X_n\}$.

(For the purpose of this problem, you can only use the knowledge about the Poisson and Gamma distributions provided in this problem.)

(a) Show that the sample mean $\hat{\lambda} = \frac{1}{n}\sum_{i=1}^n X_i$ is the maximum likelihood estimate (MLE) of λ and it is unbiased ($\mathbb{E}(\hat{\lambda}) = \lambda$). [8pt]

(b) Now let's be Bayesian and put a prior distribution over λ. Assuming that λ follows a Gamma distribution with the parameters (α, β), its probability density function:

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

where $\Gamma(\alpha) = (\alpha-1)!$ (here we assume α is a positive integer). Compute the posterior distribution over λ. [6pt]

(c) Derive an analytic expression for the maximum a posterior (MAP) of λ under $Gamma(\alpha, \beta)$ prior. [6pt]

★ **SOLUTION:**

(a) Write down the log-likelihood

$$\ln P(\mathcal{D}|\lambda) = \ln e^{-n\lambda}\prod_{i=1}^n \frac{\lambda^{X_i}}{X_i!} = -n\lambda + \sum_{i=1}^n \{X_i \ln\lambda - \ln(X_i!)\}.$$

The MLE $\hat{\lambda} = \arg\max_\lambda P(\mathcal{D}|\lambda) = \arg\max_\lambda \ln P(\mathcal{D}|\lambda)$, which can be obtained by setting the gradient of $\ln P(\mathcal{D}|\lambda)$ with respect to λ to 0. More specifically:

$$\frac{d}{d\lambda}\ln P(\mathcal{D}|\lambda) = -n + \frac{1}{\lambda}\sum_{i=1}^n X_i = 0 \Longrightarrow \hat{\lambda} = \frac{1}{n}\sum_{i=1}^n X_i$$

Since $X_1, \ldots, X_n$ are i.i.d. from $Poisson(\lambda)$, for any X_i, $\mathbb{E}(X_i) = \lambda$. $\hat{\lambda}$ is unbiased because:

$$\mathbb{E}(\hat{\lambda}) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^n X_i\right) = \frac{1}{n}\sum_{i=1}^n \mathbb{E}(X_i) = \frac{1}{n}\sum_{i=1}^n \lambda = \lambda$$

(b)

$$\begin{aligned} p(\lambda|\mathcal{D}) &\propto P(\mathcal{D}|\lambda)p(\lambda) \\ &= e^{-n\lambda}\left(\prod_{i=1}^n \frac{\lambda^{X_i}}{X_i!}\right)\frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda} \\ &\propto \lambda^{\sum_{i=1}^n X_i + \alpha - 1}e^{-n\lambda-\beta\lambda} \end{aligned}$$

Therefore, the posterior distribution $p(\lambda|\mathcal{D}) \sim Gamma(\sum_{i=1}^n X_i + \alpha, n + \beta)$

(c) The MAP $\lambda^* = \arg\max_\lambda p(\lambda|\mathcal{D}) = \arg\max_\lambda \ln p(\lambda|\mathcal{D})$. Since $p(\lambda|\mathcal{D}) \propto \lambda^{\sum_{i=1}^n X_i+\alpha-1} e^{-n\lambda-\beta\lambda}$,

$$\ln p(\lambda|\mathcal{D}) = \left(\sum_{i=1}^n X_i + \alpha - 1\right) \ln \lambda - (n+\beta)\lambda + C,$$

where C is a constant with respect to λ. Take the gradient of $\ln p(\lambda|\mathcal{D})$ with respect to λ and set it to 0:

$$\frac{d}{d\lambda} \ln p(\lambda|\mathcal{D}) = \frac{\sum_{i=1}^n X_i + \alpha - 1}{\lambda} - (n+\beta) = 0 \Longrightarrow \lambda^* = \frac{\sum_{i=1}^n X_i + \alpha - 1}{n+\beta}$$

3 Decision Tree [Yi Zhang & Carl Doersch, 50 points]

In this question, you will write our decision tree code and perform experiments with it. You will observe and discuss the overfitting and post-pruning of the decision trees. Our data is a binary classification data set with discrete attributes, and we only require the decision tree to be able to process this kind of data. All resources are provided in the file hw1_dt.zip, including the training, validation (i.e., pruning) and testing data sets, a partially implemented decision tree codebase in C (with a few core parts removed), and necessary instructions to compile and run the codebase. **Note:** use of this codebase in **not** required. If you are not comfortable with coding in C, feel free to choose any other language to implement your own decision tree, as long as the tree can perform the experiments we require on the particular data set we provided. While this codebase is not a part of the question, we included it so that, hopefully, many of you will be able to avoid the tedious implementation details and focus instead on the interesting parts of the decision tree algorithm.

Building the decision tree: we build the decision tree as we learned in the class. Given the *training set*, we will start from a single root node with all the training examples assigned to it. Then for each node, if the assigned examples are not pure (i.e., not with the same label), we consider further spliting this node using a "best" attribute. Selecting the best attribute for a given node is the most important part for building decision trees, which is achieved by maximizing the information gain of the split, or equivalently minimizing the weighted average entropy after the splitting (i.e., the conditional entropy given the attribute, shown as $H_S(Y|A)$ in page 11 and 12 of the slides). We will stop spliting a node if: 1) the node is pure; or 2) we cannot find any attribute that leads to a positive information gain.

Checking a specific node for pruning: Pruning a node means removing the subtree beneath it, keeping the node as a leaf. As a result, all training examples assigned to the subtree are assigned to this node. Examples assigned to the node may not all have the same label, and in this case the label attached to this node is the label of the majority class (and examples of minority classes in this node are misclassified, and usually the classification accuracy on the *training set* will decrease). For performance reasons, we use a criterion different from the lecture: given a specific node and the *validation set* (i.e., the pruning set), we prune this node if the classification accuracy of the resulting new tree on the validation set improves at least $EPSILON$. $EPSILON$ is the threshold of minimal improvement for pruning. For now, we set $EPSILON$ as 0.005 (i.e., 0.5%), this default value has already been set in our codebase.

Post-pruning a decision tree: top-down and bottom-up. In order to post-prune the entire decision tree, we basically need to perform a tree traversal, and check all the nodes along the traversal. We consider **depth-first traversal**, which can be easily implemented as one function via *recursive calls*. By placing the recursive calls at different locations of the function, we can make **two choices**: 1) check the current node before invoking the recursive calls on its children; 2) invoke the recursive calls on its children before checking the current node. Note that if a node is checked and actually pruned, we will no longer travel to its children. We initially call the traversal function at the root node, and clearly the two choices we mentioned will lead to different orders of checking the tree nodes. We call the first one **the top-down approach** since it checks (and tries to prune) the parent node before recursively checking children, and we call the second one **the bottom-up approach** since it checks children before checking the parent.

Implementation and the C codebase. The C codebase provides a decision tree implementation (with a few parts removed by the TAs) with much more functionality than what we need in this question. So if you decide to use the C codebase, you only need to make changes on a few files (as detailed later) without

really digging into every detail of this codebase. For a quick guide on how to compile and run the codebase, see **quick_start.txt** in the hw1_dt.zip file.

Data files. We use a noisy mushroom data set for this problem. Using this data set, we will train decision trees to classify each mushroom as poisonous or not, using 22 discrete features such as cap shape, cap color and gill size. There are three data files in hw1_dt.zip: noisy10_train.ssv, noisy10_valid.ssv, and noisy10_test.ssv. They are training set, validation set (i.e., pruning set), and testing set. The format of each file is: first three lines are data statistics (number of variables plus label, variable names, properties of each variable), and from the 4th line is the data, where each line is an example and each column is either the label (the first column) or a variable. You don't need to worry about the data format if you use our codebase.

3.1 Complete the implementation [20 points]

To fully implement the decision tree using the C codebase, there are mainly **two places** in the codebase we need to make changes: 1) **entropy.c**: the file implementing and using the entropy function to calculate information gain and choose the best splitting attribute when building the decision tree (search the comment "YOU MUST MODIFY THIS FUNCTION" in this file to find the place to add your code) ; 2) **prune-dt.c**: the file implementing the post-pruning of the tree (search the comment "YOU MUST MODIFY THIS FUNCTION" in this file to find the place to add your code).

Print the code you added in **entropy.c** and **prune-dt.c** and attach to your homework writeup. Note: if you choose to implement your decision tree without using the codebase, just print and attach your code to the writeup.

★ **SOLUTION:** See the function "Entropy" in **entropy.c** and the function "PruneDecisionTree" in **prune-dt.c** from the solution code.

■ **Common mistake 1:** Not checking the boundary condition when calculating the entropy. If a node contains no positive example or no negative examples, we should directly return 0.0 as the entropy instead of attempting to calculate it, i.e., we don't want to compute $\log_2(0)$.

■ **Common mistake 2:** Not converting *int* to *double* before calculating the quotient of two numbers. C is not as smart as Matlab and R: we need to make sure at least either numerator or denominator is a floating point number before computing their division.

3.2 Experiments with different post-pruning strategies [20 points]

We've discussed the top-down and the bottom-up approaches to travel and prune the tree, which you should already implemented in **prune-dt.c**. Run the codebase (or your own implementation) with both approaches, using the training set for building the tree, the validation set for post-pruning, and testing set to finally test the classification accuracy (again, see **quick_start.txt** for compiling and running the codebase).

Report in your homework 1) for the fully grown tree (without post-pruning): the tree size (i.e., the number of nodes and the depth of the tree), the classification accuracy on the training set, and the classification accuracy on the testing set; 2) for the post-pruned tree with top-down approach: the tree size, the classification accuracy on the training set, and the classification accuracy on the testing set; 3) for the post-pruned tree with bottom-up approach: the tree size, the classification accuracy on the training set, and the classification accuracy on the testing set. Note: all the information can be found from the output when we run the codebase. [**10 points**]

Discuss how different pruning approaches affect the size of the tree, training accuracy, and testing accuracy. Also comment on the difference between the training accuracy and the testing accuracy for each different tree (i.e., the full tree and two pruned trees) [**10 points**].

Table 1: Number of nodes as $EPSILON$ changes

	$EPSILON = 0.001$	$EPSILON = 0.005$	$EPSILON = 0.01$	$EPSILON = 0.03$
Top-Down	8	116	704	2040
Bottom-Up	636	681	1303	2040

★ **SOLUTION:** The full-grown tree has 2919 nodes and its depth is 12, with the training accuracy as 99.7% and the testing accuracy as 79.6%. The top-down pruned tree has 116 nodes and its depth is 8, with the training accuracy as 89.6% and the testing accuracy as 89.0%. The bottom-up pruned tree has 681 nodes and its depth is 11, with the training accuracy as 92.2% and the testing accuracy as 88.1%.

The top-down pruning strategy tries to prune higher-level nodes (i.e., those close to the root) before attempting to prune lower-level nodes (i.e., those close to the leaves), so it is a more aggressive pruning strategy and tends to produce smaller post-pruned trees. Since the resulting tree is small, i.e., a less complex model, the training accuracy will generally be lower than that of the full-grown tree (which "overfits" the training samples), but the testing accuracy will usually be higher than that of the full-grown tree as the less complex model generalizes to unseen testing samples better.

The bottom-up pruning strategy tries to prune children nodes before attempting to prune parents, so it is not as aggressive as the top-down strategy and thus will tend to prune less nodes and produce larger post-pruned trees (compared to the top-down pruned trees). As a result, the training accuracy of the resulting tree will generally be higher than the top-down pruned tree (as it is larger and more complex than a top-down pruned tree), but lower than that of a full-grown tree (since the pruned tree is still smaller and thus less complex than the full-grown tree). The testing accuracy of bottom-up pruned tree is usually higher than the full-grown tree (as pruning helps to prevent overfitting). It's difficult to predict which pruned tree will have lower testing accuracy than the other, because both of the following cases could happen: (1) the top-down pruned tree is over-pruned and thus is too simple to get good testing accuracy; (2) the bottom-up pruned tree is not sufficiently pruned and thus still overfit the training samples to certain degree. In our results, the bottom-up pruned tree has slightly lower testing accuracy (88.1%) than that of the top-down pruned tree (89.0%), indicating the case (2) might happen here.

The gap between the training accuracy and the testing accuracy is a good indicator of how much the model overfits the training samples. As we can see, the full grown tree with 2919 nodes has a large gap: 99.7% training accuracy and 79.6% testing accuracy, indicating serious overfitting. The bottom-up pruned tree with 681 nodes has a small gap: 92.2% training accuracy and 88.1% testing accuracy, indicating slight overfitting. The top-down pruned tree with 116 nodes has almost no gap: 89.6% training accuracy and 89.0% testing accuracy, indicating almost no overfitting.

Finally, we want to clarify that, although in this question the smallest tree (i.e., the top-down pruned tree) achieves the best testing accuracy, it is not always the case that the simplest model is the best. Over-simplified model cannot perform well.

3.3 Experiments with different threshold $EPSILON$ [10 points]

When checking each node, we require a minimal improvement of validation accuracy $EPSILON$ for pruning. So far we use the default $EPSILON = 0.005$(i.e, 0.5%). For both top-down and bottom up pruning, change $EPSILON$ and report the number of nodes in the pruned tree for $EPSILON = 0.001, 0.005, 0.01, 0.03$. Briefly explain your results (one or two sentences will suffice).

NOTE: in the codebase, $EPSILON$ is defined in **auxi.h**: search "#define EPSILON 0.005" to find the location of $EPSILON$.

★ **SOLUTION:** See the Table 1 for detailed results. Generally speaking, larger $EPSILON$ will require more improvement of validation accuracy to approve a pruning, so increasing $EPSILON$ will tend to prune less nodes and produce larger post-pruned trees.

Solution for Assignment 1 : Intro to Probability and Statistics, PAC learning

10-701/15-781: Machine Learning (Fall 2004)

Due: Sept. 30th 2004, Thursday, Start of class

Question 1. Basic Probability (18 pts)

- 1.1 (2 pts) Suppose that A is an event such that $Pr(A) = 0$ and that B is any other event. Prove that A and B are independent events.
  Answer:
  Since the event $A \wedge B$ is a subset of the event A, and Pr(A) = 0, so $Pr(A \wedge B) = 0$. Hence $Pr(A \wedge B) = 0 = Pr(A) * Pr(B)$.

- 1.2 (3 pts) Prove: Let $a_1, a_2, ..., a_n$ be possible values of A. Then for any event B, $P(B = b) = \sum_{i=1}^{n} P(B = b|A = a_i) * P(A = a_i)$

  Answer:
  $P(B = b) = P(B = b \wedge (TRUE))$
  $= P(B = b \wedge (A = a_1 \vee A = a_2 \vee ... \vee A = a_n))$
  $= P((B = b \wedge A = a_1) \vee (B = b \wedge A = a_2)... \vee (B = b \wedge A = a_n)))$
  $= P(B = b \wedge A = a_1) + P((B = b \wedge A = a_2) \vee ... \vee (B = b \wedge A = a_n))) - P((B = b \wedge A = a_1) \wedge ((B = b \wedge A = a_2) \vee ... \vee (B = b \wedge A = a_n))))$
  $= P(B = b \wedge A = a_1) + P((B = b \wedge A = a_2) \vee ... \vee (B = b \wedge A = a_n)))$
  $= ...$
  $= \sum_{i=1}^{n} P(B = b \cap A = a_i)$
  $= \sum_{i=1}^{n} P(B = b|A = a_i) * P(A = a_i)$

- 1.3 (5 pts) Soldier A and Soldier B are practicing shooting. The probability that A would miss the target is 0.2 and the probability that B would miss the target is 0.5. The probability that both A and B would miss the targets is 0.1.

  - What is the probability that at least one of the two will miss the target?
  Answer: $P(A \vee B) = P(A) + P(B) - P(A \wedge B) = 0.6$

  - What is the probability that exactly one of the two soldiers will miss the target?
  Answer: $P(A \vee \bar{B}) + P(B \vee \bar{A}) = 0.5$

- 1.4 (4 pts) A box contains three cards. One card is red on both sides, one card is green on both sides, and one card is red on one side and green on the other. Then we randomly select one card from this box, and we can know the color of the selected card's upper side. If this side is green, what is the probability that the other side of the card is also green?
  Answer:
  $P(the - other - side - green|this - side - green) = \frac{P(both - sides - green)}{P(this - side - green)} = \frac{1/3}{1/2} = \frac{2}{3}$

- 1.5 (4 pts) Suppose that the p.d.f. of a random variable X is:

$$f(x) = \begin{cases} cx^2, & for 1 \le x \le 2 \\ 0, & otherwise \end{cases} \quad (1)$$

- What is the value of constant c?
Answer:
$\int_1^2 cx^2 dx = \frac{7}{3}c = 1 \rightarrow c = \frac{3}{7}$

- Sketch the p.d.f.

- $Pr(X > 3/2) =?$
Answer: $\int_{3/2}^2 cx^2 dx = 37/56$

Question 2. Expectation (18 pts)

- 2.1 (4 pts) If an integer between 100 and 200 is to be chosen at random, what is the expected value?
Answer: $E(X) = \frac{1}{101}(100 + 101 + ... + 200) = 150$

- 2.2 (5 pts) A rabbit is playing a jumping game with friends. She starts from the origin of a real line and moves along the line in jumps of one step. For each jump, she flips a coin. If heads, she would jump one step to the left (i.e. negative direction). Otherwise, she would jump one step to the right. The chance of heads is p ($0 \le p \le 1$). What is the expected value of her position after n jumps ? (assume each step is in equal length and assume one step as one unit on the real line)
Answer:
For the ith jumping, $E(X_i) = (-1)p + (1)(1-p) = 1 - 2p$, So the position after n jumps is:
$E(X_1 + X_2 + ... + X_n) = E(X_1) + E(X_2) + ... + E(X_n) = n(1-2p)$

- 2.3 (4 pts) Suppose that the random variable X has a uniform distribution on interval [0, 1]. Random variable Y has a uniform distribution on the interval [4, 10]. X and Y are independent. Suppose a rectangle is to be constructed for which the lengths of two adjacent sides are X and Y. So what is the expected value of the area of this rectangle?
Answer:
Since X and Y are independent, $E(X \wedge Y) = E(X) * E(Y) = 0.5 * 7 = 3.5$

- 2.4 (5 pts) Suppose that X is a random variable. $E(X) = \mu, Var(X) = \sigma^2$, then what is the value of $E[X(X-1)] =?$
Answer:
$E[X(X-1)] = E[X^2] - E[X] = var(X) + E[X]^2 - \mu = \sigma^2 + \mu^2 - \mu$

Question 3. Normal Distribution (6 pts)

Suppose X has a normal distribution with mean 1 and variance 4. Find the value of the following:
a. $Pr(X \le 3)$
Answer:
$Pr(X \le 3) = Pr(Z \le \frac{3-1}{\sqrt{4}}) = \Phi(1) = 0.8413$

b. $Pr(|X| \le 2)$
Answer: $Pr(|X| \le 2) = \Phi(\frac{2-1}{2}) - \Phi(\frac{-2-1}{2}) = \Phi(0.5) - (1 - \Phi(1.5)) = 0.6247$

Question 4. Bayes' Theorem (8 pts)

In a certain day care class, 30 percent of the children have grey eyes, 50 percent of the children have blue eyes, and the other 20 percent's eyes are in other colors. One day they play a game together. In the first run, 65 percent of the grey eye kids were selected into the game, 82 percent of the blue eye kids selected in, and 50 percent of the kids with other colors were chosen. So if a child is selected at random from the class, and we know that he was not in the first run game, what is the probability that he has blue eyes?

Answer:

Assume B: blue eyes; O: other color eyes; G: grey eyes; NF: not in the first run game

$P(B|NF) = \frac{P(B)P(NF|B)}{P(B)P(NF|B)+P(O)P(NF|O)+P(G)P(NF|G)} = \frac{0.5*0.18}{0.5*0.18+0.2*0.5+0.3*0.35} = 0.3051$

Question 5. Probabilistic Inference (15 pts)

Imagine there are three boxes labelled A, B and C. Two of them are empty, and one contains a prize. Unfortunately, they are all closed and you don't know where the prize is. You first pick a box at random, say box A. However, before you open it, box B is opened by someone, and you see that it is empty. You now have to make your final choice as to what box to open: A or C.

Question: For each of the cases below, answer what box would you open so as to maximize the chances that the box you open contains the prize. Support your arguments by computing the probability of the prize being in box A and C.

Here are the three strategies according to which box B was chosen to be opened:

1. (5 pts) In this strategy if you first pick a box (in this case A) with a prize, then one of the other two boxes is opened at random. On the other hand, if you first choose a box that has no prize, then the empty box that you did not pick is chosen.

2. (5 pts) In this strategy it is just one of the two boxes that you did not pick is chosen at random (in this case it is a random choice between B and C).

3. (5 pts) In this strategy one of empty boxes is chosen at random (independently of whether you initially pick a box with a prize or not).

Answer:

Let SpB stand for a random event of "someone picks box B". In all the cases the prior (before box B was opened) probabilities that prize is in box A, B, or C are $P(A) = P(B) = P(C) = 1/3$. The differences are in the conditional probabilities: $P(SpB|A), P(SpB|B), P(SpB|C)$. In all three cases we compute posterior (after box B was opened) probabilities. We then pick a box with the highest probability of containing a prize.

1. $P(SpB|A) = 1/2$; $P(SpB|B) = 0$; $P(SpB|C) = 1$;
   $P(A|SpB) = \frac{P(SpB|A)P(A)}{P(SpB)}$ and $P(C|SpB) = \frac{P(SpB|C)P(C)}{P(SpB)}$;
   $P(SpB) = P(SpB|A)P(A) + P(SpB|B)P(B) + P(SpB|C)P(C) = 1/2 * 1/3 + 0 * 1/3 + 1 * 1/3 = 1/2$;
   $P(A|SpB) = \frac{1/2*1/3}{1/2} = 1/3$ and $P(C|SpB) = \frac{1*1/3}{1/2} = 2/3$;

2. $P(SpB|A) = 1/2$; $P(SpB|B) = 1/2$; $P(SpB|C) = 1/2$;
   Unlike in the previous sub-question, here the box that was opened by someone (namely, box B) could have contained a prize. Therefore, the posterior probabilities we are interested in are: $P(A|SpB \wedge \neg B)$ and $P(C|SpB \wedge \neg B)$.
   $P(A|SpB \wedge \neg B) = \frac{P(SpB\wedge\neg B|A)P(A)}{P(SpB\wedge\neg B)}$ and $P(C|SpB \wedge \neg B) = \frac{P(SpB\wedge\neg B|C)P(C)}{P(SpB\wedge\neg B)}$;
   $P(SpB \wedge \neg B|A) = P(SpB|A)P(\neg B|A) = 1/2 * 1 = 1/2$
   $P(SpB \wedge \neg B|B) = 0$;
   $P(SpB \wedge \neg B|C) = P(SpB|C)P(\neg B|C) = 1/2 * 1 = 1/2$
   $P(SpB \wedge \neg B) = P(SpB \wedge \neg B|A)P(A) + P(SpB \wedge \neg B|B)P(B) + P(SpB \wedge \neg B|C)P(C) = 1/2 * 1/3 + 0 * 1/3 + 1/2 * 1/3 = 1/3$;
   $P(A|SpB \wedge \neg B) = \frac{1/2*1/3}{1/3} = 1/2$ and $P(C|SpB \wedge \neg B) = \frac{1/2*1/3}{1/3} = 1/2$;

3. $P(SpB|A) = 1/2$; $P(SpB|B) = 0$; $P(SpB|C) = 1/2$;
   $P(A|SpB) = \frac{P(SpB|A)P(A)}{P(SpB)}$ and $P(C|SpB) = \frac{P(SpB|C)P(C)}{P(SpB)}$;
   $P(SpB) = P(SpB|A)P(A) + P(SpB|B)P(B) + P(SpB|C)P(C) = 1/2 * 1/3 + 0 * 1/3 + 1/2 * 1/3 = 1/3$;
   $P(A|SpB) = \frac{1/2*1/3}{1/3} = 1/2$ and $P(C|SpB) = \frac{1/2*1/3}{1/3} = 1/2$;

Question 6. PAC-learning I (15pts)

Consider an image classification problem. Suppose an algorithm first splits each image into $n = 4$ blocks (the blocks are non-overlapping and each block is at the same location and of constant size across all images) and computes some scalar feature value for each of the blocks (e.g., average intensity of the pixels within the block). Suppose that this feature is discrete and can take $m = 10$ values. The classification function classifies an image as 1 whenever each of the n feature values lies within some interval that is specific to this feature (i.e., the value of the first feature is between a_1 and b_1, the value of the second feature is between a_2 and b_2, and so on), and 0 otherwise. We would like to learn these intervals (a and b values for each interval) automatically based on a training set of images. All the other parameters such as locations and sizes of the blocks are *not* being learned. The following questions are helpful in understanding the requirements on the size of the training set.

1. (7 pts) What is the size of the hypothesis space H? Assume that only intervals with $a_i \leq b_i$ are considered for learning.

2. (4 pts) Assuming noiseless data and that the function we are trying to learn is capable of perfect classification, give an upper bound on the size of the training set required to be sure with 99% probability that the learned function will have true error rate of at most 5%.

3. (4 pts) Compare $|H|$ (the answer to question 1) and the required training dataset size R (the answer to question 2). Why does R not seem to be very affected by the number of possible hypotheses? What parameter does make R increase quickly and why? (Please provide only a few sentences for each question).

Answer:

1. The number of possible intervals for any particular feature value is computed as follows: for a given a_i the possible b_i values are from a_i to m (that is, $m - a_i + 1$ values); hence, the number of possible intervals is $m + (m-1) + (m-2) + \ldots + 1 = \frac{m(m+1)}{2}$;
   Since there are n features and we use a boolean conjunction function $|H| = (\frac{m(m+1)}{2})^n = 55^4$;

2. $R \geq \frac{0.69}{\epsilon}(log_2(55^4) + log_2(1/\delta)) = 410.82$;

3. In terms of formula, R is logarithmically related to the number of possible hypotheses and inverse proportionally to ϵ. Thus, ϵ affects R much stronger. Intuitively speaking, if a learned hypothesis is consistent with a large number of iid data points, then chances are it will classify correctly the test data points as well. This will hold independently of how many hypotheses we have. On the other hand, in order to guarantee a very small misclassification rate (ϵ), the hypothesis needs to be trained on a very large number of samples (so that they cover almost all of the input space).

Question 7. PAC-learning II (20pts)

Consider a learning problem in which input datapoints are real numbers distributed uniformly in between a and b, and output is binary. The true function we are trying to learn is $x < c^*$ for some $a \leq c^* \leq b$ (that is, output 1 whenever $x < c^*$ and 0 otherwise). The set of hypotheses is therefore: $H = \{(x < c) | a \leq c \leq b\}$ (the hypothesis space is therefore infinite: all real values of c in between a and b). Assuming that we have m datapoints for training, derive an upper bound on the probability of learning a hypothesis that will have

a true classification error larger than ϵ. The derivation should be done in the same spirit as the one used to derive a PAC bound on the probability of learning a 'bad' h for the case when hypothesis space H is finite. Do not use bounds that are based on VC-dim (please ignore this sentence if you do not know VC-dim anyway). Give the bound in terms of a, b, c^*, m and ϵ. Then evaluate it numerically for the following values: $a = 0$, $b = 2$, $c^* = 1$, $m = 20$ and $\epsilon = 0.1$. (Hint: you may need to use integrals).

Answer:

There are few possible answers to this. Here are some (others are also possible):

1.

$$\begin{aligned}
P(\text{we learn h' such that } \mathit{trueerror}(h') > \epsilon) &\leq \\
P(\text{the set H contains h' such that } \mathit{trueerror}(h') > \epsilon) &= \\
P(\exists h',\ h' \text{ is consistent with } m \text{ examples } \mathit{and}\ \mathit{trueerror}(h') > \epsilon) &= \\
P(\exists c, c \text{ is consistent with } x_1 \dots x_m\ \mathit{and}\ \mathit{trueerror}(h = (x < c)) > \epsilon) &\leq \\
P(x_1 \dots x_m \notin [\max(c^* - \epsilon(b-a), a); c^*]\ \mathit{or}\ x_1 \dots x_m \notin [c^*; \min(c^* + \epsilon(b-a), b)]) &= \\
P(x_1 \dots x_m \notin [\max(c^* - \epsilon(b-a), a); c^*]) + & \\
P(x_1 \dots x_m \notin [c^*; \min(c^* + \epsilon(b-a), b)]) - & \\
P(x_1 \dots x_m \notin [\max(c^* - \epsilon(b-a), a); c^*]\ \mathit{and}\ x_1 \dots x_m \notin [c^*; \min(c^* + \epsilon(b-a), b)]) &= \\
(1 - \frac{c^* - \max(c^* - \epsilon(b-a), a)}{b-a})^m + & \\
(1 - \frac{\min(c^* + \epsilon(b-a), b) - c^*}{b-a})^m - & \\
(1 - \frac{\min(c^* + \epsilon(b-a), b) - \max(c^* - \epsilon(b-a), a)}{b-a})^m &
\end{aligned}$$

For a well-behaved c^* we have: $\delta \leq 2(1-\epsilon)^m - (1-2\epsilon)^m$.

2. This version is *almost* correct and deserves a full credit if given.

$$\begin{aligned}
P(\text{we learn h' such that } \mathit{trueerror}(h') > \epsilon) &\leq \\
P(\text{the set H contains h' such that } \mathit{trueerror}(h') > \epsilon) &= \\
P(\exists h',\ h' \text{ is consistent with } m \text{ examples } \mathit{and}\ \mathit{trueerror}(h') > \epsilon) &= \\
P(\exists c, c \text{ is consistent with } x_1 \dots x_m\ \mathit{and}\ \mathit{trueerror}(h = (x < c)) > \epsilon) &\leq \\
\int_a^b P(c \text{ is consistent with } x_1 \dots x_m\ \mathit{and}\ \mathit{trueerror}(h = (x < c)) > \epsilon) dc &= \\
\int_a^b P(x_1 \dots x_m \notin [c, c^*] \text{ if } c < c^* \text{ or } x_1 \dots x_m \notin [c^*, c] \text{ if } c > c^*\ \mathit{and}\ |c, c^*| > \epsilon(b-a)) dc &= \\
\int_a^{\max(c^* - \epsilon(b-a), a)} P(x_1 \dots x_m \notin [c, c^*]) dc + \int_{\min(c^* + \epsilon(b-a), b)}^b P(x_1 \dots x_m \notin [c^*, c]) dc &= \\
\int_a^{\max(c^* - \epsilon(b-a), a)} (1 - (c^* - c)/(b-a))^m dc + \int_{\min(c^* + \epsilon(b-a), b)}^b (1 - (c - c^*)/(b-a))^m dc &= \\
1/(b-a)^m * (\int_a^{\max(c^* - \epsilon(b-a), a)} (b - a - c^* + c)^m dc + \int_{\min(c^* + \epsilon(b-a), b)}^b (b - a - c + c^*)^m dc) &= \\
1/((m+1)(b-a))^m * ((b - a - c^* + \max(c^* - \epsilon(b-a), a))^{m+1} - (b - a - c^* + a)^{m+1} - & \\
(b - a - b + c^*)^{m+1} + (b - a - \min(c^* + \epsilon(b-a), b) + c^*)^{m+1}) &= \\
1/((m+1)(b-a))^m * ((b - a - c^* + \max(c^* - \epsilon(b-a), a))^{m+1} - (b - c^*)^{m+1} - & \\
(-a + c^*)^{m+1} + (b - a - \min(c^* + \epsilon(b-a), b) + c^*)^{m+1}) &
\end{aligned}$$

10-701 Machine Learning, Spring 2011: Homework 2

Due: Friday Feb. 4 at 4pm in Sharon Cavlovich's office (GHC 8215)

Instructions There are 3 questions on this assignment. The last question involves coding. Please submit your writeup as 3 **separate** sets of pages according to TAs, with your name and userid on each set. If you choose to work with a partner for question 3, you (as a team) should submit one set of pages for that question, labeled with both names/userids. You should still submit the solutions to questions 1-2 separately.

1 Probability [Xi Chen, 30 points]

1. This problem studies the relationship between entropy, conditional entropy, mutual information, conditional independence, and expected values.

   Consider random variables X and Y with joint probability density $p(X, Y)$. The expected value of any function $f(X, Y)$ of these variables is defined as $\mathbb{E}_p f(X, Y) = \int p(X, Y) f(X, Y) dxdy$ (i.e., it is the value of $f(X, Y)$ averaged over the different values X and Y can take on, weighted by their probabilities.) The expected value of a quantity is also sometimes called its "expectation".

   The entropy, joint entropy and conditional entropy can be expressed as the following expectations:

   - Entropy: $H(X) = -\mathbb{E}_p \ln p(X) = -\int p(X) \ln p(X) dx = -\int p(X, Y) \ln p(X) dxdy$
   - Joint entropy: $H(X, Y) = -\mathbb{E}_p \ln p(X, Y) = -\int p(X, Y) \ln p(X, Y) dxdy$
   - Conditional entropy: $H(X|Y) = -\mathbb{E}_p \ln p(X|Y) = -\int p(X, Y) \ln p(X|Y) dxdy$

   (a) In class we defined mutual information as

   $$I(X, Y) \triangleq H(X) - H(X|Y).$$

   Please use the linearity property of the expectation (i.e. $\mathbb{E}_p(X + Y) = \mathbb{E}_p(X) + \mathbb{E}_p(Y)$) to express $I(X, Y)$ as the expected value of a specific quantity. [2pt]

   (b) Use the linearity property of the expectation to prove the chain rule for entropy[1] [3pt]:

   $$H(X, Y) = H(Y) + H(X|Y).$$

   Notice that given this chain rule for entropy, we can easily derive that $I(X, Y) = H(X) + H(Y) - H(X, Y)$.

   (c) Recall the *conditional mutual information* between random variables X and Y given Z is defined by:

   $$I(X, Y|Z) \triangleq H(X|Z) - H(X|Y, Z).$$

   Express $I(X, Y|Z)$ as the expected value of a specific quantity. Also, state a conditional independence assumption that will guarantee $I(X, Y|Z) = 0$ [5pt].

[1] For your own interest, you may compare the chain rule for entropy to the chain rule in probability theory: $P(X, Y) = P(Y)P(X|Y)$

★ **SOLUTION:**

(a) Using the linearity property of the expectation:

$$\begin{aligned} I(X,Y) &= H(X) - H(X|Y) \\ &= -\mathbb{E}_p \ln p(X) + \mathbb{E}_p \ln p(X|Y) \\ &= -\mathbb{E}_p \ln \frac{p(X)}{p(X|Y)} \\ &= -\mathbb{E}_p \ln \frac{p(X)p(Y)}{p(X,Y)} \end{aligned}$$

(b)

$$\begin{aligned} H(Y) + H(X|Y) &= -\mathbb{E}_p \ln p(Y) - \mathbb{E}_p \ln p(X|Y) \\ &= -\mathbb{E}_p \left(\ln p(Y) + \ln p(X|Y)\right) \\ &= -\mathbb{E}_p \ln \left(p(X|Y)p(Y)\right) \\ &= -\mathbb{E}_p \left(\ln p(X,Y)\right) = H(X,Y) \end{aligned}$$

(c)

$$\begin{aligned} I(X,Y|Z) &= H(X|Z) - H(X|Y,Z) \\ &= -\mathbb{E}_p \ln p(X|Z) + \mathbb{E}_p \ln p(X|Y,Z) \\ &= -\mathbb{E}_p \ln \frac{p(X|Z)}{p(X|Y,Z)} \\ &= -\mathbb{E}_p \ln \frac{p(X|Z)p(Y|Z)}{p(X|Y,Z)p(Y|Z)} \\ &= -\mathbb{E}_p \ln \frac{p(X|Z)p(Y|Z)}{p(X,Y|Z)} \end{aligned}$$

If X and Y are *conditionally independent* given Z (i.e. $p(X|Z)p(Y|Z) = p(X,Y|Z)$), $I(X,Y|Z) = 0$.

2. Given two random variables X and Y, let $\mathbb{E}X$ and $\mathbb{E}Y$ denote the means (i.e., expected values) of X and Y and let σ_X and σ_Y denote the standard deviations of X and Y. Here are three quantities that are commonly used to characterize the relationship between X and Y:

   - Covariance: $\text{cov}(X,Y) = \mathbb{E}[(X - \mathbb{E}X)(Y - \mathbb{E}Y)] = \mathbb{E}(XY) - \mathbb{E}X\mathbb{E}Y$
   - Correlation: $\rho_{XY} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$
   - Mutual Information: $I(X,Y) = H(X) - H(X|Y) = KL\left(p(X,Y)||p(X)p(Y)\right)$
     where $KL(p||q) \equiv -\int p(x) \ln \frac{q(x)}{p(x)} dx$ is the Kullback-Leibler(KL) distance [2].

   (a) Recall the Cauchy-Schwarz inequality: for any two non-degenerate [3] random variables X and Y:

   $$\{\mathbb{E}(XY)\}^2 \leq \mathbb{E}(X^2)\mathbb{E}(Y^2),$$

   where the equality holds if and and only if $P(X = aY) = 1$ for some non-zero constant a.
   Use the Cauchy-Schwarz inequality to prove that for any two non-degenerate random variables X and Y, the absolute value of the correlation between X and Y is less than or equal to 1, i.e. $|\rho_{XY}| \leq 1$. [3pt]

   (b) Show conditions under which we have $\rho_{XY} = 1$ and the conditions under which $\rho_{XY} = -1$. [2pt]

[2] As shown in homework 1, the definitions of mutual information based on KL divergence and entropy are equivalent.
[3] A random variable that can only take on one value (i.e., a constant) is called a degenerate random variable.

(c) If $I(X,Y)=0$, can we conclude that $\rho_{XY}=0$ (or equivalently, $\mathrm{cov}(X,Y)=0$) ? If so, please give the proof; if not, please find the two random variables X and Y such that $I(X,Y)=0$ but $\rho_{XY}\neq 0$. [5pt]

(d) If $\rho_{XY}=0$, can we conclude that $I(X,Y)=0$? If so, please give the proof; if not, please find the two random variables X and Y such that $\rho_{XY}=0$ but $I(X,Y)\neq 0$. [10pt]

★ **SOLUTION:**

(a) We want to show that $|\rho_{XY}|\leq 1$. Plugging in the definition of ρ_{XY}, this is equivalent to showing that:

$$\left(\mathrm{cov}(X,Y)\right)^2 \leq \mathrm{var}(X)\mathrm{var}(Y),$$

which is further equivalent to showing that

$$\left\{\mathbb{E}\left((X-\mathbb{E}X)(Y-\mathbb{E}Y)\right)\right\}^2 \leq E\left((X-\mathbb{E}X)^2\right)E\left((Y-\mathbb{E}Y)^2\right).$$

Now note that $X-\mathbb{E}X$ and $Y-\mathbb{E}Y$ are themselves random variables, so the Cauchy-Schwarz inequality directly applies to them, and proves our statement $|\rho_{XY}|\leq 1$.

(b) By the Cauchy-Schwarz inequality, we get that $\left(\mathrm{cov}(X,Y)\right)^2 \leq \mathrm{var}(X)\mathrm{var}(Y)$ (i.e. $|\rho_{XY}|=1$) if and only if

$$P(X-\mathbb{E}X=a(Y-\mathbb{E}Y))=1,$$

for some real a. Therefore, if X is linear in Y, i.e. $X=aY+b$, we have $|\rho_{XY}|=1$. Plug this condition into $\mathrm{cov}(X,Y)$:

$$\begin{aligned}\mathrm{cov}(X,Y) &= \mathbb{E}(XY)-\mathbb{E}X\mathbb{E}Y\\ &= \mathbb{E}(X(aX+b))-\mathbb{E}X\mathbb{E}(aX+b)\\ &= a\mathbb{E}(X^2)-a(\mathbb{E}X)^2\\ &= a\mathrm{var}(X)\end{aligned}$$

Since $\mathrm{var}(X)>0$, if $a>0$, we have $\mathrm{cov}(X,Y)>0$ so $\rho_{XY}>0$. Since $|\rho_{XY}|=1$, $\rho_{XY}=1$. In contrast, if $a<0$, we have $\mathrm{cov}(X,Y)<0$ so $\rho_{XY}<0$. Since $|\rho_{XY}|=1$, $\rho_{XY}=-1$.
In summary, if $X=aY+b$ and $a>0$, $\rho_{XY}=1$; if $X=aY+b$ and $a<0$, $\rho_{XY}=-1$.

(c) If $I(X,Y)=0$, X and Y are statistically independent, i.e. $p(X,Y)=p(X)p(Y)$. We have

$$\begin{aligned}\mathrm{cov}(X,Y) &= \mathbb{E}(XY)-\mathbb{E}X\mathbb{E}Y\\ &= \int\int xyp(x,y)dxdy-\int xp(x)dx\int yp(y)dy\\ &= \int\int xyp(x)p(y)dxdy-\int xp(x)dx\int yp(y)dy\\ &= \int xp(x)dx\int yp(y)dy-\int xp(x)dx\int yp(y)dy\\ &= 0.\end{aligned}$$

In summary, if $I(X,Y)=0$, we can conclude that $\rho_{XY}=0$.

(d) The conclusion is that if $\rho_{XY}=0$, X and Y may not be independent, and hence $I(X,Y)$ may not be zero. Now we provide two uncorrelated random variables X and Y which are dependent. You may come up with your own example and will get full remark if your example is correct.
Let $X\sim N(0,1)$ follow a standard normal distribution and $Y=X^2$. From the definition of independence, X and Y are not independent. Since $x^3p(x)$ is an odd function, $\mathbb{E}(XY)=\mathbb{E}(X^3)=\int x^3p(x)=0$. Since $\mathbb{E}X=0$, we have

$$\begin{aligned}\mathrm{cov}(X,Y) &= \mathbb{E}(XY)-\mathbb{E}X\mathbb{E}Y\\ &= \mathbb{E}(X^3)-\mathbb{E}X\mathbb{E}(X^2)\\ &= 0-0=0\end{aligned}$$

Therefore, we have $\rho_{XY}=0$ but X and Y are dependent.

2 Generative and Discriminative Classifiers: Gaussian (Naive) Bayes and Logistic Regression [Yi Zhang, 30 points]

Recall that a generative classifier estimates $P(\mathbf{X}, Y) = P(Y)P(\mathbf{X}|Y)$, while a discriminative classifier directly estimates $P(Y|\mathbf{X})$[4]. For clarity, we highlight $\mathbf{X}$ in bold to emphasize that it usually represents a vector of multiple attributes, i.e., $\mathbf{X} =< X_1, X_2, \ldots, X_n >$. However, this question does **not** require students to derive the answer in vector/matrix notation.

In class we have observed an interesting relationship between a discriminative classifier (logistic regression) and a generative classifier (Gaussian naive Bayes): the form of $P(Y|\mathbf{X})$ derived from the assumptions of **a specific class** of Gaussian naive Bayes classifiers is precisely the form used by logistic regression. The derivation can be found in the required reading, **Mitchell: Naive Bayes and Logistic Regression, Section 3.1 (page 8 - 10)**. In that reading, we made the following assumptions for Gaussian naive Bayes classifiers to model $P(\mathbf{X}, Y) = P(Y)P(\mathbf{X}|Y)$ (rephrased from the required reading, with a few comments):

1. Y is a boolean variable following a Bernoulli distribution, with parameter $\pi = P(Y = 1)$ and thus $P(Y = 0) = 1 - \pi$.
2. $\mathbf{X} =< X_1, X_2, \ldots, X_n >$, where each attribute X_i is a continuous random variable. For each X_i, $P(X_i|Y = k)$ is a Gaussian distribution $N(\mu_{ik}, \sigma_i)$. Note that σ_i is the standard deviation of the Gaussian distribution (and thus σ_i^2 is the variance), which does **not** depend on k.
3. For all $i \neq j$, X_i and X_j are conditionally independent given Y. This is why this type of classifier is called "naive".

We say this is **a specific class** of Gaussian naive Bayes classifiers because we have made an assumption that the standard deviation σ_i of $P(X_i|Y = k)$ does not depend on the value k of Y. This is **not** a general assumption for Gaussian naive Bayes classifiers.

2.1 General Gaussian naive Bayes Classifiers and Logistic Regression [15 points]

Let's make our Gaussian naive Bayes classifiers a little more general by removing the assumption that the standard deviation σ_i of $P(X_i|Y = k)$ does not depend on k. As a result, for each X_i, $P(X_i|Y = k)$ is a Gaussian distribution $N(\mu_{ik}, \sigma_{ik})$, where $i = 1, 2, \ldots, n$ and $k = 0, 1$. Note that now the standard deviation σ_{ik} of $P(X_i|Y = k)$ depends on both the attribute index i and the value k of Y.

Question: is the new form of $P(Y|\mathbf{X})$ implied by this more general Gaussian naive Bayes classifier still the form used by logistic regression? Derive the new form of $P(Y|\mathbf{X})$ to prove your answer.

★ **SOLUTION:** **No**, the new $P(Y|\mathbf{X})$ implied by this more general Gaussian naive classifier is no longer the form used by logistic regression. Here is the derivation.

As shown in page 8 of [Mitchell: Naive Bayes and Logistic Regression], we have:

$$
\begin{aligned}
P(Y = 1|\mathbf{X}) &= \frac{P(Y = 1)P(\mathbf{X}|Y = 1)}{P(Y = 1)P(\mathbf{X}|Y = 1) + P(Y = 0)P(\mathbf{X}|Y = 0)} \\
&= \frac{1}{1 + \frac{P(Y=0)P(\mathbf{X}|Y=0)}{P(Y=1)P(\mathbf{X}|Y=1)}} \\
&= \frac{1}{1 + \exp(\ln \frac{P(Y=0)P(\mathbf{X}|Y=0)}{P(Y=1)P(\mathbf{X}|Y=1)})} \\
&= \frac{1}{1 + \exp(\ln \frac{1-\pi}{\pi} + \ln \frac{P(\mathbf{X}|Y=0)}{P(\mathbf{X}|Y=1)})} \\
&= \frac{1}{1 + \exp(\ln \frac{1-\pi}{\pi} + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}
\end{aligned}
$$

[4]Note that certain discriminative classifiers are non-probabilistic: they directly estimate a function $f : \mathbf{X} \rightarrow Y$ instead of $P(Y|\mathbf{X})$. We will see such classifiers later this semester, but it is beyond the scope of this question.

Now the standard deviation σ_{ik} of $P(X_i|Y=k)$ depends on both the attribute index i and the value k of Y, so we have:

$$\begin{aligned}
\sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)} &= \sum_i \ln \frac{\frac{1}{\sqrt{2\pi\sigma_{i0}^2}} \exp(\frac{-(X_i-\mu_{i0})^2}{2\sigma_{i0}^2})}{\frac{1}{\sqrt{2\pi\sigma_{i1}^2}} \exp(\frac{-(X_i-\mu_{i1})^2}{2\sigma_{i1}^2})} \\
&= \sum_i \ln \frac{\sigma_{i1}}{\sigma_{i0}} + \sum_i (\frac{(X_i-\mu_{i1})^2}{2\sigma_{i1}^2} - \frac{(X_i-\mu_{i0})^2}{2\sigma_{i0}^2}) \\
&= \sum_i \ln \frac{\sigma_{i1}}{\sigma_{i0}} + \sum_i \frac{(\sigma_{i0}^2-\sigma_{i1}^2)X_i^2 + 2(\mu_{i0}\sigma_{i1}^2 - \mu_{i1}\sigma_{i0}^2)X_i + \mu_{i1}^2\sigma_{i0}^2 - \mu_{i0}^2\sigma_{i1}^2}{2\sigma_{i0}^2\sigma_{i1}^2} \\
&= \sum_i (\ln \frac{\sigma_{i1}}{\sigma_{i0}} + \frac{\mu_{i1}^2\sigma_{i0}^2 - \mu_{i0}^2\sigma_{i1}^2}{2\sigma_{i0}^2\sigma_{i1}^2}) + \sum_i \frac{(\mu_{i0}\sigma_{i1}^2 - \mu_{i1}\sigma_{i0}^2)}{\sigma_{i0}^2\sigma_{i1}^2} X_i + \sum_i \frac{(\sigma_{i0}^2-\sigma_{i1}^2)}{2\sigma_{i0}^2\sigma_{i1}^2} X_i^2
\end{aligned}$$

As a result, we have $P(Y=1|\mathbf{X})$ as follows:

$$\begin{aligned}
P(Y=1|\mathbf{X}) &= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi} + \sum_i \ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)})} \\
&= \frac{1}{1+\exp(w_0 + \sum_i w_i X_i + \sum_i v_i X_i^2)}
\end{aligned}$$

where

$$\begin{aligned}
w_0 &= \ln\frac{1-\pi}{\pi} + \sum_i (\ln\frac{\sigma_{i1}}{\sigma_{i0}} + \frac{\mu_{i1}^2\sigma_{i0}^2 - \mu_{i0}^2\sigma_{i1}^2}{2\sigma_{i0}^2\sigma_{i1}^2}) \\
w_i &= \frac{(\mu_{i0}\sigma_{i1}^2 - \mu_{i1}\sigma_{i0}^2)}{\sigma_{i0}^2\sigma_{i1}^2} \\
v_i &= \frac{(\sigma_{i0}^2-\sigma_{i1}^2)}{2\sigma_{i0}^2\sigma_{i1}^2}
\end{aligned}$$

In general we do not have $\sigma_{i1} = \sigma_{i0}$ so the quadratic term $v_i X_i^2$ in $P(Y=1|\mathbf{X})$ will not disappear. Therefore the form of $P(Y=1|\mathbf{X})$ is no longer the form of logistic regression.

2.2 Gaussian Bayes Classifiers and Logistic Regression [15 points]

Students in 10-701 are all smart, so clearly we will not be satisfied by only studying a "naive" classifier. In this part, we will turn our attention to a specific class of Gaussian Bayes classifiers (without "naive", yeah!). We consider the following assumptions for our Gaussian Bayes classifiers:

1. Y is a boolean variable following a Bernoulli distribution, with parameter $\pi = P(Y=1)$ and thus $P(Y=0) = 1-\pi$.
2. $\mathbf{X} =< X_1, X_2 >$, i.e., we only consider **two** attributes, where each attribute X_i is a continuous random variable. X_1 and X_2 are **not** conditionally independent given Y. We assume $P(X_1, X_2|Y=k)$ is a **bivariate Gaussian distribution** $N(\mu_{1k}, \mu_{2k}, \sigma_1, \sigma_2, \rho)$, where μ_{1k} and μ_{2k} are means of X_1 and X_2, σ_1 and σ_2 are standard deviations of X_1 and X_2, and ρ is the **correlation** between X_1 and X_2. Note that μ_{1k} and μ_{2k} depend on the value k of Y, but σ_1, σ_2, and ρ do **not** depend on Y. Also recall that the density of a bivariate Gaussian distribution, given $(\mu_{1k}, \mu_{2k}, \sigma_1, \sigma_2, \rho)$, is:

$$P(X_1, X_2|Y=k) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp[-\frac{\sigma_2^2(X_1-\mu_{1k})^2 + \sigma_1^2(X_2-\mu_{2k})^2 - 2\rho\sigma_1\sigma_2(X_1-\mu_{1k})(X_2-\mu_{2k})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}]$$

Question: is the form of $P(Y|\mathbf{X})$ implied by such not-so-naive Gaussian Bayes classifiers still the form used by logistic regression? Derive the form of $P(Y|\mathbf{X})$ to prove your answer.

★ **SOLUTION:** **Yes**, the new $P(Y|\mathbf{X})$ implied by this not-so-naive Gaussian Bayes classifier is still the form used by logistic regression.

$$\begin{aligned}
P(Y=1|\mathbf{X}) &= \frac{P(Y=1)P(\mathbf{X}|Y=1)}{P(Y=1)P(\mathbf{X}|Y=1)+P(Y=0)P(\mathbf{X}|Y=0)} \\
&= \frac{1}{1+\frac{P(Y=0)P(\mathbf{X}|Y=0)}{P(Y=1)P(\mathbf{X}|Y=1)}} \\
&= \frac{1}{1+\exp(\ln\frac{P(Y=0)P(\mathbf{X}|Y=0)}{P(Y=1)P(\mathbf{X}|Y=1)})} \\
&= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\ln\frac{P(\mathbf{X}|Y=0)}{P(\mathbf{X}|Y=1)})} \\
&= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\ln\frac{P(X_1,X_2|Y=0)}{P(X_1,X_2|Y=1)})}
\end{aligned}$$

Note that we must work on the joint distribution $P(X_1, X_2|Y=0)$ and $P(X_1, X_2|Y=1)$ because X_1 and X_2 are no longer conditionally independent given Y. We proceed by focusing on the term $\ln\frac{P(X_1,X_2|Y=0)}{P(X_1,X_2|Y=1)}$:

$$\begin{aligned}
\ln\frac{P(X_1,X_2|Y=0)}{P(X_1,X_2|Y=1)} &= \ln\frac{\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}}{\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}}+\ln\exp[(*)] \\
&= \ln\exp[(*)] \\
&= (*)
\end{aligned}$$

where (*) is the following formulation, obtained as the difference between the exponential parts of the bivariate Gaussian densities $P(X_1, X_2|Y=0)$ and $P(X_1, X_2|Y=1)$:

$$\begin{aligned}
(*) &= \frac{\sigma_2^2(X_1-\mu_{11})^2+\sigma_1^2(X_2-\mu_{21})^2-2\rho\sigma_1\sigma_2(X_1-\mu_{11})(X_2-\mu_{21})}{2(1-\rho^2)\sigma_1^2\sigma_2^2} \\
&\quad - \frac{\sigma_2^2(X_1-\mu_{10})^2+\sigma_1^2(X_2-\mu_{20})^2-2\rho\sigma_1\sigma_2(X_1-\mu_{10})(X_2-\mu_{20})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}
\end{aligned}$$

It turns out, by a few steps of derivations, that all quadratic terms X_1^2, X_2^2 and X_1X_2 are canceled in the above (*) and therefore we end up with the following:

$$\begin{aligned}
(*) &= \frac{2\sigma_2^2(\mu_{10}-\mu_{11})+2\rho\sigma_1\sigma_2(\mu_{21}-\mu_{20})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}X_1 \\
&\quad + \frac{2\sigma_1^2(\mu_{20}-\mu_{21})+2\rho\sigma_1\sigma_2(\mu_{11}-\mu_{10})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}X_2 \\
&\quad + \frac{\sigma_2^2(\mu_{11}^2-\mu_{10}^2)+\sigma_1^2(\mu_{21}^2-\mu_{20}^2)+2\rho\sigma_1\sigma_2(\mu_{10}\mu_{20}-\mu_{11}\mu_{21})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}
\end{aligned}$$

As a result, we have:

$$\begin{aligned}
P(Y=1|\mathbf{X}) &= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\ln\frac{P(X_1,X_2|Y=0)}{P(X_1,X_2|Y=1)})} \\
&= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+(*))} \\
&= \frac{1}{1+\exp(w_0+w_1X_1+w_2X_2)}
\end{aligned}$$

where

$$
\begin{aligned}
w_0 &= \ln\frac{1-\pi}{\pi} + \frac{\sigma_2^2(\mu_{11}^2-\mu_{10}^2)+\sigma_1^2(\mu_{21}^2-\mu_{20}^2)+2\rho\sigma_1\sigma_2(\mu_{10}\mu_{20}-\mu_{11}\mu_{21})}{2(1-\rho^2)\sigma_1^2\sigma_2^2} \\
w_1 &= \frac{2\sigma_2^2(\mu_{10}-\mu_{11})+2\rho\sigma_1\sigma_2(\mu_{21}-\mu_{20})}{2(1-\rho^2)\sigma_1^2\sigma_2^2} \\
w_2 &= \frac{2\sigma_1^2(\mu_{20}-\mu_{21})+2\rho\sigma_1\sigma_2(\mu_{11}-\mu_{10})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}
\end{aligned}
$$

This form of $P(Y=1|\mathbf{X})$ is still the form represented by logistic regression.

SPECIAL NOTES: we intentionally choose only two attributes $\mathbf{X} =< X_1, X_2 >$ because the density of *bivariate* Gaussian distribution can be expressed using scalars instead of vectors and matrices. This way, students do not need to derive the answer using vector/matrix algebra. However, if you are comfortable with vector and matrix algebra, please feel free to use following **alternative assumptions** and derive your answer in vector/matrix notation (which may turn out to be less time-consuming):

1. Y is a boolean variable following a Bernoulli distribution, with parameter $\pi = P(Y=1)$ and thus $P(Y=0) = 1-\pi$.
2. $\mathbf{X} =< X_1, X_2, \ldots, X_n >$ are **not** conditionally independent given Y, and $P(\mathbf{X}|Y=k)$ follows a **multivariate normal distribution** $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$. Note that $\boldsymbol{\mu}_k$ is the $n \times 1$ mean vector depending on the value of Y, and $\boldsymbol{\Sigma}$ is the $n \times n$ **covariance** matrix, which does **not** depend on Y. Also, you should be familiar with the density of multivariate normal distribution in vector/matrix notation.

You may choose to derive your answer for **either** the bivariate normal version using scalars **or** the multivariate normal version using vector/matrix notation. Derive both versions will **not** gain extra credits.

★ **SOLUTION:** In fact, deriving the answer in matrix and vector notation is much simpler and elegant. Now $\mathbf{X}$, μ_1 and μ_2 are $n \times 1$ vectors, and Σ is the $n \times n$ matrix. Similarly, we start with:

$$
\begin{aligned}
P(Y=1|\mathbf{X}) &= \frac{P(Y=1)P(\mathbf{X}|Y=1)}{P(Y=1)P(\mathbf{X}|Y=1)+P(Y=0)P(\mathbf{X}|Y=0)} \\
&= \frac{1}{1+\frac{P(Y=0)P(\mathbf{X}|Y=0)}{P(Y=1)P(\mathbf{X}|Y=1)}} \\
&= \frac{1}{1+\exp(\ln\frac{P(Y=0)P(\mathbf{X}|Y=0)}{P(Y=1)P(\mathbf{X}|Y=1)})} \\
&= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\ln\frac{P(\mathbf{X}|Y=0)}{P(\mathbf{X}|Y=1)})}
\end{aligned}
$$

Then we focus on the term $\ln\frac{P(\mathbf{X}|Y=0)}{P(\mathbf{X}|Y=1)}$:

$$
\begin{aligned}
\ln\frac{P(\mathbf{X}|Y=0)}{P(\mathbf{X}|Y=1)} &= \ln\frac{\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}}{\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}} + \ln\exp[(*)] \\
&= \ln\exp[(*)] \\
&= (*)
\end{aligned}
$$

where, again, (*) is the formulation obtained as the difference between the exponential parts of two *multivariate* Gaussian densities $P(\mathbf{X}|Y=0)$ and $P(\mathbf{X}|Y=1)$:

$$
\begin{aligned}
(*) &= \frac{1}{2}[(X-\mu_1)^T\Sigma^{-1}(X-\mu_1) - (X-\mu_0)^T\Sigma^{-1}(X-\mu_0)] \\
&= (\mu_0^T-\mu_1^T)\Sigma^{-1}X + \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0
\end{aligned}
$$

As a result, we have:

$$\begin{aligned} P(Y=1|\mathbf{X}) &= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1-\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0+(\mu_0^T-\mu_1^T)\Sigma^{-1}X)} \\ &= \frac{1}{1+\exp(w_0+\mathbf{w}^T\mathbf{X})} \end{aligned}$$

where $w_0 = \ln\frac{1-\pi}{\pi}+\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1-\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0$ is a scalar and $\mathbf{w}=\Sigma^{-1}(\mu_0-\mu_1)$ is an $n\times 1$ parameter vector.

This is still the form of logistic regression (in vector and matrix notation).

3 Naive Bayes Document Classifier [Carl Doersch, 40 points]

In this question, you will implement the Naive Bayes document classifier and apply it to the classic 20 newsgroups dataset[5]. In this dataset, each document is a posting that was made to one of 20 different usenet newsgroups. Our goal is to write a program which can predict which newsgroup a given document was posted to.

For this question, you may write your code and solution in teams of at most 2. If you decide to do this, you should submit one copy of your solutions to question 3 (both code and answers to questions) per team. This copy should be clearly marked with the names of both team members.

3.1 Model

Say we have a document D containing n words; call the words $\{X_1,...,X_n\}$. The value of random variable X_i is the word found in position i in the document. We wish to predict the label Y of the document, which can be one of m categories, We could use the model:

$$P(Y|X_1...X_n) \propto P(X_1...X_n|Y)P(Y) = P(Y)\prod_i P(X_i|Y)$$

That is, each X_i is sampled from some distribution that depends on its position X_i and the document category Y. As usual with discrete data, we assume that $P(X_i|Y)$ is a multinomial distribution over some vocabulary V; that is, each X_i can take one of $|V|$ possible values corresponding to the words in the vocabulary. Therefore, in this model, we are assuming (roughly) that for any pair of document positions i and j, $P(X_i|Y)$ may be completely different from $P(X_j|Y)$.

Question 3.1: In your answer sheet, explain in a sentence or two why it would be difficult to accurately estimate the parameters of this model on a reasonable set of documents (e.g. 1000 documents, each 1000 words long, where each word comes from a 50,000 word vocabulary). **[3 points]**

★ **SOLUTION:** In this model, each position in a given document is assumed to have its own probability distribution. Each document gives has only one word at each position, so if there are M documents then we must estimate the parameters a roughly 50,000-dimensional distribution using only M samples from that distribution. In only a thousand documents, there will not be enough samples.

To see it another way, the fact that a word w appeared at the i'th position of the document gives us information about the distribution at another position j. Namely, in English, it is possible to rearrange the words in a document without significantly altering the document's meaning, and therefore the fact that w appeared at i means that it is more likely to appear at position j. Thus, it would be *statistically inefficient* to not to make use of the information in estimating the parameters of the distribution of X_j.

To improve the model, we will make the additional assumption that:

$$\forall i,j \quad P(X_i|Y) = p(X_j|Y)$$

Thus, in addition to estimating $P(Y)$, you must estimate the parameters for the single distribution $P(X|Y)$, which we define to be equal to $P(X_i|Y)$ for all X_i. Each word in a document is assumed to be an *iid* draw from this distribution.

[5] http://people.csail.mit.edu/jrennie/20Newsgroups/

3.2 Data

The data file (available on the website) contains six files:

1. **vocabulary.txt** is a list of the words that may appear in documents. The line number is word's id in other files. That is, the first word ('archive') has wordId 1, the second ('name') has wordId 2, etc.

2. **newsgrouplabels.txt** is a list of newsgroups from which a docment may have come. Again, the line number corresponds to the label's id, which is used in the .label files. The first line ('alt.atheism') has id 1, etc.

3. **train.label** Each line corresponds to the label for one document from the training set. Again, the document's id (docId) is the line number.

4. **test.label** The same as train.label, except that the labels are for the test documents.

5. **train.data** Specifies the counts for each of the words used in each of the documents. Each line is of the form "`docId wordId count`", where `count` specifies the number of times the word with id `wordId` in the training document with id `docId`. All word/document pairs that do not appear in the file have count 0.

6. **test.data** Same as train.data, except that it specified counts for test documents.

If you are using matlab, the functions `textread` and `sparse` will be useful in reading these files.

3.3 Implementation

Your first task is to implement the Naive Bayes classifier specified above. You should estimate $P(Y)$ using the MLE, and estimate $P(X|Y)$ using a MAP estimate with the prior distribution $Dirichlet(1+\alpha, ..., 1+\alpha)$, where $\alpha = 1/|V|$ and V is vocabulary.

Question 3.2: In your answer sheet, report your overall testing accuracy (Nubmer of correctly classified documents in the test set over the total number of test documents), and print out the confusion matrix (the matrix C, where c_{ij} is the number of times a document with ground truth category j was classified as category i). **[7 points]**

★ **SOLUTION:** The final accuracy of this classifier is **78.52%**, with the following confusion matrix:

		Predicted Class																			
		alt.atheism	comp.graphics	comp.os.ms-windows.misc	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	comp.windows.x	misc.forsale	rec.autos	rec.motorcycles	rec.sport.baseball	rec.sport.hockey	sci.crypt	sci.electronics	sci.med	sci.space	soc.religion.christian	talk.politics.guns	talk.politics.mideast	talk.politics.misc	talk.religion.misc
True Class	alt.atheism	249	0	0	0	0	1	0	0	1	0	0	2	0	3	3	24	2	3	4	26
	comp.graphics	0	286	13	14	9	22	4	1	1	0	1	11	8	6	10	1	2	0	0	0
	comp.os.ms-windows.misc	1	33	204	57	19	21	4	2	3	0	0	12	5	10	8	3	1	0	5	3
	comp.sys.ibm.pc.hardware	0	11	30	277	20	1	10	2	1	0	1	4	32	1	2	0	0	0	0	0
	comp.sys.mac.hardware	0	17	13	30	269	0	12	2	2	0	0	3	21	8	4	0	1	0	1	0
	comp.windows.x	0	54	16	6	3	285	1	1	3	0	0	5	3	6	4	0	1	1	1	0
	misc.forsale	0	7	5	32	16	1	270	17	8	1	2	0	7	4	6	0	2	1	2	1
	rec.autos	0	3	1	2	0	0	14	331	17	0	0	1	13	0	4	2	0	0	6	1
	rec.motorcycles	0	1	0	1	0	0	2	27	360	0	0	0	3	1	0	0	1	1	0	0
	rec.sport.baseball	0	0	0	1	1	0	2	1	2	352	17	0	1	3	3	5	2	1	5	1
	rec.sport.hockey	2	0	1	0	0	0	2	1	2	4	383	0	0	0	0	1	2	0	1	0
	sci.crypt	0	3	0	3	4	1	0	0	0	1	1	362	2	2	2	0	9	0	5	0
	sci.electronics	3	20	4	25	7	4	8	11	6	0	0	21	264	9	7	1	3	0	0	0
	sci.med	5	7	0	3	0	0	3	5	4	1	0	1	8	320	8	7	6	5	8	2
	sci.space	0	8	0	1	0	3	1	0	1	0	1	4	6	5	343	3	2	1	12	1
	soc.religion.christian	11	2	0	0	0	2	1	0	0	0	0	0	0	2	0	362	0	1	2	15
	talk.politics.guns	1	1	0	0	0	1	1	2	1	1	0	4	0	5	2	1	303	5	23	13
	talk.politics.mideast	12	1	0	1	0	0	1	2	0	2	0	2	1	0	0	6	3	326	18	1
	talk.politics.misc	6	1	0	0	1	1	0	0	0	0	0	5	0	10	6	2	63	6	196	13
	talk.religion.misc	39	3	0	0	0	0	0	0	1	1	0	1	0	2	6	27	10	3	7	151

Question 3.3: Are there any newsgroups that the algorithm confuses more often than others? Why do you think this is? [**2 points**]

★ **SOLUTION:** From the confusion matrix, it is clear that newsgroups with a similar topics are confused frequently. Notably, those related to computers (eg comp.os.ms-windows.misc and comp.sys.ibm.pc.hardware), those related to politics (e.g. talk.politics.guns and talk.politics.misc), and those related to religion (alt.atheism and talk.religon.misc). Newsgroups with similar topics have similar words that identify them. For example, we would expect the computer-related groups to all use computer terms frequently.

3.4 Priors and Overfitting

In your initial implementation, you used a prior $Dirichlet(1+\alpha, ..., 1+\alpha)$ to estimate $P(X|Y)$, and I told you set $\alpha = 1/|V|$. Hopefully you wondered where this value came from. In practice, the choice of prior is a difficult question in Bayesian learning: either we must use domain knowledge, or we must look at the performance of different values on some validation set. Here we will use the performance on the testing set to gauge the effect of α [6].

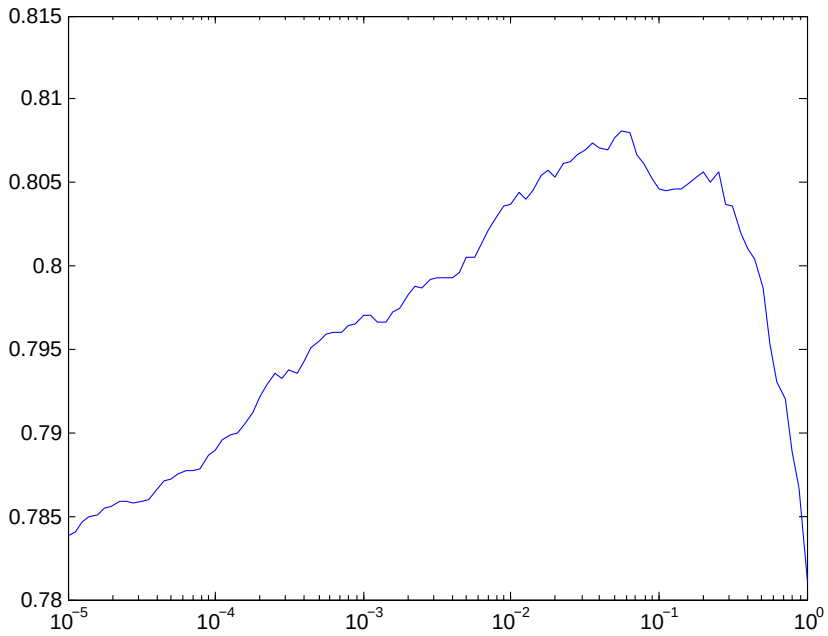
Question 3.4: Re-train your Naive Bayes classifier for values of α between .00001 and 1 and report the accuracy over the test set for each value of α. Create a plot with values of α on the x-axis and accuracy on the y-axis. Use a logarithmic scale for the x-axis (in Matlab, the `semilogx` command). Explain in a few sentences why accuracy drops for both small and large values of α [**5 points**]

★ **SOLUTION:** For very small values of α, we have that the probability of rare words not seen during training for a given class tends to zero. There are many testing documents that contain words seen only in one or two training documents, and often these training documents are of a different class than the test document. As α tends to zero, the probabilities of these rare words tends to dominate.

A number of students attributed the poor performance at small values of α to 'overfitting'. While this is strictly speaking correct (the classifier estimates $P(X|Y)$ to be smaller than is realistic simply because that was the case in the data), simply attributing this to overfitting is not a sophisticated answer. Different classifiers overfit for different reasons, and understanding the differences is an important goal for you as students.

For large values of α, we see a classic underfitting behavior: the final parameter estimates are closer tend toward the prior as α increases, and the prior is just something we made up. In particular, the classifier tends to underestimate the importance of rare words: for example, if α is 1 and we see only one occurrence of the word w in the category C (and we see the same number of words in each category), then the final parameter estimates are 2/21 for category C and 19/21 that it would be something else. Furthermore, the most informative words tend to be relatively uncommon, and so we would like to rely on these rare words more.

[6] It is tempting to choose α to be the one with the best performance on the testing set. However, if we do this, then we can no longer assume that the classifier's performance on the test set is an unbiased estimate of the classifier's performance in general. The act of choosing α based on the test set is equivalent to training on the test set; like any training procedure, this choice is subject to overfitting.



3.5 Identifying Important Features

One useful property of Naive Bayes is that its simplicity makes it easy to understand why the classifier behaves the way it does. This can be useful both while debugging your algorithm and for understanding your dataset in general. For example, it is possible to identify which words are strong indicators of the category labels we're interested in.

Question 3.5: Propose a method for ranking the words in the dataset based on how much the classifier 'relies on' them when performing its classification (hint: information theory will help). Your metric should use only the classifier's estimates of $P(Y)$ and $P(X|Y)$. It should give high scores to those words that appear frequently in one or a few of the newsgroups but not in other ones. Words that are used frequently in general English ('the', 'of', etc.) should have lower scores, as well as words that only appear appear extremely rarely throughout the whole dataset. Finally, your method this should be an overall ranking for the words, not a per-category ranking.[**3 points**]

★ **SOLUTION:** First of all, a handful of people whined about not liking the open-endedness of this problem. I hate to say it, but nebulous problems like this are common in machine learning–this problem was actually inspired by something I worked on last summer in industry. The goal was to design a metric for finding documents similar to some query document, and part of the procedure involved classifying words in the query document into one of 100 categories, based on the word itself and the word's context. The algorithm initially didn't work as well as I thought it should have, and the only path to improving its performance was to understand what these classifiers were 'relying on' in order to do their classification–some way of understanding the classifiers' internal workings, and even I wasn't sure what I was looking for. In the end I designed a metric based on information theory and, after looking at hundreds of word lists printed from these classifiers, I eventually found a way to fix the problem. I felt this experience was valuable enough that I should pass it on to all of you.

There were many acceptable solutions to this question, but probably the most successful ones studied the distribution of indicator variables $X_i = (X == the\_i'th\_word\_in\_V)$. A common error was the use of imprecise notation: note that $H(X|Y)$ and $I(X,Y)$ are just single numbers; these expressions have different meanings than the expressions $H(X_i|Y)$ and $I(X_i,Y)$.

The first measure is $H(Y|X_i = True)$, the entropy of the label given a document with a single word w_i. Intuitively, this value will be low if a word appears most of the time in a single class, because the distribution $P(Y|X_i = True)$ will be highly peaked. More concretely (and abbreviating $True$ as T),

$$\begin{aligned} H(Y|X_i = T) &= -\sum_k P(Y = y_k|X_i = T) log(P(Y = y_k|X_i = T) \\ &= -E_{P(Y|X_i=T)} log(P(Y = y_k|X_i = T) \\ &= -E_{P(Y|X_i=T)} log \frac{P(X_i=T|Y=y_k)P(Y=y_k)}{P(X_i=T)} \\ &= -E_{P(Y|X_i=T)} log \frac{P(X_i=T|Y=y_k)}{P(X_i=T)} - E_{P(Y|X_i=T)} log(P(Y = y_k)) \end{aligned}$$

Note that

$$log \frac{P(X_i = T|Y = y_k)}{P(X_i = T)}$$

is exactly what gets added to Naive Bayes' internal estimate of the posterior probability $log(P(Y))$ at each step of the algorithm (although in implementations we usually ignore the constant $P(X_i = T)$). Furthermore, the expectation is over the posterior distribution of the class labels given the appearance of word w_i. Thus, the first term of this measure can be interpreted as the expected change in the classifier's estimate of the log-probability of the 'correct' class given the appearance of word w_i. The second term tends to be very small relative to the first term since P(Y) is close to uniform; I found that the word list is the same with or without it.

Another popular measure was $I(X_i, Y)$, which Prof. Mitchell said was quite useful in fMRI data. Intuiively, this measures the amount of information we learn by observing X_i. An issue with this measure is that Naive Bayes only really learns from X_i in the event that $X_i = True$, and essentially ignores this variable when $X_i = False$ (thus, the issue was introduced because we're computing our measure on X_i rather than on X). Note that this is not the case in fMRI data (i.e. you compute the mutual information directly on the features used for classification), which explains why mutual information works better in that domain. Note that $X_i = False$ most of the time for informative words, so in the formula:

$$I(X_i, Y) = H(X_i) - H(X_i|Y) =$$

$$- \sum_{x_i \in \{T,F\}} P(X_i = x_i) \left[log(X_i = x_i) - \sum_k P(Y = y_k|X_i = x_i) log(P(Y = y_k|X_i = x_i)) \right]$$

We see that the term for $x_i = F$ tends to dominate even though it is essentially meaningless.

Another disadvantage of this metric is that it's more difficult to implement; the majority of people who tried made some mistake. Most notably, quite a few people forgot to sum over the possible values of x_i.

Question 3.6: Implement your method, set α back to $1/|V|$, and print out the 100 words with the highest measure. **[2 points]**

★ **SOLUTION:** For the metric $H(Y|X_i = True)$:

```
''nhl'', ''stephanopoulos'', ''leafs'', ''alomar'', ''wolverine'', ''crypto'', ''lemieux'',
''oname'', ''rsa'', ''athos'', ''ripem'', ''rbi'', ''firearm'', ''powerbook'', ''pitcher'',
''bruins'', ''dyer'', ''lindros'', ''lciii'', ''ahl'', ''fprintf'', ''candida'', ''azerbaijan'',
''baerga'', ''args'', ''iisi'', ''gilmour'', ''clh'', ''gfci'', ''pitchers'', ''gainey'', ''clemens'',
''dodgers'', ''jagr'', ''sabretooth'', ''liefeld'', ''hawks'', ''hobgoblin'', ''rlk'', ''adb'',
''crypt'', ''anonymity'', ''aspi'', ''countersteering'', ''xfree'', ''punisher'', ''recchi'',
''cipher'', ''oilers'', ''soderstrom'', ''azerbaijani'', ''obp'', ''goalie'', ''libxmu'', ''inning'',
''xmu'', ''sdpa'', ''argic'', ''serdar'', ''sumgait'', ''denning'', ''ioccc'', ''obfuscated'',
''umu'', ''nsmca'', ''dineen'', ''ranck'', ''xdm'', ''rayshade'', ''gaza'', ''stderr'', ''dpy'',
''cardinals'', ''potvin'', ''orbiter'', ''sandberg'', ''imake'', ''plaintext'', ''whalers'',
''moncton'', ''jaeger'', ''uccxkvb'', ''mydisplay'', ''wip'', ''hicnet'', ''homicides'', ''bontchev'',
''canadiens'', ''messier'', ''bure'', ''bikers'', ''cryptographic'', ''ssto'', ''motorcycling'',
''infante'', ''karabakh'', ''baku'', ''mutants'', ''keown'', ''cousineau''
```

For the metric $I(X_i, Y)$:

```
''windows'', ''god'', ''he'', ''scsi'', ''car'', ''drive'', ''space'', ''team'', ''dos'', ''bike'',
''file'', ''of'', ''that'', ''mb'', ''game'', ''key'', ''mac'', ''jesus'', ''window'', ''dod'',
''hockey'', ''the'', ''graphics'', ''card'', ''image'', ''his'', ''gun'', ''encryption'', ''sale'',
''apple'', ''government'', ''season'', ''we'', ''games'', ''israel'', ''disk'', ''files'',
''ide'', ''controller'', ''players'', ''shipping'', ''chip'', ''program'', ''was'', ''cars'',
''nasa'', ''win'', ''year'', ''were'', ''they'', ''turkish'', ''motif'', ''people'', ''armenian'',
''play'', ''drives'', ''bible'', ''use'', ''widget'', ''pc'', ''clipper'', ''offer'', ''jpeg'',
''baseball'', ''bus'', ''my'', ''nhl'', ''software'', ''is'', ''db'', ''server'', ''jews'',
''os'', ''israeli'', ''output'', ''data'', ''system'', ''who'', ''league'', ''armenians'',
''for'', ''christian'', ''christians'', ''entry'', ''mhz'', ''ftp'', ''price'', ''christ'',
''guns'', ''thanks'', ''church'', ''color'', ''teams'', ''privacy'', ''condition'', ''launch'',
''him'', ''com'', ''monitor'', ''ram''
```

(Note the presence of the words 'car', 'of', 'that', etc.

Question 3.7: If the points in the training dataset were not sampled independently at random from the same distribution of data we plan to classify in the future, we might call that training set *biased*. Dataset bias is a problem because the performance of a classifier on a biased dataset will not accurately reflect its future performance in the real world. Look again at the words your classifier is 'relying on'. Do you see any signs of dataset bias? [**3 points**]

★ **SOLUTION:** While I don't know exactly how this dataset was collected, it is certain that the dataset was collected over some finite time period in the past. That means our classifier will tend to rely on some words that are specific to this time period. For the first word list, 'stephanopolous' refers to a politician who may not be around in the future, and 'whalers' refers to the Connecticut hockey team that was actually being desolved at the same time as this dataset was being collected. For the second list, 'ghz' has almost certainly replaced 'mhz' in modern computer discussions, and the controversy regarding Turkey and Armenia is far less newsworthy today. As a result, you should expect the classification accuracy on the 20-newsgroups testing set to significantly overestimate the classification accuracy your algorithm would have on a testing sample from the same newsgroups taken today.

Sadly, there is a lot of bad machine learning research that has resulted from biased datasets. Researchers will train an algorithm on some dataset and find that the performance is excellent, but then apply it in the real world and find that the performance is terrible. This is especially common in vision datasets, where there is a tendency to always photograph a given object in the same environment or in the same pose. In your own research, make sure your datasets are realistic!

3.6 Hand-in for Question 3

Print out the code that you used to solve problem 3. This should include the code for the Naive Bayes classifier, the code for modifying α, and the code for identifying important features. Submit this code in class on the due date, along with your answers to questions 3.1-3.7. [**15 points**]

★ **SOLUTION:** The code for this question is provided in the solution code file on the webpage.

Solution for Assignment 2: MLE, EM, Regression

10-701/15-781: Machine Learning (Fall 2004)

Due: Oct. 14th 2004, Thursday, In class,

Question 1. Maximum Likelihood Estimation (20pts)

Suppose X is a binary random variable that takes value 0 with probability p and value 1 with probability $1-p$. Let $X_1, \ldots, X_n$ be iid samples of X.

1.1 (5pts) Compute an MLE estimate of p (denote it by $\hat{p}$).

1.2 (5pts) Is $\hat{p}$ an unbiased estimate of p? Prove the answer.

1.3 (5pts) Compute the expected square error of $\hat{p}$ in terms of p.

1.4 (5pts) Prove that if you know that p lies in the interval $[\frac{1}{4}; \frac{3}{4}]$ and you are given only $n = 3$ samples of X, then $\hat{p}$ is an inadmissible estimator of p when minimizing the expected square error of estimation. (An estimator δ of a parameter θ is said to be *inadmissible* when there exists a different estimator δ' such that $R(\theta, \delta') \leq R(\theta, \delta)$ for all θ and $R(\theta, \delta') < R(\theta, \delta)$ for some θ, where $R(\theta, \delta)$ is a risk function and in this problem it is the expected square error of the estimator).

Answer:

1.1 $\hat{p} = \arg\max_p P(X_1, \ldots, X_n|p) = \arg\max_p \prod_{i=1}^n P(X_i|p) = \arg\max_p p^k(1-p)^{n-k} =$
$\arg\max_p log(p^k(1-p)^{n-k}) = \arg\max_p(k * logp + (n-k) * log(1-p))$,
where k is the number of 0's in $X_1, \ldots, X_n$
$\frac{d(k*logp+(n-k)*log(1-p))}{dp} = \frac{k}{p} - \frac{n-k}{1-p} = 0$
Hence, $k(1-p) - (n-k)p = 0$
$\hat{p} = \frac{k}{n}$

1.2 $E\{\hat{p}\} = E\{\frac{k}{n}\} = \frac{E\{k\}}{n} = \frac{np}{n} = p$,
where we used the fact that $E\{k\} = np$ because k is a binomial random variable.
Alternatively, $E\{k\} = E\{n - \sum_{i=1}^n X_i\} = n - \sum_{i=1}^n E\{X_i\} = n - n(1-p) = np$.

1.3 $E\{(\hat{p} - p)^2\} = E\{\hat{p}^2\} - 2 * E\{\hat{p}\}p + p^2 = \frac{E\{k^2\}}{n^2} - 2p^2 + p^2 =$
$\frac{Var\{k\}+E^2\{k\}}{n^2} - p^2 = \frac{np(1-p)+(np)^2}{n^2} - p^2 = \frac{p}{n}(1-p)$,
where we used the fact that $Var\{k\} = np(1-p)$ because k is a binomial random variable,
and $Var\{k\} = E\{k^2\} - E^2\{k\}$.

1.4 Consider another estimator $\tilde{p} = 1/2$.
$E\{(\tilde{p} - p)^2\} = (1/2 - p)^2$
For $p = 1/2$ we have $E\{(\tilde{p} - p)^2\} = 0 < E\{(\hat{p} - p)^2\} = 1/12$
We now need to show $E\{(\tilde{p} - p)^2\} \leq E\{(\hat{p} - p)^2\}$ over $p \in [1/4; 3/4]$
$E\{(\tilde{p} - p)^2\} - E\{(\hat{p} - p)^2\} = (1/2 - p)^2 - 1/3 * p(1-p) = 1/4 - 4/3p + 4/3p^2$
This is a parabola going up, so we need to show that it lies below or equal to zero for $p \in [1/4; 3/4]$
It is equivalent to showing that it is below or equal to 0 at boundary points.
In fact it is: at both $p = 1/4$ and $p = 3/4$ $1/4 - 4/3p + 4/3p^2 = 0$

Question 2. EM (25pts)

For the following questions, please give clear step by step derivation.

2.1 (12pts) Suppose that the p.d.f. of a random variable X has a 2-component mixture form:

$$p_\alpha(x) = \alpha * p_1(x) + (1-\alpha) * p_2(x) \tag{1}$$

One component is the density model $p_1(x)$ and the other component is the density model $p_2(x)$. We know both $p_1(x)$ and $p_2(x)$. We do not know α. Given that $\{ x_1, x_2, ..., x_n \}$ are iid samples from the distribution of X, please give an EM algorithm for estimating α. (Describe the E-step and M-step clearly in your answer).

2.2 (13 pts) Suppose that $Y_1 \sim exp(1/\theta_1)$ and $Y_2 \sim exp(1/\theta_2)$, and $\theta_1 \neq \theta_2$. Y_1 and Y_2 are independent. Let $X = Y_1 + Y_2$ denote the sum of Y_1 and Y_2, Given that $\{ x_1, x_2, ..., x_n \}$ are iid samples from the distribution of X.

- Derive an expression for the density of X in terms of θ_1 and θ_2
(Hint1: The density of Y_1 is $f_{\theta_1}(y) = \theta_1 e^{-\theta_1 y}$, similarly for Y_2)
(Hint2: You could first derive CDF of X, $F(x) = P(Y_1 + Y_2 < x) = \int_0^x \int_0^{x-y_1} f_{\theta_1}(y_1) f_{\theta_2}(y_2) dy_2 dy_1$)

- Derive the E-step and M-step, and give explicit expressions for the parameter updates in the EM process for computing the MLE of θ_1 and θ_2.

Answer:

2.1 The question is a simple case of Bilmes's paper Page 3 and 4. (Jeff. Bilmes, "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models")

Let Y=$(y_1, y_2, ..., y_N)$ inform us which component "generated" each data item. $y_i \in \{1, 2\}$ for each i, and $y_i = k$ if the i^{th} sample was generated by the k^{th} mixture component.

E-step: We calculate the expectation of the complete likelihood:

$Q(\theta, \theta^{old}) = E_{P(Y|X,\theta^{old})}[log(L(\theta|X,Y))] = E_{P(Y|X,\theta^{old})}[\sum_{i=1}^N log(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i}))]$
*note: Y is a vector $(y_1, y_2, ..., y_N)$; And y_i only depends on x_i
$\therefore Q(\theta, \theta^{old}) = \sum_{i=1}^N \{E_{P(y_i|x_i,\theta^{old})}[log(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i}))]\} = \sum_{i=1}^N \{\sum_{y_i=1}^M [log(\alpha_{y_i} p_{y_i}(x_i|\theta_{y_i})) * p(y_i|x_i, \theta^{old})]\}$
*note: Then we could use l to substitute y_i
$Q(\theta, \theta^{old})$
$= \sum_{i=1}^N \{\sum_{l=1}^M [log(\alpha_l p_l(x_i|\theta_l)) * p(l|x_i, \theta^{old})]\}$
$= \sum_{i=1}^N \{\sum_{l=1}^M [log(\alpha_l) * p(l|x_i, \theta^{old})]\} + \sum_{i=1}^N \{\sum_{l=1}^M [log(p_l(x_i|\theta_l)) * p(l|x_i, \theta^{old})]\}$

For our problem, $M = 1$, $\alpha_1 = alpha$ and $\alpha_2 = 1 - alpha$. Maximize the expression in terms of $alpha$,

$$\frac{\partial}{\partial \alpha}[\sum_{i=1}^N log(\alpha) * p(y_i = 1|x_i, theta^{old}) + \sum_{i=1}^N log(1-\alpha) * p(y_i = 2|x_i, theta^{old})] = 0 \tag{2}$$

$$\frac{\sum_{i=1}^N p(y_i = 1|x_i, theta^{old})}{\alpha} = \frac{\sum_{i=1}^N p(y_i = 2|x_i, theta^{old})}{1-\alpha}] \tag{3}$$

$$\alpha = \frac{\sum_{i=1}^N p(y_i = 1|x_i, theta^{old})]}{\sum_{i=1}^N p(y_i = 1|x_i, theta^{old}) + \sum_{i=1}^N p(y_i = 2|x_i, theta^{old})} \tag{4}$$

$$\alpha = \frac{\sum_{i=1}^{N} p(y_i = 1|x_i, theta^{old})]}{N} \tag{5}$$

$p(y_i = 1|x_i, theta^{old}) = \frac{\alpha p_1(x_i)}{\alpha p_1(x_i)+(1-\alpha)p_2(x_i)}$
$p(y_i = 2|x_i, theta^{old}) = \frac{(1-\alpha)p_2(x_i)}{\alpha p_1(x_i)+(1-\alpha)p_2(x_i)}$

2.2 First to get pdf of X:

- The density of Y_1 is $f_{\theta_1}(y) = \theta_1 e^{-\theta_1 y}$ similarly for Y_2)
- First derive CDF of X

$$F(x) = P(Y_1 + Y_2 < x) = \int_0^x \int_0^{x-y_1} f_{\theta_1}(y_1) f_{\theta_2}(y_2) dy_2 dy_1 \tag{6}$$

$$F(x) = \int_0^x \theta_1 e^{-\theta_1 y_1} \int_0^{x-y_1} \theta_2 e^{-\theta_2 y_2} dy_2 dy_1 \tag{7}$$

$$F(x) = 1 - \frac{\theta_2 e^{-\theta_1 x} - \theta_1 e^{-\theta_2 x}}{\theta_2 - \theta_1} \tag{8}$$

$$p(x) = \frac{\partial F(x)}{\partial x} = \frac{\theta_1 \theta_2}{\theta_2 - \theta_1}[e^{-\theta_1 x} - e^{-\theta_2 x}] \tag{9}$$

Then let us derive EM steps for estimating parameters: θ_1 and θ_2

- Way1: The same framework as the Reference paper and the first page of this note.
  - E step: Expectation of the complete likelihood
  The expectation of complete log-likelihood is:

$$Q(\theta, \theta^{old}) = E_{P(Y|X,\theta^{old})}[log(L(\theta|X, Y))] = E_{P(Y|X,\theta^{old})}[\sum_{i=1}^{N} log(f_{\theta_1}(y_{i,1}) f_{\theta_2}(y_{i,2}))] \tag{10}$$

$$Q(\theta, \theta^{old}) = E_{P(Y|X,\theta^{old})}\{\sum_{i=1}^{N}[log(\theta_1) - \theta_1 * y_{i,1} + log(\theta_2) - \theta_2 * y_{i,2}]\} \tag{11}$$

  - M step: Maximization of the above expectation
  Maximize the above Q, in terms of $theta_1$, $theta_2$, we could get:

$$\frac{\partial Q(\theta, \theta^{old})}{\partial \theta_1} = 0 \Rightarrow \theta_1 = \frac{n}{\sum_{i=1}^{n} E_{p(y_{i,1}|x_i,\theta^{old})}[y_{i,1}]} \tag{12}$$

$$\frac{\partial Q(\theta, \theta^{old})}{\partial \theta_2} = 0 \Rightarrow \theta_2 = \frac{n}{\sum_{i=1}^{n} E_{p(y_{i,1}|x_i,\theta^{old})}[y_{i,2}]} \tag{13}$$

- Way2: Use the simple thinking style of EM as the GMM slides.
  - E step: Get expected value of the hidden variables $y_{i,1}$
  See the following derivation for $E_{p(y_{i,1}|x_i,\theta^{old})}[y_{i,1}]$
  - M step: Based on the expected value of $y_{i,1}$, derive θ
  Due to $Y_1 \sim exp(\frac{1}{\theta_1})$, the MLE estimation of exponential model's parameter $\frac{1}{\theta_1}$ is the sample mean of Y_1, then we could get that:

$$\frac{1}{\theta_1} = \frac{1}{n} \sum_{i=1}^{n} E_{p(y_{i,1}|x_i,\theta^{old})}[y_{i,1}] \tag{14}$$

$$\theta_1 = \frac{n}{\sum_{i=1}^{n} E_{p(y_{i,1}|x_i,\theta^{old})}[y_{i,1}]} \tag{15}$$

Same for θ_2:

$$\theta_2 = \frac{n}{\sum_{i=1}^{n} E_{p(y_{i,1}|x_i,\theta^{old})}[y_{i,2}]} \tag{16}$$

In both the above two ways, we need to get the $E_{p(y_{i,1}|x_i,\theta^{old})}[y_{i,1}]$:
Y_1 is our hidden variable. Then for the given $\{ x_1, x_2, ..., x_n \}$ samples, there would be corresponding $\{ y_{1,1}, y_{2,1}, ..., y_{n,1} \}$.

$$p(y_{i,1}|x_i,\theta^{old}) = \frac{p(x_i,y_{i,1})|\theta^{old}}{p(x_i|\theta^{old})} = \frac{\theta_1^{old} e^{-\theta_1^{old} y_{i,1}} \theta_2^{old} e^{-\theta_2^{old}(x_i - y_{i,2})}}{p(x_i|\theta^{old})} \tag{17}$$

$$p(y_{i,1}|x_i,\theta^{old}) = (\theta_2^{old} - \theta_1^{old}) \frac{e^{y_{i,1}(\theta_2^{old} - \theta_1^{old})}}{e^{x_i(\theta_2^{old} - \theta_1^{old})} - 1} \tag{18}$$

$$E_{p(y_{i,1}|x_i,\theta^{old})}[y_{i,1}] = \int_0^{x_i} y_{i,1} p(y_{i,1}|x_i,\theta^{old}) dy_{i,1} = \frac{x_i e^{x_i(\theta_2^{old} - \theta_1^{old})}}{e^{x_i(\theta_2^{old} - \theta_1^{old})} - 1} - \frac{1}{\theta_2^{old} - \theta_1^{old}} \tag{19}$$

$$E_{p(y_{i,1}|x_i,\theta^{old})}[y_{i,2}] = E_{p(y_{i,1}|x_i,\theta^{old})}[x_i - y_{i,1}] = x_i - E_{p(y_{i,1}|x_i,\theta^{old})}[y_{i,1}] \tag{20}$$

Question 3. Gaussian mixtures (35pts)

In this problem you will implement a Gaussian mixture model algorithm and will apply it to the problem of clustering gene expression data. Gene expression measures the levels of messenger RNA (mRNA) in the cell. The data you will be working with is from a model organism called yeast, and the measurements were taken to study the cell cycle system in that organism. The cell cycle system is one of the most important biological systems playing a major role in development and cancer.

All implementation should be done in Matlab. At the end of each sub-problem where you need to implement a new function we specify the prototype of the function.

3.1 Download the file 'alphaVals.txt'. This file contains 18 time points (every 7 minutes from 0 to 119) measuring the log expression ratios of 745 cycling genes. Each row in this file corresponds to one of the genes. Also, download the file 'geneNames.txt' which contains the names of these genes. For some of the genes, we are missing some of their values due to problems with the microarray technology (the tools used to measure gene expression). These cases are represented by values greater than 100.

3.2 (17pts) Implement (in matlab) an EM algorithm for learning a mixture of five (18-dimensional) Gaussians. It should learn means, convariance matrices and weights for each of the Gaussian. You can assume, however, independence between the different data points, resulting in a diagonal covariance matrix. How can you deal with the missing data? Why is this correct? Plot the centers identified for each of the five classes. Each center should be plotted as a time-series of 18 time points. Hand this plot with your solutions.

Here is the prototype of the matlab function you need to implement:

$$function[mu, s, w] = emcluster(x, k, ploton); \tag{21}$$

x is input data, where each row is an 18-dimensional sample. Values above 100 represent missing values. k is the number of desired clusters. *ploton* is either 1 or 0. If 1, then before returning the function plots log-likelihood of the data after each EM iteration (the function will have to store the log-likelihood of the data after each iteration, and then plot these values as a function of iteration number at the end). If 0, the function does not plot anything. The function outputs mu, a matrix

with k rows and 18 columns (each row is a center of a cluster), s is also k by 18, with each row being diagonal elements of the corresponding covariance matrix, and w is a column vector of size k, where $w(i)$ is a weight for ith cluster.

Answer: We have put a student code online. The implementation is pretty clear in terms of each step of the GMM iteration. A lot of students gave much more concise code than the one we chose. The purpose of the sample code is for those students who are not quite familiar with matlab coding.

The plot of the log-likelihood should be increasing.

The plots of the centers of each cluster should look like a sinusoid shape though with different phases (starting at a different point in the time series.

3.3 (3pts) How many more parameters would you have had to assign if we remove the independence assumption above? Explain.

Answer: The number of clusters times the number of covariances,
which is $k * ((d-1) + (d-2) + \ldots + 1) = \frac{kd}{2}(d-1)$,
where $d = 18$ in our case.

3.4 (8pts) Suggest and implement a method for determining the number of Gaussians (or classes) that are the most appropriate for this data. Please confine the set of choices to values in between 2 and 7. (Hint: the method can use an empirical evaluation of clustering results for each possible number of classes). Explain the method.

Here is the prototype of the matlab function you need to implement:

$$function[k, mu, s, w] = clust(x); \tag{22}$$

x is input data, where each row is an 18-dimensional sample. Once again values above 100 represent missing values. k is the number of classes selected by the function. mu, s and w are defined as in 3.2.

Answer: This is essentially a model selection question. You could use different model selection ways to solve it.
- cross validation
- train-test
- Minimum description length (One student used. If you do not know it, do not bother)
- BIC (Some student used. If you do not know it, do not bother)

3.5 (5pts) Use the Gaussians determined in (d) to perform hard clustering of your data by finding, for each gene i the Gaussian j that maximizes the likelihood: $p(i|j)$. Use the function 'printSelectedGenes.m' to write the names of the genes in each of the clusters to a separate file.

Here is the prototype of the matlab function you need to implement:

$$function[c] = hardclust(x, k, mu, s, w); \tag{23}$$

x is defined as before. k, mu, s, w are the output variables from the function written in 3.4 and are therefore defined there. c is a column vector of the same length as the number of rows in x. For each row, it should indicate the cluster the corresponding gene belongs to. The function should also write out files as specified above. The filenames should be: clust1, clust2, ..., clustk.

Answer: For each data point, assign the cluster that has the maximum probability for this point.

3.6 (2 pts) Use compSigClust.m to perform the statistical significance test (everything is already implemented here, so just use the function). Hand in a printout with the top three categories for each cluster

(this is the output of compSigClust.m).

Answer: Just run the code we provided on the cluster files you got above.

Question 4. Regression (20pts)

Linear regression models a real-valued output Y given an input vector X as

$$Y|X \sim Normal(\mu(X), \sigma^2)$$

where the mean is a linear function of the input: $\mu(X) = \beta^T X = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$

Logistic regression models a binary output Y by

$$Y|X \sim Bernoulli(\theta(X))$$

where the Bernoulli parameter is related to $\beta^T X$ by the logit transformation

$$logit(\theta(X)) \equiv \log\left(\frac{\theta(X)}{1-\theta(x)}\right) = \beta^T X$$

Given data $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, for each of the two regression models above , show that at the MLE $\hat{\beta}$

$$\sum_{i=1}^{n} x_i * y_i = \sum_{i=1}^{n} x_i * E[\ Y\ |X = x_i, \beta = \hat{\beta}]$$

Answer:

4.1 For linear regression: (the general way is by 'Maximum Likelihood Estimation'.)

$$Y|X \sim Normal(\mu(X), \sigma^2)$$

We could write the log likelihood as:

$$LL = log(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(y_i - \mu(x_i))^2}{2\sigma^2})) = \sum_{i=1}^{n} log(\frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(y_i - \beta^T x_i)^2}{2\sigma^2})) \tag{24}$$

$$\frac{\partial LL}{\partial \beta} = 0 \Rightarrow \frac{\partial \sum_{i=1}^{n}(y_i - \beta^T x_i)^2}{\partial \beta} = 0 \tag{25}$$

$$\Rightarrow \sum_{i=1}^{n} x_i * (y_i - \beta^T x_i) = 0 \Rightarrow \sum_{i=1}^{n} x_i * y_i = \sum_{i=1}^{n} x_i * (\hat{\beta}^T x_i) \tag{26}$$

Note: $E[\ Y\ |X = x_i, \beta = \hat{\beta}] = \mu(x_i) = \hat{\beta}^T x_i \Rightarrow \sum_{i=1}^{n} x_i * y_i = \sum_{i=1}^{n} x_i * E[\ Y\ |X = x_i, \beta = \hat{\beta}]$

4.2 For logistic regression: (the general way is still by 'Maximum Likelihood Estimation'.)
Logistic regression models a binary output Y by $Y|X \sim Bernoulli(\theta(X))$

$$\log\left(\frac{\theta(X)}{1-\theta(x)}\right) = \beta^T X \Rightarrow \theta(x) = \frac{\exp(\beta^T x)}{1+\exp(\beta^T x)} \text{ and } 1 - \theta(x) = \frac{1}{1+\exp(\beta^T x)}$$

We could write the log likelihood as:

$$LL = log(\prod_{i=1}^{n}\{\theta(x_i)^{y_i} * (1-\theta(x_i))^{1-y_i}\}) = \sum_{i=1}^{n}\{y_i * log(\theta(x_i)) + (1-y_i) * log(1-\theta(x_i))\}) \tag{27}$$

Plug $\theta(x_i)$ and $1-\theta(x_i)$ in, we get

$$LL = \sum_{i=1}^{n}\{y_i * (\beta^T x_i) - \log(1+\exp(\beta^T x_i))\} \tag{28}$$

$$\frac{\partial LL}{\partial \beta} = 0 \Rightarrow \sum_{i=1}^{n} x_i * y_i = \sum_{i=1}^{n} x_i * \frac{\exp(\beta^T x_i)}{1+\exp(\beta^T x_i)} \tag{29}$$

Note: by the property of Bernoulli distribution: $E[\ \mathrm{Y}\ |X = x_i, \beta = \hat{\beta}] = \theta(x_i) = \frac{\exp(\beta^T x_i)}{1+\exp(\beta^T x_i)}$

$$\Rightarrow \textstyle\sum_{i=1}^{n} x_i * y_i = \sum_{i=1}^{n} x_i * E[\ \mathrm{Y}\ |X = x_i, \beta = \hat{\beta}]$$

Note:
Actually in the above solutions, the full log likelihood function should look like the following first:

log-likelihood
$= log(\prod_{i=1}^{n}\{p(x_i, y_i)\})$
$= log(\prod_{i=1}^{n}\{p_{(Y|X)}(y_i|x_i) * p_X(x_i)\})$
$= log(\{\prod_{i=1}^{n} p_{(Y|X)}(y_i|x_i)\} * \{\prod_{i=1}^{n} p_X(x_i)\})$
$= log(\{\prod_{i=1}^{n} p_{(Y|X)}(y_i|x_i)\}) + log(\{\prod_{i=1}^{n} p_X(x_i)\})$
$= LL + LL_x$

Because LL_x do not involve with the parameter, in our maximization, we could just consider maximizing LL .

10-701 Machine Learning, Spring 2011: Homework 3

Due: Feb 17th, at the beginning of class

Instructions There are 2 questions on this assignment. The last question involves coding. Please submit your writeup as 2 **separate** sets of pages according to TAs, with your name and userid on each set.

1 Bayes Net [Carl, 40 points]

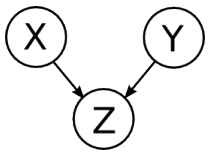


Figure 1: Graphical model for question 1.1

1. In this problem, you will study the behavior of the simple graphical model shown in figure 1, often called a 'collider'. Here, the probability of Z depends on two variables X and Y.

   (a) Write the factored distribution represented by this graph. That is, write the distribution as a product of factors each of which could be replaced by entries in the conditional probability tables (CPTs) in order to obtain the probability of an event.

   ★ **SOLUTION:** $P(X,Y,Z) = P(X)P(X)P(Z|X,Y)$

   (b) Assume that X,Y, and Z are binary. Write the CPTs for a distribution such that $P(X=1) < P(X=1|Y=1,Z=1) < P(X=1|Z=1)$. Compute the values $P(X=1)$,$P(X=1|Y=1,Z=1)$, and $P(X=1|Z=1)$.

   ★ **SOLUTION:** The trick here was to think about examples from class. The example of $X = car\_battery\_dead$, $Y = no\_gas$, $Z = engine\_starts$ almost behaves the way we want: knowing that the car doesn't start makes it more likely that the battery is dead ($P(X=1) < P(X=1|Z=1)$), and knowing that there is no gas makes it less likely that the battery is dead ($P(X=1|Y=1,Z=1) < P(X=1|Z=1)$). To enforce that $P(X=1) < P(X=1|Y=1,Z=1)$, we need to make sure that Y doesn't completely explain away the evidence $Z=1$: i.e., $P(Z=1|X=1,Y=1) > P(Z=1|X=0,Y=1)$. It turns out that the choices of $P(X)$ and $P(Y)$ don't make any difference unless they're 0 or 1. The following distribution works:
   $P(X=1) = .5; P(Y=1) = .5$
   $P(Z=1|X=1,Y=1) = 1$
   $P(Z=1|X=0,Y=1) = .5$
   $P(Z=1|X=1,Y=0) = .5$
   $P(Z=1|X=0,Y=0) = 0$
   Thus, we have $P(X) = .5$, $P(X|Y,Z) = P(X,Y,Z)/P(Y,Z) = .25/(.25+.125) = .66$, $P(X|Z) = P(X,Z)/P(Z) = (.25+.125)/(.25+.125+.125+0) = .75$,

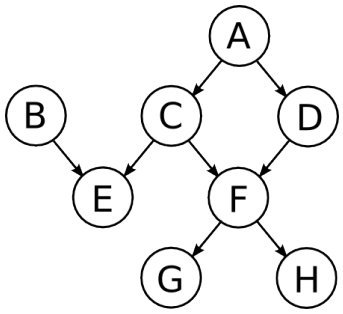


Figure 2: Graphical model for question 1.2

2. Let $P(A, B, C, D, E, F, G, H)$ factorize according to the graph given in figure 2. Prove or disprove whether the following independence properties hold.

   (a) $(A \perp H)$

   ★ **SOLUTION:** **false**: the path $A - D - F - H$ is not blocked.

   (b) $(B \perp G)$

   ★ **SOLUTION:** **true**: all paths from B to G pass through E, which is not known and therefore blocks.

   (c) $(B \perp G | E)$

   ★ **SOLUTION:** **false**: the path $B - E - C - F - G$ is not blocked.

   (d) $(C \perp D | G, A)$

   ★ **SOLUTION:** **false**: The path $C - F - D$ is not blocked (note that since we condition on G, and G is a descendant of F, F does not block.

   (e) $(C \perp D)$

   ★ **SOLUTION:** **false**: The path $C - A - D$ is not blocked since we do not condition on A.

   (f) $(B \perp D | C)$

   ★ **SOLUTION:** **true**: for the same reason that $B \perp G$.

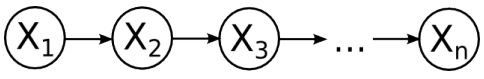


Figure 3: Graphical model for question 1.3

3. Let $\mathbf{X} = \{X_1, ..., X_n\}$ have a distribution $P(\mathbf{X})$ that factorizes according to the graph given in figure 3, and assume each X_i is binary. Design a polynomial-time algorithm which will compute the following

quantities. Use pseudocode for your answer. In your pseudocode, assume that the variables `px1` and `px` already exist, where `px1` is the prior probability $P(X_1 = 1)$, and `px` is an n-1 by 2 dimensional array, where `px[i][j]` is $P(X_{i+1} = 1 | X_{\mathtt{i}} = \mathtt{j} - 1)$, for $\mathtt{j} \in 1, 2$. For this problem, you may assume that the numbers have arbitrary precision, so you do not need to worry about underflow.

(a) Compute $P(X_n = 1 | X_1 = \mathtt{x1})$. Assume that `x1` has already been initialized with the value of $X_1 \in \{0.1\}$.

★ **SOLUTION:** For the following, I will use $P(X_k)$ to denote $P(X_k = 1)$. First, notice that:

$$\begin{aligned} P(X_n|X_1) &= \frac{P(X_n, X_1)}{P(X_1)} \\ &= \frac{\sum_{j\in\{0,1\}} P(X_n, X_{n-1} = j, X_1)}{P(X_1)} \\ &= \frac{\sum_{j\in\{0,1\}} P(X_n|X_{n-1} = j, X_1)P(X_{n-1} = j, X_1)}{P(X_1)} \end{aligned}$$

Using the conditional independence assumptions of our graph and rearranging, this may be rewritten:

$$\sum_{j\in\{0,1\}} P(X_n|X_{n-1} = j)P(X_{n-1} = j, X_1)$$

Thus, assuming we have a linear time algorithm to compute $P(X_{n-1}, X_1)$, then computing $P(X_n, X_1)$ remains linear. The actual code looks something like this:

```
function compute_pxn_giv_x1(px,px1,x1)
  P_xi_giv_x1=x1          //initialize with P(X_1=1|X_1=x1),
                          //which is trivially 0 or 1.
  for i=1 to n
    P_xi_giv_x1=(1-P_xi_giv_x1)*px[i][1]+P_xi_giv_x1*px[i][2]
  end
  return P_xi_giv_x1
end
```

(b) Compute $P(X_1 = 1 | X_n = \mathtt{xn})$. Assume that `xn` has already been initialized with the value of $X_n \in \{0.1\}$.

★ **SOLUTION:** The easiest solution is to use Bayes rule:

$$P(X_1|X_n) = \frac{P(X_n|X_1)P(X_1)}{P(X_n)} = \frac{P(X_n|X_1)P(X_1)}{P(X_n|X_1)P(X_1) + P(X_n|\neg X_1)P(\neg X_1)}$$

We already have expressions for all of these terms, so the final solution is:

```
function compute_px1_giv_xn(px,px1,xn)
  P_xn_giv_x1=compute_pxn_giv_x1(px,px1,1)
  P_xn_giv_not_x1=compute_pxn_giv_x1(px,px1,0)
  P_x1_giv_xn=P_xn_giv_x1*px1/(P_xn_giv_x1*px1 + P_xn_giv_not_x1*(1-px1))
  return P_x1_giv_xn
end
```

2 Multi-class Logistic Regression with Applications to Handwritten Digit Recognition [Xi, 60 points]

Recall the multi-class logistic regression model on page 6 of the lecture on Jan 27th. Assume that we have K different classes and each input $\mathbf{x}$ is a d dimensional vector. The posterior probability is given by:

$$\begin{aligned} P(Y=k|X=\mathbf{x}) &= \frac{\exp(\mathbf{w}_k^T\mathbf{x})}{1+\sum_{l=1}^{K-1}\exp(\mathbf{w}_l^T\mathbf{x})} \quad \text{for} \quad k=1,\ldots,K-1 \\ P(Y=K|X=\mathbf{x}) &= \frac{1}{1+\sum_{l=1}^{K-1}\exp(\mathbf{w}_l^T\mathbf{x})}, \end{aligned}$$

where $\mathbf{w}_k^T$ denotes the transpose of the $\mathbf{w}_k$ vector and $\mathbf{w}_k^T\mathbf{x}$ is the inner product of $\mathbf{w}_k$ and $\mathbf{x}$ (i.e., $\mathbf{w}_k^T\mathbf{x} = \sum_i w_{ki}x_i$). To simplify, we *ignore* the constant term w_{k0} in the logistic regression model.

For ease of notation, we can replace the two above expressions for $P(Y|X)$ by a single expression. We do this by introducing a fixed, pseudo parameter vector: $\mathbf{w}_K = \mathbf{0}$. Now for any class label k, we simply write

$$P(Y=k|X=\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T\mathbf{x})}{1+\sum_{l=1}^{K-1}\exp(\mathbf{w}_l^T\mathbf{x})} \quad \text{for} \quad k=1,\ldots,K$$

1. How many parameters do we need to estimate? What are these parameters? [5pt]
2. Given n training samples: $\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\ldots,(\mathbf{x}_n,y_n)\}$, please write down explicitly the log likelihood function and simplify it as much as you can:

   $$L(\mathbf{w}_1,\ldots,\mathbf{w}_{K-1}) = \sum_{i=1}^{n} \ln P(Y=y_i|X=\mathbf{x}_i).$$

   [5pt]
3. Compute the gradient of L with respect to each $\mathbf{w}_k$ and simply it. [5pt]
4. Now we add the regularization term $\frac{\lambda}{2}\sum_{l=1}^{K-1}\|\mathbf{w}_l\|_2^2$ and define a new objective function:

   $$f(\mathbf{w}_1,\ldots,\mathbf{w}_{K-1}) = L(\mathbf{w}_1,\ldots,\mathbf{w}_{K-1}) - \frac{\lambda}{2}\sum_{l=1}^{K-1}\|\mathbf{w}_l\|_2^2.$$

   Compute the gradient of f with respect to each $\mathbf{w}_k$. [5pt]
5. In this part, you are going to play with the "handwritten digit recognition" problem on the USPS digit dataset. Each hand-written digital image is 16 by 16 pixels. If we treat the value of each pixel as a boolean feature (either 0 for black or 1 for white), then each example has $16\times 16 = 256$ $\{0,1\}$-valued features, and hence $\mathbf{x}$ has 256 dimension. Each digit (i.e. 1,2,3,4,5,6,7,8,9,0) corresponds to a class label y ($y = 1...K, K = 10$). For each digit, we have 600 training samples and 500 testing samples. You can view these images at http://www.cs.nyu.edu/~roweis/data/usps_0.jpg, ..., http://www.cs.nyu.edu/~roweis/data/usps_9.jpg.

   Please download the data from the website. Load the **usps_digital.mat** file in **usps_digital.zip** into MATLAB. You will have four matrices:

   - tr_X: training input matrix with the dimension 6000×256.
   - tr_y: training label of the length 6000, each element is from 1 to 10.
   - te_X: testing input matrix with the dimension 5000×256.
   - te_y: testing label of the length 5000, each element is from 1 to 10.

For those who do NOT want to use MATLAB, we also provide the text file for these four matrices in **usps_digital.zip**.

Note that if you want to view the image of a particular training/testing example in MATLAB, say the 1000th training example, you may use the MATLAB command: **imshow(reshape(tr_X(1000,:),16,16))**

(a) [15pt] Use the *gradient ascent algorithm* to train a multi-class logistic regression classifier. Plot (1) the objective value (log-likelihood), (2) the training accuracy, and (3) the testing accuracy versus the number of iterations. Report your final testing accuracy, i.e. the fraction of test images that are correctly classified.

Note that you must choose a suitable learning rate (i.e. stepsize) of the gradient ascent algorithm. A hint is that your learning rate cannot be too large otherwise your objective will increase only for the first few iterations.

In addition, you need to choose a suitable stopping criterion. You might use the number of iterations, the decrease of the objective value, or the maximum of the L2 norms of the gradient with respect to each $\mathbf{w}_k$. Or you might watch the increase of the testing accuracy and stop the optimization when the accuracy is stable.

(b) [20pt] Now we add the regularization term $\frac{\lambda}{2}\sum_{l=1}^{K-1}\|\mathbf{w}_l\|_2^2$. For $\lambda = 1, 10, 100, 1000$, report the final testing accuracies.

(c) [5pt] What can you conclude from the above experiment? (hint: the relationship between the regularization weight and the prediction performance).

★ SOLUTION:

1. We need to estimate $\{\mathbf{w}_1, \ldots, \mathbf{w}_{K-1}\}$, where each one is a d-dimensional vector. Therefore, we need to estimate in total $(K-1)*d$ parameters.

2.
$$\begin{aligned} L(\mathbf{w}_1, \ldots, \mathbf{w}_{K-1}) &= \sum_{i=1}^{n} \ln P(Y = y_i | X = \mathbf{x}_i) \\ &= \sum_{i=1}^{n} \ln \frac{\exp(\mathbf{w}_{y_i}^T \mathbf{x}_i)}{1 + \sum_{l=1}^{K-1} \exp(\mathbf{w}_l^T \mathbf{x}_i)} \\ &= \sum_{i=1}^{n} \left[\mathbf{w}_{y_i}^T \mathbf{x}_i - \ln \left(1 + \sum_{l=1}^{K-1} \exp(\mathbf{w}_l^T \mathbf{x}_i) \right) \right] \end{aligned}$$

3.
$$\begin{aligned} \nabla L(\mathbf{w}_k) &= \sum_{i=1}^{n} \left[I(y_i = k)\mathbf{x}_i - \frac{\exp(\mathbf{w}_k^T \mathbf{x}_i)}{1 + \sum_{l=1}^{K-1} \exp(\mathbf{w}_l^T \mathbf{x}_i)} \mathbf{x}_i \right] \\ &= \sum_{i=1}^{n} \left[I(y_i = k) - \frac{\exp(\mathbf{w}_k^T \mathbf{x}_i)}{1 + \sum_{l=1}^{K-1} \exp(\mathbf{w}_l^T \mathbf{x}_i)} \right] \mathbf{x}_i \\ &= \sum_{i=1}^{n} \left[I(y_i = k) - P(y = k | X = \mathbf{x}_i) \right] \mathbf{x}_i \end{aligned}$$

4.
$$\nabla f(\mathbf{w}_k) = \nabla L(\mathbf{w}_k) - \lambda w_k.$$

5. (a) I use the stepsize $\eta = 0.0001$ and run the gradient ascent method for 5000 iterations. The objective value vs. the number of iterations; training error vs. the number of iterations; testing error vs. the number of iterations are presented in Figure 4.

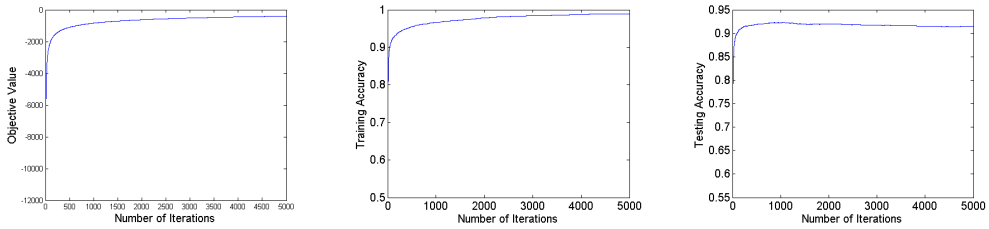


Figure 4: Performance of multi-class logistic regression on USPS digital dataset.

λ	0	1	10	100	1000
Testing Accuracy	91.44 %	91.58 %	91.92 %	89.74 %	79.78 %

Table 1: Comparison of testing accuracy for different regularization parameters.

(b) For $\lambda = 0, 1, 10, 100, 1000$, the comparison of the testing accuracy is presented in Table 1.

(c) From the above result, we can see that adding the regularization could avoid overfitting and lead to better generalization performance (e.g. $\lambda = 1, 10$). However, the regularization cannot be too large. Although a larger regularization can decrease the variance, it introduces additional bias and may lead to worse generalization performance.

Assignment 3: Classifiers, Clustering

10-701/15-781: Machine Learning (Fall 2004)

Out: Oct. 14th, 2004
Due: Oct. 28th 2004, Thursday, Start of class,

a *This assignment has four problems to test your understanding about classifiers and regression. Each of problems 1, 2, 3, and 5 are worth 15%, with problem 4 worth 40%.*

b *For the questions requiring programming, please use Matlab. You need to submit your code to the TAs (we'll provide instructions on how soon).*

c *For questions and clarifications, contact Max (questions 1 to 3) (maxim+@cs.cmu.edu) or Dave (questions 4 and 5) (dif+781@cmu.edu).*

d *Policy on collaboration:*

Homeworks will be done individually: each student must hand in their own answers. It is acceptable, however, for students to collaborate in figuring out answers and helping each other solve the problems. We will be assuming that, as participants in a graduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.

d *Policy on late homework:*

Homework is worth full credit at the beginning of class on the due date. It is worth half credit for the next 48 hours. It is worth zero credit after that. You must turn in all of the 4 homeworks, even if for zero credit, in order to pass the course.

Question 1. Naive and Joint Bayes Classifiers (15 pts)

1.1 (5 pts) Suppose A and B are independent binary random variables, each having a 50% chance of being 0. Construct a boolean function $y = f(A, B)$ where A is not independent of B given y but for which a naive Bayes classifier will have a 0% error rate (assuming infinite training data). Prove that the classifier will have 0% error rate.

Answer: Consider the function $y = A \vee B$. Since the training data is infinite, the estimated probabilities will get estimated as follows: $P(y = 0) = \frac{1}{4}$, $P(y = 1) = \frac{3}{4}$;
$P(A = 0|y = 0) = P(B = 0|y = 0) = 1$, $P(A = 0|y = 1) = P(B = 0|y = 1) = \frac{1}{3}$;

For any inpute $A = a, B = b$, the naive classifier predicts y that maximizes $P(A = a|y)P(B = b|y)P(y)$. Hence, the classifier will predict y-values as follows:
$A = 0, B = 0$: $P(A = 0|y = 0)P(B = 0|y = 0)P(y = 0) = 1/4$, $P(A = 0|y = 1)P(B = 0|y = 1)P(y = 1) = 1/12$, prediction: $y = 0$.
$A = 0, B = 1$: $P(A = 0|y = 0)P(B = 1|y = 0)P(y = 0) = 0$, $P(A = 0|y = 1)P(B = 1|y = 1)P(y = 1) = 1/6$, $y = 1$ is predicted.
$A = 1, B = 0$: $P(A = 1|y = 0)P(B = 0|y = 0)P(y = 0) = 0$, $P(A = 1|y = 1)P(B = 0|y = 1)P(y = 1) = 1/6$, $y = 1$ is predicted.
$A = 1, B = 1$: $P(A = 1|y = 0)P(B = 1|y = 0)P(y = 0) = 0$, $P(A = 1|y = 1)P(B = 1|y = 1)P(y = 1) = 1/3$, $y = 1$ is predicted.
Thus, the classifier has zero error rate.

1.2 (10 pts) Suppose we have a function $y = (A \wedge B) \vee \neg(B \vee C)$, where A, B, C are again independent binary random variables, each having a 50% chance of being 0.
a) (3 pts) How many parameters a naive Bayes classifier needs to estimate (without counting $P(\neg x)$ as a parameter if $P(x)$ is already counted as an estimated parameter)? What will be the error rate of the naive Bayes classifier (assuming infinite training data)?

Answer: A naive classifier will need to estimate the probability of class $y = 0$: $P(y = 0)$, and the probabilities of input values within each class: $P(A = 0|y = 0)$, $P(B = 0|y = 0)$, $P(C = 0|y = 0)$, $P(A = 0|y = 1)$, $P(B = 0|y = 1)$, $P(C = 0|y = 1)$.
Altogether it is 7 parameters.
In general, for m input binary variables it would have been: $2m + 1$.

To compute error rate we can construct a Boolean table of the function and use it to estimate probabilities (since we assume infinite training data).

A	B	C	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

The estimated parameters are:
$P(y = 0) = 1/2$;
$P(A = 0|y = 0) = 3/4$, $P(B = 0|y = 0) = 1/2$, $P(C = 0|y = 0) = 1/4$
$P(A = 0|y = 1) = 1/4$, $P(B = 0|y = 1) = 1/2$, $P(C = 0|y = 1) = 3/4$

For any input $A = a, B = b, C = c$ the classifier predicts y that maximizes $P(A = a|y)P(B = b|y)P(C = c|y)P(y)$.
The predictions of the classifier are then as follows (assuming that in case of a tie it always predicts 1):
$A = 0, B = 0, C = 0$: $y_{pred} = 1$
$A = 0, B = 0, C = 1$: $y_{pred} = 0$
$A = 0, B = 1, C = 0$: $y_{pred} = 1$
$A = 0, B = 1, C = 1$: $y_{pred} = 0$
$A = 1, B = 0, C = 0$: $y_{pred} = 1$
$A = 1, B = 0, C = 1$: $y_{pred} = 1$
$A = 1, B = 1, C = 0$: $y_{pred} = 1$
$A = 1, B = 1, C = 1$: $y_{pred} = 1$

From this, we can compute the error rate as the number of mistakes over the number of possible inputs (since each input is equally likely): error rate $= 2/8 = 0.25$.

b) (3 pts) How many parameters a joint Bayes classifier needs to estimate? What will be the error rate of the joint Bayes classifier (assuming infinite training data)?

Answer: A joint classifier will need to estimate a probability of one of the two classes, say $P(y = 0)$, and probabilities for all possible inputs within each class (minus one within each class since the sum of all input value probabilities sums to one), that is: $P(A = 0, B = 0, C = 0|y = 0), P(A = 0, B = 0, C = 0|y = 1), P(A = 0, B = 0, C = 1|y = 0), P(A = 0, B = 0, C = 1|y = 1), \ldots P(A = 1, B = 1, C = 0|y = 0), P(A = 1, B = 1, C = 0|y = 1)$.

Thus, it is $1 + 2 * (2^3 - 1) = 15$.
In general, for m input binary random variables it would have been $1 + 2 * (2^m - 1)$.

The error rate of a joint Bayes classifier is zero (assuming infinite training data) since it can model an arbitrary complex Boolean function.

c) (4 pts) Suggest a Bayes classifier that will need to estimate less number of parameters than a joint Bayes classifier but will still have a zero error rate (assuming infinite training data). Show that the classifier has this error rate.

Answer: Consider a Bayes classifier that assumes that A is independent of C when conditioned on B and on y (unlike a naive Bayes classifier that assumes that A, B, C are all independent of each other when conditioned on y).

Then $P(A, B, C|y) = P(A, C|B, y)P(B|y) = P(A|B, y)P(C|B, y)P(B|y)$.
For any input $A = a, B = b, C = c$ the classifier then predicts y so as to maximize $P(A = a|B = b, y)P(C = c|B = b, y)P(B = b|y)P(y)$.

The parameters it needs to estimate are: $P(y = 0)$, $P(B = 0|y = 0)$, $P(B = 0|y = 1)$,
$P(A = 0|B = 0, y = 0)$, $P(A = 0|B = 0, y = 1)$, $P(A = 0|B = 1, y = 0)$, $P(A = 0|B = 1, y = 1)$,
$P(C = 0|B = 0, y = 0)$, $P(C = 0|B = 0, y = 1)$, $P(C = 0|B = 1, y = 0)$, $P(C = 0|B = 1, y = 1)$
Thus, there are 11 parameters to estimate.

Using the Boolean table above, we can estimate the parameters as follows: $P(y = 0) = 1/2$
$P(B = 0|y = 0) = 1/2$, $P(B = 0|y = 1) = 1/2$,
$P(A = 0|B = 0, y = 0) = 1/2$, $P(A = 0|B = 0, y = 1) = 1/2$, $P(A = 0|B = 1, y = 0) = 1$,
$P(A = 0|B = 1, y = 1) = 0$,
$P(C = 0|B = 0, y = 0) = 0$, $P(C = 0|B = 0, y = 1) = 1$, $P(C = 0|B = 1, y = 0) = 1/2$,
$P(C = 0|B = 1, y = 1) = 1/2$

The predictions of the classifier are then as follows (there are no ties):
$A = 0, B = 0, C = 0$: $y_{pred} = 1$
$A = 0, B = 0, C = 1$: $y_{pred} = 0$
$A = 0, B = 1, C = 0$: $y_{pred} = 0$
$A = 0, B = 1, C = 1$: $y_{pred} = 0$
$A = 1, B = 0, C = 0$: $y_{pred} = 1$
$A = 1, B = 0, C = 1$: $y_{pred} = 0$
$A = 1, B = 1, C = 0$: $y_{pred} = 1$
$A = 1, B = 1, C = 1$: $y_{pred} = 1$

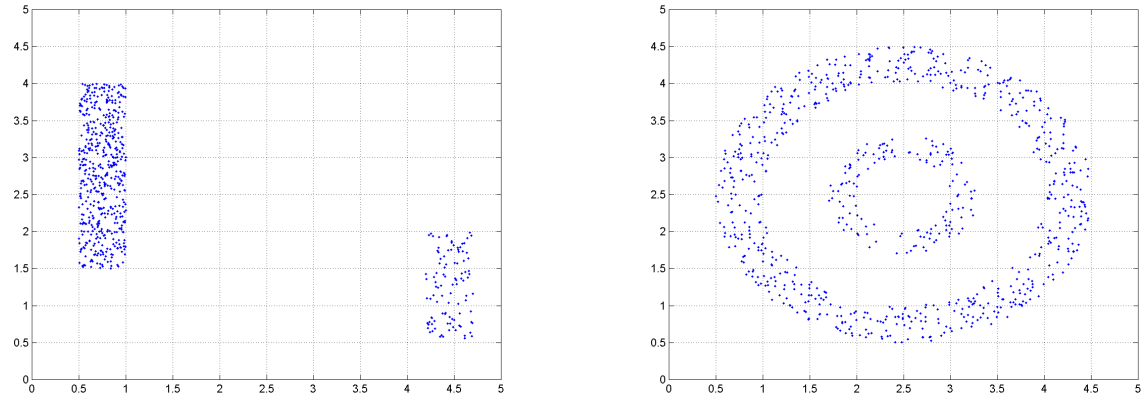
This corresponds to zero error rate.

Question 2. Spectral Clustering I (15 pts)

In this problem we will analyze the operation of one of the variants of spectral clustering methods on two datasets shown in Figure 1. For each of the datasets (unless directed otherwise) please answer the following questions.

2.1 (3 pts) The first step is to build an affinity matrix. The matrix defines the degree of similarity between points.
a) Suppose we use the L2 norm to construct the following affinity matrix (let x_i denote an ith datapoint):

$$A(i, j) = A(j, i) = \begin{cases} 1 & \text{if } |x_i - x_j|_2 < \theta \\ 0 & \text{otherwise} \end{cases} \quad (1)$$



(a) (b)

Figure 1: Plots for problem 2

What θ value would you choose and why?

b) Suppose instead we use Gaussian kernel for our affinity matrix:

$$A(i,j) = exp\left(-\frac{|x_i - x_j|_2}{2 * \sigma^2}\right) \tag{2}$$

What σ value would you choose and why?

Answer: In general, you want to choose such a parameter that the similarity between points in different clusters is 0 (or close to it), while the similarity between *neighboring* points in the same cluster is close to 1. The answers to this question are based purely on eye-balling. So, for the affinity matrix in (a) and the dataset in Figure 1(a) θ in between about 2.5 and 3 will result in an ideal case. For the dataset in Figure 1(b) it is less clear, but a value of 0.5, for example, will give us what we want.

For the Gaussian kernel, we want to set σ to obtain the same effect (even though we can not really achieve 0 similarity). So, for example, for the dataset in Figure 1(a) $\sigma = 0.5$ and for the dataset in Figure 1(a) $\sigma = 0.3$ should separate points reasonably.

2.2 (4 pts) The second step is to compute first k dominant eigenvectors of the affinity matrix, where k is the number of clusters we want to have
a) For the dataset in Figure 1(a) and the affinity matrix defined by equation 1 is there a value of θ for which you can compute analytically eigenvalues corresponding to the first two dominant eigenvectors? If not, explain why not. If yes, compute and write these eigenvalues down.

Answer: Yes, consider $\theta = 2.6$. It will result in the affinity matrix that is a block matrix: $A = \begin{pmatrix} 1_{nxn} & 0 \\ 0 & 1_{mxm} \end{pmatrix}$
where 1_{nxn} is a block of ones of size n by n, and 1_{mxm} a block of ones of size m by m. n is the number of points in left cluster, and m is the number of points in the right cluster.

Such matrix has one eigenvalue $\lambda_1 = n$ (with a corresponding eigenvector $v_1 = [1 \dots 1_n 0 \dots 0]^T$), and a second eigenvalue $\lambda_2 = m$ (with a corresponding eigenvector $v_2 = [0 \dots 0_n 1 \dots 1]^T$). These are in fact the only eigenvalues of A, since $rank(A)$ is clearly 2.

2.3 (4 pts) The third step is to cluster the rows of the matrix Y into k clusters using K-means (or a similar algorithm), where Y is constructed by placing k dominant eigenvectors into columns and re-normalizing the rows (to make each row a unit vector).
a) For the dataset in Figure 1(a) and the affinity matrix defined by equation 1 write down your best guess for the coordinates of $k = 2$ cluster centers.

Answer: Given the eigenvectors $v_1 = [1 \dots 1_n 0 \dots 0]^T$ and $v_2 = [0 \dots 0_n 1 \dots 1]^T$, we have: $Y = \begin{pmatrix} 1 & 0 \\ \vdots & \vdots \\ 1_n & 0_n \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{pmatrix}$

If we run K-means on the rows of Y, we get two clusters with centers at $< 1, 0 >$ and $< 0, 1 >$.

2.4 (4 pts) Finally, given the clusters on matrix Y, a point x_i is declared to be in cluster j iff the ith row of Y is in cluster j.
a) What are the final clusters you would expect to obtain for each of the datasets? Provide a rough sketch of the clusters to give an idea.

Answer: This should result in almost perfect clustering (that is, will cluster each connected component together).

b) What are the clusters you would expect to obtain if using EM algorithm for Gaussian Mixture Models with 2 clusters? Also provide a rough sketch of the clusters.

Answer: The dataset in Figure 1(a) should be clustered perfectly, while the dataset in Figure 1(b) will probably be split in two clusters - half of the inner and outer circles in one cluster, and the other halves of the circles in the second cluster. Thus, the dataset in Figure 1(b) is clustered incorrectly (assuming we want each circle to form its own cluster)

Question 3. Spectral Clustering II (15 pts)

The version of spectral clustering we have studied in class made use of matrix $A = D^{-1/2}WD^{-1/2}$. W is an affinity matrix with $w_{ij} = w_{ji}$ being a non-negative distance between points x_i and x_j. D is a diagonal matrix whose i^{th} diagonal element, d_{ii}, is the sum of W's i^{th} row. In the following you will need to prove several properties about A that are important for a good understanding of spectral clustering.

For the proofs you might find useful the following property: for any symmetric matrix B with all non-negative entries if u is an eigenvector with all positive entries, then no other independent eigenvector of B has the same eigenvalue.

3.1 (3 pts) Show that a vector $v_1 = [d_{11}^{\frac{1}{2}} d_{22}^{\frac{1}{2}} \dots d_{nn}^{\frac{1}{2}}]^T$ is an eigenvector of A with an eigenvalue $\lambda_1 = 1$.

Answer: We need to show that $Av_1 = v_1$.

$D^{-1/2}WD^{-1/2}v_1 =$
$D^{-1/2}WD^{-1/2}[d_{11}^{\frac{1}{2}} d_{22}^{\frac{1}{2}} \dots d_{nn}^{\frac{1}{2}}]^T =$
$D^{-1/2}WD^{-1/2}[d_{11}^{\frac{1}{2}} d_{22}^{\frac{1}{2}} \dots d_{nn}^{\frac{1}{2}}]^T =$
$D^{-1/2}W[11 \dots 1]^T =$
$\left[\frac{\sum_j (w_{1j})}{d_{11}^{\frac{1}{2}}} \dots \frac{\sum_j (w_{nj})}{d_{nn}^{\frac{1}{2}}}\right]^T =$
$\left[d_{11}^{1/2} \dots d_{nn}^{1/2}\right]^T =$
v_1

3.2 (4 pts) Prove that $\lambda_1 = 1$ is the largest eigenvalue of A.

Answer: We have just shown that A has an eigenvalue of $\lambda_v = 1$ with a corresponding fully positive eigenvector v. We need to show no eigenvector can have an eigenvalue larger than 1. We prove by contradiction. Suppose not and there exists an eigenvector u, s.t. $\lambda_u > 1$.

Let us derive a scaled version of this vector, $u' = c * u$, where $c = \min_{i,s.t.u_i>0} \frac{v_i}{u_i}$ (Such c exists because v is fully positive, u is orthogonal to v since A is symmetric, and therefore in order to have $u \cdot v = 0$, u must contain both negative and positive elements). Clearly, u' is still an eigenvector of A with the same eigenvalue. We also now have the following: $\forall i, u'_i \leq v_i$ and $\exists j, u'_j = v_j$. As a result, $v - u' \geq 0$, where 0 is a column vector of zeros. Consequently, we must have $A(v-u') \geq 0$ since A is non-negative.

On the other hand, $A(v - u') = Av - Au' = \lambda_v v - \lambda_u u' = v - \lambda_u u'$. However, since $\lambda_u > 1$, we will have $v_j - \lambda_u u'_j < 0$ for $j, u'_j = v_j$. Thus, $A(v - u') \ngeq 0$.

3.3 (4 pts) Prove that all eigenvectors orthogonal to v_1 will have an eigenvalue strictly smaller than 1.

This property shows that if points are viewed as vertices in a Markov graph with transition probabilities proportional to distances between points (elements of W), then v_1 is the only eigenvector needed to compute the probability distribution over states matrix P^∞ (whose $(ij)^{th}$ element shows the probability of being at state j if starting at state i) after infinitely many steps.

Answer: This follows directly from the hint and the property we proved in 3.2.

3.4 (4 pts) Show that $P^\infty = D^{-\frac{1}{2}}(v_1 v_1^T)D^{\frac{1}{2}}$, where $P = D^{-1}W$ is the probability transition matrix.

Answer: $P^\infty =$
$D^{-\frac{1}{2}}(A^\infty)D^{\frac{1}{2}} =$
$D^{-\frac{1}{2}}(\lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \ldots)^\infty D^{\frac{1}{2}}$.

Since the eigenvectors of a symmetric matrix A can always be made orthonormal, the dot-product of any two distinct eigenvectors will be 0 and the dot-product of same eigenvectors will be 1. Hence, $(\lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \ldots)^\infty = (\lambda_1^\infty v_1 v_1^T + \lambda_2^\infty v_2 v_2^T + \ldots)$. Finally, since $\forall i > 1, |\lambda_i < 1|$, $P^\infty = D^{-\frac{1}{2}}(v_1 v_1^T)D^{\frac{1}{2}}$. (Here, we took a bit of a shortcut: in 3.4 we have proved that $\lambda_i < 1$, but the fact that $\lambda_i > -1$ follows exactly from the same arguments as in 3.2 except using $\max_{i,s.t.u_i<0}$ instead of $\min_{i,s.t.u_i>0}$ in the computations of c)

Question 4. Classifiers (40 pts)

In this problem you will implement two common classifiers and use them to predict attribute values for a series of data. The data you will be working with is a series of records containing 4 continuous-valued attributes and one Boolean class attribute. You will be attempting to predict the Boolean attribute value based on the values of the first four.

All implementation should be done in Matlab. For each problem, we specify the prototype of the required function(s). Please follow these prototypes! It should make your life easier and keep your TA's from getting crotchety.

4.1 (12 pts) Create a naïve Bayes classifier with the assumption that each attribute value for a particular record is generated from a Gaussian with μ and σ determined by the class of the record. In other words, if a record is from class 1, then its first four attributes (a_1, a_2, a_3, a_4) will have their values generated from a Gaussian centered at μ_1 with diagonal covariance matrix Σ_1. You're going to use this assumption to classify a series of new records.

Here is the prototype of the matlab function you need to implement:

$$\textbf{function } percent\_correct = \textbf{gaussian\_naive\_classify}(training\_data, testing\_data) \qquad (3)$$

training_data is Gaussian-generated input data (with an arbitrary number of rows), where each row contains 4 continuous values and one Boolean value. *testing_data* is of the same format. *percent_correct* is the percent of records from *testing_data* whose classes were accurately predicted.

Report the accuracy of your classifier when using the training data given in 'ind_training_data.txt' and testing data in 'ind_testing_data.txt'.

Answer: The classifier should have an accuracy of 96%

4.2 (13 pts) Create a joint classifier with the assumption that each record has its attribute vector generated from a single multi-dimensional Gaussian with mean vector and covariance matrix determined by the class of the record. So, if a record is from class 1, then its attribute vector (a_1, a_2, a_3, a_4) is generated from a Gaussian centered at μ_1 with a full (non-diagonal) covariance matrix Σ_1.

Here is the prototype of the matlab function you need to implement:

$$\textbf{function } \textit{percent\_correct} = \textbf{gaussian\_joint\_classify}(\textit{training\_data}, \textit{testing\_data}) \tag{4}$$

again, *training_data* is input data (with an arbitrary number of rows), where each row contains 4 continuous values and one Boolean value. *testing_data* is of the same format. *percent_correct* is the percent of records from *testing_data* whose classes were accurately predicted.

Report the accuracy of your classifier when using the training data given in 'dep_training_data.txt' and testing data in 'dep_testing_data.txt'.

Answer: The classifier should have an accuracy of 98%

4.3 (15 pts) Now find a general algorithm for performing classification that assumes each attribute is generated independently, but each from a different arbitrary distribution based on the class of the record. Your approach should use a set of training data to optimally split each attribute's continuous space into a collection of non-overlapping intervals, with each interval having an associated class prediction.

For each attribute, you should find the number of intervals (at most 3) that provides the best class prediction over the training data. These intervals can then be used for classifying new records for testing.

[Hint: from training data we can compute $P(class = 1 \mid a_i \in X)$ for each interval X associated with attribute a_i. When given a new record, we can use the independence assumption to multiply these probabilities together and see which class is more likely overall.]

Here is the prototype of the matlab function you need to implement:

$$\textbf{function } \textit{percent\_correct} = \textbf{general\_naive\_classify}(\textit{training\_data}, \textit{testing\_data}) \tag{5}$$

again, *training_data* is input data (with an arbitrary number of rows), where each row contains 4 continuous values and one Boolean value (as in training_data.txt). *testing_data* is of the same format. *percent_correct* is the percent of records from *testing_data* whose classes were accurately predicted.

Report the accuracy of your classifier when using the training data given in 'ind_training_data.txt' and testing data in 'ind_testing_data.txt'. Also report the number of intervals used for each attribute. Why is this number not the same for each attribute, and what does this mean about the underlying Gaussian distributions used to generate each attribute's value? How does the performance of this approach compare with the naïve Gaussian classifier from 4.1? Why is it different/the same?

Answer: The method hinted at is as follows.

For each attribute a_i (from $a_1 to a_4$), do the following:

(a) Sort the records by attribute a_i.

(b) Find the best value of a_i to use to split the records into two intervals. To do this, manually try each possible value given in the records as the split value s_i, and look at the classification accuracy when all records with $a_i < s_i$ are assigned the class 1 and all records with $a_i > s_i$ are assigned the class 0 (and vice versa).

(c) Do the same for two split points.

(d) Select the split point(s) and classification which gave the maximum accuracy. Record the classification accuracies: $P(class = 1 | a_i \leq s_i)$, $P(class = 1 | a_i > s_i)$.

Once we have these probability measures, we can predict the class of any new record pretty easily. The math (for two attributes) is as follows:

$$
\begin{aligned}
P(class = 1 | a_1 \leq s_1 \wedge a_2 > s_2) &= P(a_1 \leq s_1 \wedge a_2 > s_2 \wedge class = 1) \\
&= P(a_1 \leq s_1 | a_2 > s_2 \wedge class = 1) \cdot P(a_2 > s_2 | class = 1) \\
&\quad \cdot P(class = 1) / P(a_1 \leq s_1 \wedge a_2 > s_2) \\
&= [P(a_1 \leq s_1 | class = 1)] \cdot P(a_2 > s_2 | class = 1) \\
&\quad \cdot P(class = 1) / P(a_1 \leq s_1 \wedge a_2 > s_2) \\
&= [P(class = 1 | a_1 \leq s_1) \cdot P(a_1 \leq s_1) / P(class = 1)] \cdot \ldots \\
&= P(class = 1 | a_1 \leq s_1) \cdot P(class = 1 | a_2 > s_2) / P(class = 1)^2
\end{aligned}
$$

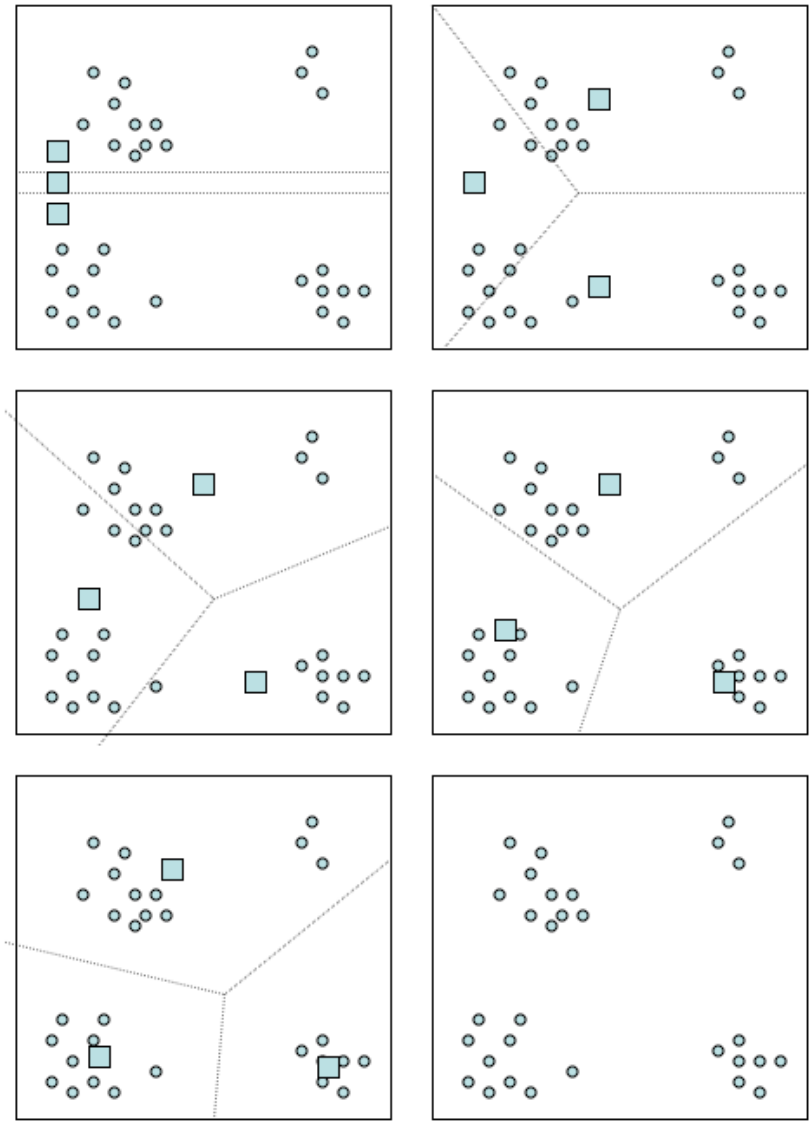
And since we have $P(class = 1) = P(class = 0)$ in our training set, we can remove the denominator, so that we can just multiply the individual $P(class = 1 | a_1 \leq s_1)$ terms together and see which class gives the higher overall result.

If the method hinted at was implemented, the classifier should have an accuracy of 82%. 2 intervals are used for attributes 1, 3, and 4, and 3 intervals are used for attribute 2. This number differs depending on how the two Gaussians used to generate attribute a_i interact. If the variance of each Gaussian (class 1 and class 0) differs, it is possible that one class could "straddle" the other, so that one class is more likely near its mean but the other class is more likely on either side of this. This results in 3 intervals. If the variance of each Gaussian is the same, then only two intervals will be needed.

The interval classifier does worse than the independent Gaussian classifier because we are assuming uniform probability across each interval. The Gaussian classifier is able to distinguish between records within the same interval and assign them different probabilities of classification. In the same way, it is also able to blur the boundaries of the intervals.

Question 5. K-means and Hierarchical Clustering (15 pts)

5.1 (5 pts) Run K-means manually on the following dataset. Circles are data points and squares are the initial cluster centers. Draw the cluster centers and the decision boundaries that define each cluster. If no points belong to a particular cluster, assume its center does not change. Use as many of the pictures as you need for convergence.



5.2 (5 pts) Using hierarchical clustering (with the minimum distance criteria), what is the maximum height of the hierarchy tree required to cluster a set of N points? What is the minimum height? Give an example of a collection of 16 points that requires (i) a hierarchy tree of maximum height, and (ii) a hierarchy tree of minimum height.

Answer: Worst case: $N-1$**. Best case:** $ceil(log_2 N)$**. Example of set of points requiring hierarchy tree of maximum height: points on a line, where separation between adjacent points increases (e.g. on** x **axis with** x **values** $0, 1, 4, 9, 16, 25, \ldots$**). Example of set of points requiring hierarchy tree of minimum height:** $1, 2, 4, 5, 8, 9, 11, 12, 101, 102, 104, 105, 108, 109, 111, 112$**.**

5.3 (a) (3 pts) Is it possible to have Gaussian Mixture Models exhibit equivalent behavior to K-means by restricting the Gaussians used? How?

Yes. If we restrict the covariance matrices to be diagonal, with each diagonal element being the same σ^2 value (with $\sigma \to 0$), then Gaussian Mixture Models will behave like K-means.

(b) (2 pts) For what sort of data do general Gaussian Mixture Models produce much better results than K-means? Provide an example of such a dataset.

Answer: Any data that contains non-spherical clusters. Two long skinny rectangular classes is a classic example.

Assignment 4 Solutions: Decision Trees, Neural Nets, Instance-based Learning, SVMs

10-701/15-781: Machine Learning (Fall 2004)

Out: Oct. 28th, 2004
Due: Nov. 16th 2004, **Tuesday**, Start of class,

Question 1. Decision Trees

1.1 (3 pts) If we have a Decision Tree where we have split on some attribute a_i followed by a split on attribute a_j, is this tree equivalent to that produced by reversing the order of these attributes (i.e. by splitting on a_j then a_i)? Very briefly explain why this is or is not the case.

Answer: Assuming we are dealing with binary attributes and our tree splits on both with no data falling in between, then the trees will be equivalent. We are computing an AND function combining the two attribute splits, so the result is independent of the split ordering.

1.2 (5 pts) The efficiency of the Decision Tree algorithm relies heavily on its use of a greedy selection mechanism for deciding which attribute to split on. If a particular dataset has A Boolean attributes, calculate the number of information gain computations necessary to create a Decision Tree to classify the data. Assume for simplicity that all attributes are useful for classifying any single data record, and all possible data records are in the training set.

Answer: We have A info gain computations to determine root, then $A-1$ to determine *each* child of the root, and so on down the tree. At the leaves, we don't need to do the final info gain computation when only one attribute is left (but we didn't take points off if you missed this). Answer is then: $A + 2 \cdot (A-1) + 4 \cdot (A-2) + 8 \cdot (A-3) \ldots = \sum_{i=0}^{A-2} (2^i \cdot (A-i))$.

1.3 (3 pts) It's possible to improve the basic Decision Tree algorithm by having it perform limited lookahead, in effect acting less greedily. For a two-step lookahead, it would consider the best way to split the children of each attribute we are considering to split on next. It would then split on the attribute whose children produced the best combined information gain.

For instance, suppose we are considering splitting on attribute a_i. A two-step lookahead information gain calculation would compute:

$$IG_{twostep}(a_i) = \max_{a_l, a_r} \{ \frac{n_l}{n_l + n_r} IG(D_l, a_l) + \frac{n_r}{n_l + n_r} IG(D_r, a_r) \}$$

where a_l and a_r are the left and right attribute children of a_i, n_l and n_r are the number of records in the left and right children of a_i, and D_l and D_r are the sets of records themselves. $IG(D_l, a_l)$ is the information gain associated with attribute a_l over set D_l.

Briefly explain why the terms $\frac{n_l}{n_l+n_r}$ and $\frac{n_r}{n_l+n_r}$ are needed in the above equation.

Answer: These factors just account for the different relative sizes of the subsets in the partition, weighting larger subsets proportionally more. This tends to result in more balanced splits, i.e. tiny subsets having high purity don't affect the split decision unduly.

1.4 (4 pts) If we have A Boolean attributes, how many standard information gain computations need to be carried out to determine how to split the root node using the approach in 1.3? In other words, how many calls to $IG(\ldots,\ldots)$ must be made?

Answer: $A \cdot 2(A-1) = 2 \cdot A^2 - 2 \cdot A$

1.5 (5 pts) Let's look at just how much more expensive this really is. Making the same simplifying assumptions as in 1.2 and assuming we have a dataset with 10 attributes, how many levels can we compute of our standard Decision Tree for the same computation required to find the root of our two step lookahead tree? What is the equation that needs to be solved to find this number of levels for m attributes?

Answer: We can compute 4 levels. General equation to be solved is argmax_{l-1} such that $2 \cdot A(A-1) > \sum_0^l 2^l \cdot (A-l)$.

1.6 (5 pts) Does this two step lookahead approach provide a richer model class (hypothesis language)? In other words, are we able to represent new classification functions that were not representable with standard Decision Trees? Why or why not? How much does this change as we look more and more steps ahead, right up to the point where we are computing optimal trees?

Answer: No, the model class is identical. The trees cannot fundamentally represent any different functions. However, the final choice of tree in the same model class is likely to be better by doing lookahead.

Question 2. Neural networks

In this problem you'll compare the Gradient Descent training algorithm used in class with one other training algorithm of your choosing, on a particular function approximation problem. For this problem, the idea is to familiarize yourself with Gradient Descent and at least one other numerical solution technique. We expect you to spend between 1.5 and 4 hours on this problem; if you find you are well outside of this interval come see one of us.

The dataset 'data.txt' contains a series of (x, y) records, where $x \in [0, 5]$ and y is a function of x given by $y = a\sin(bx) + w$, where a and b are parameters to be learned and w is a noise term such that $w \sim N(0, \sigma^2)$. We want to learn from the data the best values of a and b to minimize the sum of squared error:

$$\underset{a,b}{\text{argmin}} \sum_{i=1}^{n} (y_i - a\sin(bx_i))^2 \tag{1}$$

Use any programming language of your choice and implement two training techniques to learn these parameters. The first technique should be Gradient Descent with a fixed learning rate, as discussed in class. The second can be any of the other numerical solutions listed in class: Levenberg-Marquardt, Newton's Method, Conjugate Gradient, Gradient Descent with dynamic learning rate and/or momentum considerations, or one of your own choice not mentioned in class.

You may want to look at a scatterplot of the data to get rough initial values for the parameters a and b. If you are getting a large sum of squared error after convergence (where large means > 100), you may want to try random restarts.

Write a short report detailing the method you chose and its relative performance in comparison to standard Gradient Descent (report the final solution obtained (values of a and b) and some measure of the computation required to reach it and/or the resistance of the approach to local minima). If possible, explain the difference in performance based on the algorithmic difference between the two approaches you implemented and the function being learned.

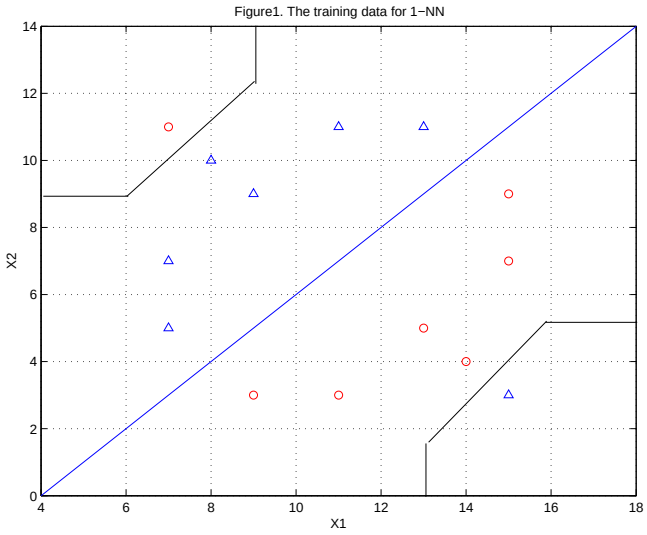
Question 3. K-Nearest-Neighbor classifier

We covered K-nearest-neighbor regression in the class. In this problem, you will get some experience for k-nearest-neighbor classification.

3.1 Let us first do a simple task manually by the 1-nearest-neighbor classifier.

Basically the classifier finds the closest (according to some distance metric) training-point to the unknown point and predicts the category of that training-point for this unknown point.

Answer:



3.4 For k = 2, 4, 6, .., You may encounter ties in the classification. Describe how you handle this situation in your above implementation.

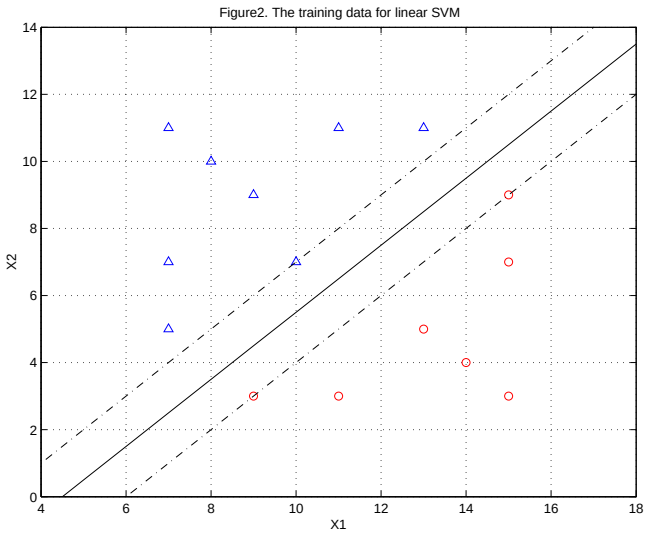
Answer: There are many possible ways to handle this tie case. For example, [1]. choose one of the class; [2]. use k-1 neighbor to decide; [3]. weighted kNN, etc.

3.5 - 3.7 Choose k and compare the four performance curves. Can you get some conclusion about the difference between 'train-test' 'v-folds cross-validation' and leave-one-out cross validation ?

Answer: We would get four curves roughly having the similar trend. The best error rate is around 0.02. If you run the program several times. You would find that LOOCV curve would be the same among multiple runs, because it does not have randomness involved. CV curves varies roughly around the LOOCV curve. "Train-test" test curve varies a lot among different runs. But anyway, roughly, as k increases, the error rate increases. From the curve, we can actually choose a small range of k (1 5) as our model selection result.

Question 4. Support Vector Machine

4.1 Let us first manually apply linear SVM in a simple case.



Answer:
Three support vectors. The decision boundary: $x_1 = x_2 - 4.5$.

4.2 Nonparametric representation in SVM
So how many dot products must be done for M test points? In the case of [4.1], how many dot products needed to classify M test points?

Answer: Suppose N_s represents the number of support vectors. The answer is : $M * N_s$.

4.3 Kernel trick in SVM.
For this radial-basis-style kernel, how to choose the number of centers ? (We know that a RBF model's centers are on each of the training data.)

Answer: Suppose N_s represents the number of support vectors. The answer is : N_s.

4.4 Maximum margin and quadratic programming in SVM
- Why is SVM training a quadratic programming (QP) rather than a linear programming (LP) problem.
- For the separable data case, how many variables are in the QP problem? How many constraints? In terms of our simple data in [4.1], what will be the numbers ?
- For the noisy (not separable) data case: How many variables are in the QP problem? How many constraints?

Answer:
- In the optimization problem solved in SVM training, there is a quadratic term in the objective function, $w^T w$, corresponding to the margin width. Quadratic programming optimizes quadratic objective functions. Linear programming optimizes only linear objective functions. Note that LP optimization methods are much faster than those available for QP.
- For separable case. Suppose the feature dimension is D, then D+1 variable. Suppose R records, R constraints.
- For noisy case. Suppose the feature dimension is D, then D+1+R variable. Suppose R records, 2R constraints.

4.5 Now let us explore the effect of basis functions on the maximum margin solution. Consider a simple one dimensional case, where we have only two training examples:

$$(x_1 = 0, y_1 = -1), (x_2 = \sqrt{2}, y_2 = 1) \tag{2}$$

Now we use a second order polynomial kernel, that means, mapping each input data x to a

$$\Phi(x) = [1, \sqrt{2}x, x^2]^T \tag{3}$$

Now we would like to find and understand the maximum margin solution: $\hat{W_1} = [\hat{\omega_1}, \hat{\omega_2}, \hat{\omega_3}]$ and $\hat{\omega_0}$ to:

$$min\|W_1\|^2 \tag{4}$$

Subject to:

$$y_1 * [\omega_0 + \Phi(x_1)^T W_1] \geqslant 1 \tag{5}$$

$$y_2 * [\omega_0 + \Phi(x_2)^T W_1] \geqslant 1 \tag{6}$$

Please report your derivations along with the specific answers.

- Using your knowledge of the maximum margin boundary, write down a vector that points in the same direction as $\hat{W_1}$
- What is the value of the margin that we can achieve in this case?
- By relating the margin and $\hat{W_1}$, provide the solution of $\hat{W_1}$.
- When we know $\hat{W_1}$, what is value of $\hat{\omega_0}$ then?

Answer:
Since there is only two examples in our data sets, one positive and one negative, they must be the support vectors that decide the decision boundary. Input x_1 and x_2 mapped to feature vectors $\Phi(x_1) = [1\ 0\ 0\ ]^T$ and $\Phi(x_2) = [1\ 2\ 2\ ]^T$

- So the direction of $\hat{W_1}$ is the same as the direction of a vector from the negative example to the positive example, in the feature space. Thus direction of $\hat{W_1} = [0\ 2\ 2]^T$
- Margin = distance from each support vector to decision boundary = $\sqrt{2}$ (only two training examples here)
- Since $\|\hat{W_1}\| = \frac{1}{margin}$. So now we know the norm and direction of our weight vector. We can calculate the weight vector as $\hat{W_1} = [0\ \frac{1}{2}\ \frac{1}{2}\ ]^T$.
- Due to the constraints satisfied at the support vectors, we have:

$$-1 * [\omega_0 + [1, 0, 0]^T W_1] = 1 \tag{7}$$

$$1 * [\omega_0 + [1, 2, 2]^T W_1] = 1 \tag{8}$$

We get ω_0 = -1.

Assignment 4: Decision Trees, Neural Nets, Instance-based Learning, SVMs

10-701/15-781: Machine Learning (Fall 2004)

Out: Oct. 28th, 2004
Due: Nov. 16th 2004, * **Tuesday** * , Start of class,

a *This assignment has four problems to test your understanding about Decision Trees, Neural Nets, Instance-based Learning, and Support Vector Machines. Each problem is worth 25%.*

b *For question 2, you can use whatever programming language you like. For question 3, please use Matlab. For both questions, you need to submit your code to the TAs (we'll provide instructions on how soon).*

c *For questions and clarifications, contact Dave (questions 1 and 2) (dif+781@cmu.edu) or Yanjun (questions 3 and 4) (qyj@cs.cmu.edu).*

d *Policy on collaboration:*

Homeworks will be done individually: each student must hand in their own answers. It is acceptable, however, for students to collaborate in figuring out answers and helping each other solve the problems. We will be assuming that, as participants in a graduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.

d *Policy on late homework:*

Homework is worth full credit at the beginning of class on the due date. It is worth half credit for the next 48 hours. It is worth zero credit after that. You must turn in all of the 4 homeworks, even if for zero credit, in order to pass the course.

Question 1. Decision Trees

1.1 If we have a Decision Tree where we have split on some attribute a_i followed by a split on attribute a_j, is this tree equivalent to that produced by reversing the order of these attributes (i.e. by splitting on a_j then a_i)? Very briefly explain why this is or is not the case.

1.2 The efficiency of the Decision Tree algorithm relies heavily on its use of a greedy selection mechanism for deciding which attribute to split on. If a particular dataset has A Boolean attributes, calculate the number of information gain computations necessary to create a Decision Tree to classify the data. Assume for simplicity that all attributes are useful for classifying any single data record, and all possible data records are in the training set.

1.3 It's possible to improve the basic Decision Tree algorithm by having it perform limited lookahead, in effect acting less greedily. For a two-step lookahead, it would consider the best way to split the children of each attribute we are considering to split on next. It would then split on the attribute whose children produced the best combined information gain.

For instance, suppose we are considering splitting on attribute a_i. A two-step lookahead information gain calculation would compute:

$$IG_{twostep}(a_i) = \max_{a_l, a_r} \{ \frac{n_l}{n_l + n_r} IG(D_l, a_l) + \frac{n_r}{n_l + n_r} IG(D_r, a_r) \}$$

where a_l and a_r are the left and right attribute children of a_i, n_l and n_r are the number of records in the left and right children of a_i, and D_l and D_r are the sets of records themselves. $IG(D_l, a_l)$ is the information gain associated with attribute a_l over set D_l.

Briefly explain why the terms $\frac{n_l}{n_l+n_r}$ and $\frac{n_r}{n_l+n_r}$ are needed in the above equation.

1.4 If we have A Boolean attributes, how many standard information gain computations need to be carried out to determine how to split the root node using the approach in 1.3? In other words, how many calls to $IG(\ldots,\ldots)$ must be made?

1.5 Let's look at just how much more expensive this really is. Making the same simplifying assumptions as in 1.2 and assuming we have a dataset with 10 attributes, how many levels can we compute of our standard Decision Tree for the same computation required to find the root of our two step lookahead tree? What is the equation that needs to be solved to find this number of levels for m attributes?

1.6 Does this two step lookahead approach provide a richer model class (hypothesis language)? In other words, are we able to represent new classification functions that were not representable with standard Decision Trees? Why or why not? How much does this change as we look more and more steps ahead, right up to the point where we are computing optimal trees?

Question 2. Neural networks

In this problem you'll compare the Gradient Descent training algorithm used in class with one other training algorithm of your choosing, on a particular function approximation problem. For this problem, the idea is to familiarize yourself with Gradient Descent and at least one other numerical solution technique. We expect you to spend between 1.5 and 4 hours on this problem; if you find you are well outside of this interval come see one of us.

The dataset 'data.txt' contains a series of (x, y) records, where $x \in [0, 5]$ and y is a function of x given by $y = a\sin(bx) + w$, where a and b are parameters to be learned and w is a noise term such that $w \sim N(0, \sigma^2)$. We want to learn from the data the best values of a and b to minimize the sum of squared error:

$$\underset{a,b}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - a\sin(bx_i))^2 \tag{1}$$

Use any programming language of your choice and implement two training techniques to learn these parameters. The first technique should be Gradient Descent with a fixed learning rate, as discussed in class. The second can be any of the other numerical solutions listed in class: Levenberg-Marquardt, Newton's Method, Conjugate Gradient, Gradient Descent with dynamic learning rate and/or momentum considerations, or one of your own choice not mentioned in class.

You may want to look at a scatterplot of the data to get rough initial values for the parameters a and b. If you are getting a large sum of squared error after convergence (where large means > 100), you may want to try random restarts.

Write a short report detailing the method you chose and its relative performance in comparison to standard Gradient Descent (report the final solution obtained (values of a and b) and some measure of the computation required to reach it and/or the resistance of the approach to local minima). If possible, explain the difference in performance based on the algorithmic difference between the two approaches you implemented and the function being learned.

Question 3. K-Nearest-Neighbor classifier

We covered K-nearest-neighbor regression in the class. In this problem, you will get some experience for k-nearest-neighbor classification.

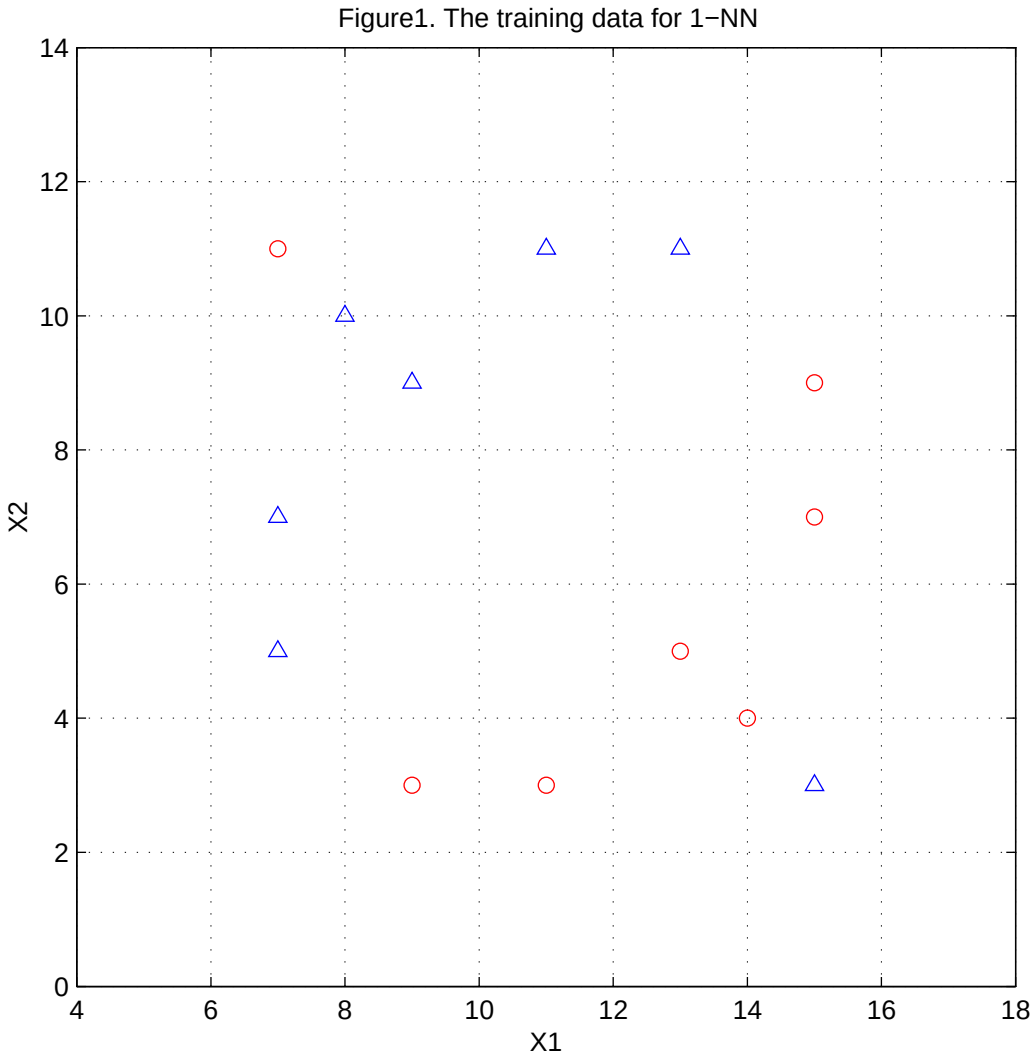
3.1 Let us first do a simple task manually by the 1-nearest-neighbor classifier.

Basically the classifier finds the closest (according to some distance metric) training-point to the unknown point and predicts the category of that training-point for this unknown point.

We have two input attributes ($x_1, x_2 \in \Re$) and one output attribute $y \in \{\triangle, o\}$. So, each example is a triple $< x_1, x_2, y >$. The training set looks like:

$Point1 =$ < 7, 11, o >
$Point2 =$ <11, 11, $\triangle$ >
$Point3 =$ <13, 11, $\triangle$ >
$Point4 =$ <8, 10, $\triangle$ >
$Point5 =$ <9, 9, $\triangle$ >
$Point6 =$ <15, 9, o>
$Point7 =$ <7, 7, $\triangle$ >
$Point8 =$ <15, 7, o>
$Point9 =$ <7, 5, $\triangle$ >
$Point10 =$ <13, 5, o>
$Point11 =$ <14, 4, o>
$Point12 =$ <9, 3, o>
$Point13 =$ <11, 3, o>
$Point14 =$ <15, 3, $\triangle$ >

Graphically, these points look like figure 1.



Evaluate the decision boundaries for 1-nearest-neighbor classification on this data. Assume the classifier use Euclidean distance metric and outputs either '△' or 'o'.
The boundary does not need to be exact - a photocopy of figure 1 with boundaries drawn approximately is sufficient. Make sure to label the regions with the classification label that would be given. (The image file is also provided separately for your convenience)

3.2 For problem [3.3] to [3.8], you need to implement a classifier in matlab and test your classifier on a real data set. The data was generated from handwritten digits, automatically scanned from envelopes by the U.S. Postal Service.

Please download the data file 'knn.data' from the course web page.
It contains 364 points. In each row, the first attribute is the class label (0 or 1), the remaining 256 attributes are features (all columns are continuous values). (You could use Matlab function load('knn.data') to load this data into matlab.)

(We understand some students are not comfortable with matlab programming. But for this classifier, it is very simple to be implemented in matlab.)

3.3 Now you will implement a k-nearest-neighbor (kNN) classifier using Matlab. For each unknown example, the kNN classifier collects its K nearest neighbors training points, and then takes the most common category among these K neighbors as the predicted label of the test point.

We assume our classification is a binary classification task. The class label would be either 0 or 1. The classifier uses Euclidean distance metric. (But you should keep in mind that normal kNN classifier supports multi-class classification.)

Here is the prototype of the matlab function you need to implement:

$$function[Y\_test] = knn(k, X\_train, Y\_train, X\_test); \tag{2}$$

X_train contains the features of the training points, where each row is a 256-dimensional vector. Y_train contains the known labels of the training points, where each row is an 1-dimensional integer either 0 or 1 . X_test contains the features of the testing points, where each row is a 256-dimensional vector. k is the number of nearest-neighbors we would consider in the classification process.

3.4 For k = 2, 4, 6, .., You may encounter ties in the classification. Describe how you handle this situation in your above implementation.

3.5 The choice of k is essential in building the KNN model. In fact, k can be regarded as one of the most important factors of the model that can strongly influence the quality of predictions.

One simple way to find k is to use 'train-test' style. Randomly choose 30% of your data to be a test set. The remainder is a training set. Build the classification model on the training set and estimate the future performance with the test set. Try different values of k to find which k works best for the testing set.

Here we use 'error rate' to measure the performance of a classifier. It equals to the percentage of incorrectly classified cases on a test set.

Please implement a matlab function to implement the above 'train-test' way for finding a good k for the kNN classifier.

Here is the prototype of the matlab function you need to implement:

$$function[TestsetErrorRate, TrainsetErrorRate] = knn\_train\_test(kArrayToTry, XData, YData); \tag{3}$$

$XData$ contains the features of the data points, where each row is a 256-dimensional vector. $YData$ contains the known labels of the points, where each row is an 1-dimensional integer either 0 or 1 $kArrayToTry$ is a $K * 1$ column vector, containing the K possible values of k you want to try. $TestsetErrorRate$ is a $K * 1$ column vector containing the testing error rate for each possible k. $TrainsetErrorRate$ is a $K * 1$ column vector containing the training error rate for each possible k.

Then test your function knn_train_test on data set 'knn.data'.

Report the plot of 'train error rate' vs. 'k' and the plot of 'test error rate' vs. 'k' for this data. (Make these two curves together in one Figure . You could use 'hold on' function in matlab to help you.) What is the best k you would choose according to these two plots?

3.6 Instead of the above 'train-test' style, we could also do 'v-folds Cross-validation' to find the best k.

v-folds Cross-validation is a well established technique that can be used to obtain estimates of model parameters that are unknown. The general idea of this method is to divide the data sample into a number of v folds (randomly drawn, disjointed sub-samples or segments). For a fixed value of k, we apply the KNN model to make predictions on the i_{th} segment (i.e., use the v-1 segments as the train examples) and evaluate the error. This process is then successively applied to all possible choices of i ($i \in \{1, ..., v\}$). At the end of the v folds (cycles), the computed errors are averaged to yield a measure of the stability of the model (how well the model predicts query points). The above steps are then repeated for various k and the value achieving the lowest error rate is then selected as the optimal value for k (optimal in a cross-validation sense).

(If you want to understand more about cross validation, please look at Andrew's Cross-Validation slides online: http://www-2.cs.cmu.edu/ awm/tutorials/overfit.html)

Then please implement a cross-validation function to choose k. Here is the prototype of the matlab function you need to implement:

$$function[cvErrorRate] = knn\_cv(kArrayToTry, XData, YData, numCVFolds); \quad (4)$$

All the dimensionality of input parameters are the same as [3.5]. $cvErrorRate$ is a $K*1$ column vector containing the cross validation error rate for each possible k.

Apply this function on the data set 'knn.data' using 10-cross-folds. Report a performance curve of 'cross validation error rate' vs. 'k'. What is the best k you would choose according to this curve?

3.7 Besides 'train-test' style and v-folds cross validation, we could also use 'leave-one-out Cross-validation (LOOCV)' to find the best k.

LOOCV means omitting each training case in turn and train the classifier model on the remaining R-1 datapoints, test on this omitted training case. When you've done all points, report the mean error rate.

Implement a LOOCV function to choose k for our kNN classifier. Here is the prototype of the matlab function you need to implement:

$$function[LoocvErrorRate] = knn\_loocv(kArrayToTry, XData, YData); \quad (5)$$

All the dimensionality of input parameters are the same as [3.5]. $LoocvErrorRate$ is a $K*1$ column vector containing LOOCV error rate for each possible k.

Apply this function on the data set 'knn.data' and report the performance curve of 'LOOCV error rate' vs. 'k'. What is the best k you would choose according to this curve?

3.8 Compare the four performance curves (from [3.4] [3.5] and [3.6]). (Make the four curves together in one Figure here.) Can you get some conclusion about the difference between 'train-test' 'v-folds cross-validation' and leave-one-out cross validation ?

Note:
We provide a matlab file 'TestKnnMain.m' to help you test the above functions. You could download it from the course web site.

Question 4. Support Vector Machine

SVM is a popular classification tool. It is composed of a number of methods, each approaching a different aspect of the learning problem: SVM = nonparametric representation + kernel trick + maximum margin + quadratic programming. We would try to cover most of them in the following questions.

4.1 Let us first manually apply linear SVM in a simple case.

Similar as [3.1], we have two input attributes ($x_1, x_2 \in \Re$) and one output attribute $y \in \{\triangle, o\}$. So, each example is a triple $< x_1, x_2, y >$.

The training set looks like:
$Point1 = <$ 7, 11, $\triangle >$
$Point2 =$ <11, 11, $\triangle >$
$Point3 =$ <13, 11, $\triangle >$

$Point4$ = <8, 10, $\triangle$ >
$Point5$ = <9, 9, $\triangle$ >
$Point6$ = <15, 9, o>
$Point7$ = <7, 7, $\triangle$ >
$Point8$ = <15, 7, o>
$Point9$ = <7, 5, $\triangle$ >
$Point10$ = <13, 5, o>
$Point11$ = <14, 4, o>
$Point12$ = <9, 3, o>
$Point13$ = <11, 3, o>
$Point14$ = <15, 3, o>
$Point15$ = <10, 7, $\triangle$ >

Graphically, these points look like figure 2.

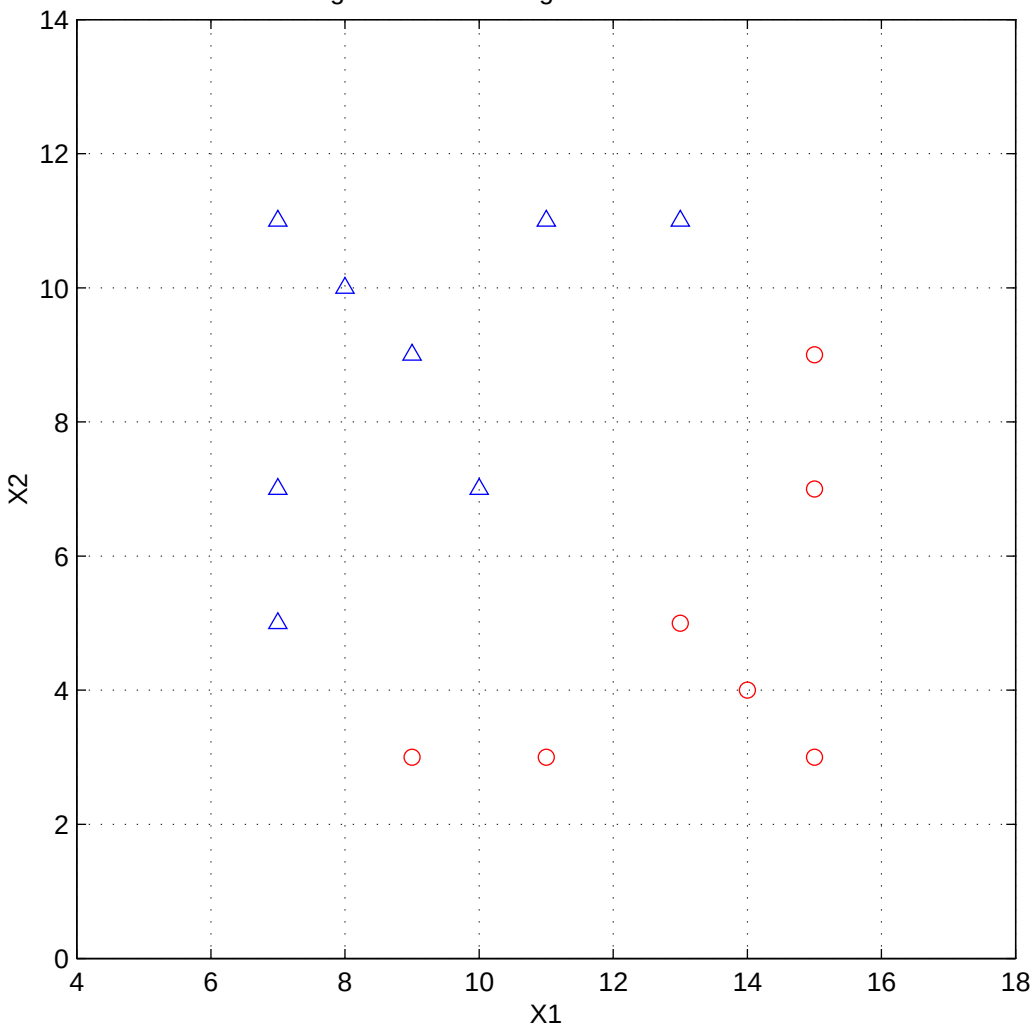
Evaluate the decision boundary for linear SVM on this data.

- Write a clear function form out for the SVM decision boundary.

- Draw on the graph both the boundary line and the two marginal lines. Make sure to label the regions with the classification lable that would be given. (The image file is also provided separately for your convenience)

- How many support vectors in this case ? Mark them clearly on the graph

Figure2. The training data for linear SVM



4.2 Nonparametric representation in SVM

Let us do a little thinking now. When all the training of SVM is done, we end up with a simple form for the SVM model (recall that we assume 2 classes, representing them as 1 and -1): The class prediction for a testing point x is the sign of

$$\sum_{k=1}^{R} \alpha_k * y_k * (\vec{x} \cdot \vec{x_k}) - b \tag{6}$$

So how many dot products must be done for M test points? In the case of [4.1], how many dot products needed to classify M test points?

4.3 Kernel trick in SVM.

It turns out that the dot product is sometimes used as a type of distance. We can replace the standard dot product 'distance' with some non-linear distance $K(\vec{x_1}, \vec{x_2})$, called 'kernel'.
So then our class prediction for a testing point x is the sign of

$$\sum_{k=1}^{R} \alpha_k * y_k * (K(\vec{x}, \vec{x_k})) - b \tag{7}$$

One example kernel we could try is

$$K(\vec{x_1}, \vec{x_2}) = e^{-\frac{\|\vec{x_1} - \vec{x_2}\|^2}{2\sigma^2}} \tag{8}$$

For this radial-basis-style kernel, how to choose the number of centers ? (We know that a RBF model's centers are on each of the training data.)

4.4 Maximum margin and quadratic programming in SVM

The idea of maximum margin, instead of maximum likelihood, is the truly new contribution of SVM's. The basic point is that if what we care about is discriminating classes, we should optimize directly for that by finding good separating planes rather than spend the efforts modeling each class's distribution accurately. While the idea of maximum margin for separating classes is simple, the resulting optimization problem turns out to be a quadratic programming problem.

- Why is SVM training a quadratic programming (QP) rather than a linear programming (LP) problem.
- For the separable data case, how many variables are in the QP problem? How many constraints? In terms of our simple data in [4.1], what will be the numbers ?
- For the noisy (not separable) data case: How many variables are in the QP problem? How many constraints?

4.5 Now let us explore the effect of basis functions on the maximum margin solution. Consider a simple one dimensional case, where we have only two training examples:

$$(x_1 = 0, y_1 = -1), (x_2 = \sqrt{2}, y_2 = 1) \tag{9}$$

Now we use a second order polynomial kernel, that means, mapping each input data x to a

$$\Phi(x) = [1, \sqrt{2}x, x^2]^T \tag{10}$$

Now we would like to find and understand the maximum margin solution: $\hat{W_1} = [\hat{\omega_1}, \hat{\omega_2}, \hat{\omega_3}]$ and $\hat{\omega_0}$ to:

$$min\|W_1\|^2 \tag{11}$$

Subject to:

$$y_1 * [\omega_0 + \Phi(x_1)^T W_1] \geqslant 1 \tag{12}$$

$$y_2 * [\omega_0 + \Phi(x_2)^T W_1] \geqslant 1 \tag{13}$$

Please report your derivations along with the specific answers.

- Using your knowledge of the maximum margin boundary, write down a vector that points in the same direction as $\hat{W_1}$

- What is the value of the margin that we can achieve in this case?

- By relating the margin and $\hat{W_1}$, provide the solution of $\hat{W_1}$.

- When we know $\hat{W_1}$, what is value of $\hat{\omega_0}$ then?

10-701 Machine Learning, Spring 2011: Homework 4

Due: Tuesday March 1st at the begining of the class

Instructions There are two questions on this assignment. Please submit your writeup as two separate sets of pages according to questions, with your name and userid on each set.

1 EM and Bayes Nets [Yi Zhang, 40 points]

In the class we learned the famous expectation-maximization algorithm, which is widely used to estimate model parameters from partially observed data. In this question, we will use **[EM]** to denote the lecture slides on Feb 17th ("Graphical models 3: EM") from the course website. Since each page of [EM] contains two slides, we will use, e.g., the upper/lower slide on page 7 of [EM], as a pointer to the first/second slide on a specific page of lecture slides.

Given the set of observed variables $\mathbf{X}$ and the set of unobserved variables $\mathbf{Z}$, as shown by the lower slide on page 6 of [EM], the EM algorithm for estimating model parameters θ from training examples is the following procedure:

1. E-Step: for each example k, use $\mathbf{X}_k$ and the current θ to calculate $P(\mathbf{Z}_k|\mathbf{X}_k, \theta)$.
2. M-Step: re-estimate θ as $\theta \leftarrow \text{argmax}_{\theta'} E_{P(\mathbf{Z}|\mathbf{X},\theta)}[\log P(\mathbf{X}, \mathbf{Z}|\theta')]$ using the training examples.
3. Iterate until convergence.

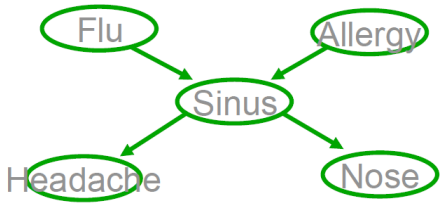


Figure 1: The Bayes net for the flu-allergy example

Now consider the familiar flu-allergy Bayes net as figure 1. Suppose F, A, H, N are observed (i.e., $\mathbf{X} = \{F, A, H, N\}$) and S is unobserved (i.e., $\mathbf{Z} = \{S\}$). Given a set of training examples $\{(f_k, a_k, h_k, n_k)\}_{k=1}^K$, the EM algorithm should proceed as follows:

1. E-Step: for each example k, use (f_k, a_k, h_k, n_k) and the current θ to calculate $P(S_k|f_k, a_k, h_k, n_k, \theta)$.
2. M-Step: re-estimate θ as $\theta \leftarrow \text{argmax}_{\theta'} \sum_{k=1}^K E_{P(S_k|f_k,a_k,h_k,n_k,\theta)}[\log P(f_k, a_k, h_k, n_k, S_k|\theta')]$.
3. Iterate until convergence.

1.1 A simplified EM algorithm

As we saw in the lower slide on page 8 of [EM], the EM algorithm can be simplified when the unobserved variable is of boolean values. In this case, the *simplified* EM algorithm is:

1. E-Step: for each example k, use $\mathbf{X}_k$ and the current θ to calculate the expected value $E(\mathbf{Z}_k|\mathbf{X}_k, \theta)$.
2. M-Step: re-estimate θ similarly to MLE on observed data, but replacing each count involving the unobserved variable by its expected count.
3. Iterate until convergence.

As a result, as shown by the upper slide on page 8 of [EM], the *simplified* EM algorithm for the flu-allergy Bayes net with a set of training examples $\{(f_k, a_k, h_k, n_k)\}_{k=1}^K$ is:

1. E-Step: for each example k, use (f_k, a_k, h_k, n_k) and the current θ to calculate the expected value $E(S_k) = E(S_k|f_k, a_k, h_k, n_k, \theta)$.
2. M-Step: re-estimate θ as MLE with expected counts, e.g., $\theta_{s|ij} \leftarrow \frac{\sum_{k=1}^K \delta(f_k=i, a_k=j)E(S_k)}{\sum_{k=1}^K \delta(f_k=i, a_k=j)}$. Recall that $\theta_{s|ij}$ stands for $P(S = 1|F = i, A = j)$.
3. Iterate until convergence.

Question [15 pts]: Given that S in the flu-allergy Bayes net is a boolean variable, prove that the simplified EM algorithm for estimating $\theta_{s|ij}$ is indeed equivalent to the standard EM algorithm. In other words, show that $\theta_{s|ij} \leftarrow \frac{\sum_{k=1}^K \delta(f_k=i, a_k=j)E(S_k)}{\sum_{k=1}^K \delta(f_k=i, a_k=j)}$ in the M-Step of the simplified EM indeed gives the corresponding parameter in $\theta \leftarrow \text{argmax}_{\theta'} \sum_{k=1}^K E_{P(S_k|f_k,a_k,h_k,n_k,\theta)}[\log P(f_k, a_k, h_k, n_k, S_k|\theta')]$ from the M-Step of the standard EM algorithm.

Note: you are required to prove this only for $\theta_{s|ij}$, and no need to worry about other parameters in θ.

Hint: the lower slide on page 4 of [EM] should be helpful. The slide shows how to derive the parameter $\theta_{s|ij}$ in $\theta \leftarrow \text{argmax}_{\theta'} \sum_{k=1}^K [\log P(f_k, a_k, h_k, n_k, s_k|\theta')]$ when all variables are observed.

★ **SOLUTION:** To find $\text{argmax}_{\theta'_{s|ij}} \sum_{k=1}^K E_{P(S_k|f_k,a_k,h_k,n_k,\theta)}[\log P(f_k, a_k, h_k, n_k, S_k|\theta')]$ in the M-Step, we need to take the derivative w.r.t. $\theta'_{s|ij}$ and set it to zero. Let's write $P(S_k|\theta) = P(S_k|f_k, a_k, h_k, n_k, \theta)$ for short. Note that $P(S_k|\theta)$ does not depend on θ' (or $\theta'_{s|ij}$).

$$
\begin{aligned}
& \frac{\partial \sum_{k=1}^K E_{P(S_k|\theta)}[\log P(f_k, a_k, h_k, n_k, S_k|\theta')]}{\partial \theta'_{s|ij}} \\
= & \frac{\partial \sum_{k=1}^K E_{P(S_k|\theta)}[\log P(S_k|f_k, a_k, \theta')]}{\partial \theta'_{s|ij}} \qquad \text{(only } P(S_k|f_k, a_k) \text{ matters)} \\
= & \frac{\partial \sum_{k=1}^K [P(S_k = 1|\theta) \log P(S_k = 1|f_k, a_k, \theta') + P(S_k = 0|\theta) \log P(S_k = 0|f_k, a_k, \theta')]}{\partial \theta'_{s|ij}} \quad \text{(expectation over } P(S_k|\theta)) \\
= & \frac{\partial \sum_{k=1}^K [E(S_k) \log P(S_k = 1|f_k, a_k, \theta') + (1 - E(S_k)) \log P(S_k = 0|f_k, a_k, \theta')]}{\partial \theta'_{s|ij}} \quad \text{(definition of } E(S_k)) \\
= & \frac{\partial \sum_{k=1}^K \delta(f_k = i, a_k = j)[E(S_k) \log P(S_k = 1|f_k, a_k, \theta') + (1 - E(S_k)) \log P(S_k = 0|f_k, a_k, \theta')]}{\partial \theta'_{s|ij}} \quad \text{(only } f_k = i, a_k = j) \\
= & \sum_{k=1}^K \delta(f_k = i, a_k = j)[E(S_k) \frac{\partial \log P(S_k = 1|f_k, a_k, \theta')}{\partial \theta'_{s|ij}} + (1 - E(S_k)) \frac{\partial \log P(S_k = 0|f_k, a_k, \theta')}{\partial \theta'_{s|ij}}] \\
= & \sum_{k=1}^K \delta(f_k = i, a_k = j)[E(S_k) \frac{\partial \log \theta'_{s|ij}}{\partial \theta'_{s|ij}} + (1 - E(S_k)) \frac{\partial \log(1 - \theta'_{s|ij})}{\partial \theta'_{s|ij}}] \\
= & \sum_{k=1}^K \delta(f_k = i, a_k = j)[E(S_k) \frac{1}{\theta'_{s|ij}} - (1 - E(S_k)) \frac{1}{(1 - \theta'_{s|ij})}]
\end{aligned}
$$

Set the above formula to zero, we have $\theta'_{s|ij} = \frac{\sum_{k=1}^K \delta(f_k=i, a_k=j)E(S_k)}{\sum_{k=1}^K \delta(f_k=i, a_k=j)}$.

1.2 Simulating the simplified EM algorithm

We are given the following $K = 8$ training examples as in Table 1, where only two examples contain unobserved values, namely, H_7 and N_8. We like to simulate a few steps of the simplified EM algorithm by hand. Note that this is not a programming question.

Table 1: Training examples for the flu-allergy Bayes net

k	F	A	S	H	N
$k = 1$	1	0	1	1	1
$k = 2$	0	1	1	1	0
$k = 3$	1	1	1	1	1
$k = 4$	0	0	0	0	0
$k = 5$	0	0	0	1	0
$k = 6$	0	0	0	0	1
$k = 7$	1	1	1	**?**	1
$k = 8$	1	1	1	1	**?**

Question A [7 pts]: given that all variables are boolean, how many parameters we need to estimate in the flu-allergy Bayes net? Also, list all the parameters we need to estimate, e.g., one parameter will be $\theta_{s|11}$, which stands for $P(S = 1|F = 1, A = 1)$.

★ **SOLUTION:** We have 10 parameters:

$$\begin{aligned}
\theta_f &= P(F = 1) \\
\theta_a &= P(A = 1) \\
\theta_{s|00} &= P(S = 1|F = 0, A = 0) \\
\theta_{s|01} &= P(S = 1|F = 0, A = 1) \\
\theta_{s|10} &= P(S = 1|F = 1, A = 0) \\
\theta_{s|11} &= P(S = 1|F = 1, A = 1) \\
\theta_{h|0} &= P(H = 1|S = 0) \\
\theta_{n|1} &= P(H = 1|S = 1) \\
\theta_{n|0} &= P(N = 1|S = 0) \\
\theta_{n|1} &= P(N = 1|S = 1)
\end{aligned}$$

Question B [6 pts]: Now we like to simulate the first E-step of the simplified EM algorithm. Before we start, we initialize all the parameters as 0.5, and then proceed to execute the E-step. What are the expectations we need to calculate in this E-step? Also, list the actual values for these expectations. (Note that only two examples contains unobserved variables).

★ **SOLUTION:** $E(H_7|f_7, a_7, s_7, n_7, \theta) = E(H_7|s_7 = 1, \theta_{h|1}) = 0.5$, and $E(N_8|f_8, a_8, s_8, h_8, \theta) = E(N_8|s_8 = 1, \theta_{n|1}) = 0.5$.

Question C [6 pts]: Now we like to simulate the first M-step. List the estimated values of all model parameters we obtain in this M-step. (Note that we use the expected count only when the variable is unobserved in an example).

★ **SOLUTION:**

$$\begin{aligned}
\theta_f &= P(F=1) = 0.5 \\
\theta_a &= P(A=1) = 0.5 \\
\theta_{s|00} &= P(S=1|F=0, A=0) = 0 \\
\theta_{s|01} &= P(S=1|F=0, A=1) = 1 \\
\theta_{s|10} &= P(S=1|F=1, A=0) = 1 \\
\theta_{s|11} &= P(S=1|F=1, A=1) = 1 \\
\theta_{h|0} &= P(H=1|S=0) = 1/3 \\
\theta_{n|1} &= P(H=1|S=1) = 4.5/5 = 0.9 \\
\theta_{n|0} &= P(N=1|S=0) = 1/3 \\
\theta_{n|1} &= P(N=1|S=1) = 3.5/5 = 0.7
\end{aligned}$$

Question D [6 pts]: Last, let's simulate the second E-step. List the actual values for all the expectations we calculate in this E-step.

★ **SOLUTION:** $E(H_7|f_7, a_7, s_7, n_7, \theta) = E(H_7|s_7 = 1, \theta_{h|1}) = 0.9$, and $E(N_8|f_8, a_8, s_8, h_8, \theta) = E(N_8|s_8 = 1, \theta_{n|1}) = 0.7$.

2 Midterm review questions [Tom Mitchell, 60 points]

This question contains some short questions adapted from previous midterm exams – a good way to review for our own on March 3.

1. Give a *one sentence* reason why [12 pts]:

   - we might prefer Decision Tree learning over Logistic Regression for a particular learning task.
   - we might prefer Logistic Regression over Naive Bayes for a particular learning task.
   - we choose parameters that minimize the sum of squared training errors in Linear Regression.

   ★ **SOLUTION:**

   (a) We prefer Decision Tree over Logistic Regression when we expect the decision boundary to be nonlinear.
   (b) We prefer Logistic Regression over Naive Bayes when the conditional independence assumption does not hold for the data. (Note that Logistic Regression does not make the conditional independence assumption).
   (c) We choose parameters that minimize the sum of squared training errors because it corresponds to find the MLE (maximum likelihood estimate) of $P(Y|X)$ assuming that $P(Y|X)$ follows a Gaussian distribution.

2. Suppose we train several classifiers to learn $f : X \rightarrow Y$, where X is the feature vector $X =< X_1, X_2, X_3 >$. Which of the following classifiers contains sufficient information to allow calculating $P(X_1, X_2, X_3, Y)$? If you answer yes, give a brief sketch of how. If you answer no, state what is missing. [12 pts]

   - Gaussian Naive Bayes
   - Logistic Regression
   - Linear Regression

★ **SOLUTION:**

(a) YES. Naive Bayes estimates $P(X_1|Y)$, $P(X_2|Y)$, $P(X_3|Y)$ and $P(Y)$ and calculates $P(X_1, X_2, X_3, Y)$ by:

$$P(X_1, X_2, X_3, Y) = P(X_1|Y)P(X_2|Y)P(X_3|Y)P(Y)$$

(b) NO. Since Logistic Regression only estimates $P(Y|X_1, X_2, X_3)$, we cannot calculate $P(X_1, X_2, X_3, Y)$ without any information of $P(X_1, X_2, X_3)$.

(c) NO. The reason is similar to Logistic Regression. In Linear Regression, we assume that $P(Y|X_1, X_2, X_3) \sim N(w_0 + w_1X_1 + w_2X_2 + w_3X_3, \sigma^2)$ and Linear Regression estimates $\{w_0, w_1, w_2, w_3\}$ via MLE. But still, we do not have any information of $P(X_1, X_2, X_3)$ and hence cannot calculate $P(X_1, X_2, X_3, Y)$.

3. True or False? If true, give a 1-2 sentence explanation. If false, a counterexample. *Your answer must fit into the space below the question* [12 pts].

- As the number of data points grows to infinity, the MLE estimate of a parameter approaches the MAP estimate, for all possible priors.

- The depth of a learned decision tree can be larger than the number of training examples used to create the tree.

- There is *no* training data set for which a decision tree learner and logistic regression will output the same decision boundary.

★ **SOLUTION:**

(a) False. Suppose the prior probability is assigned to be 1 at a single point and zero at all other points (i.e. $P(\theta = \hat{\theta}) = 1)$). In this case, MLE and MAP are different as the number of data points approaches infinity.

(b) False. Assume that we have N data points. Since each tree node contains at least 1 data point (otherwise, we will not split its parent node), there will be at most $N - 1$ splits, each of which increases the depth at most by 1. Therefore, the depth of the tree is at most N. [Note that if the number of the features is more than the number of data points, the depth of the tree is still at most N. In this case, we will not split on all the features.]

(c) False. A simple counter example is as follows: the data set contains two data points: $\{X = -1, Y = -1\}$ and $\{X = 1, Y = 1\}$. The decision surfaces for decision tree and logistic regression are the same.

4. In class we defined *conditional independence* by saying that random variable X is conditionally independent of Y given Z if and only if:

$$P(X|Y, Z) = P(X|Z) \tag{1}$$

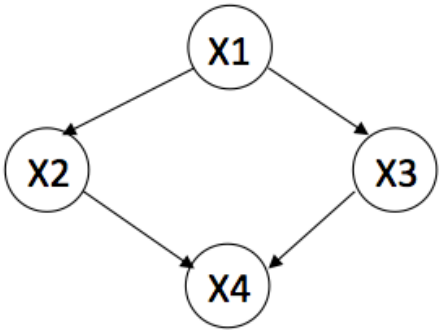
Prove that if $P(XY|Z) = P(X|Z)P(Y|Z)$, then X is conditionally independent of Y given Z (*hint: this is a two-line proof*) [4 pts].

★ **SOLUTION:** If $P(XY|Z) = P(X|Z)P(Y|Z)$, we have

$$P(X|Y,Z) = \frac{P(XY|Z)}{P(Y|Z)} = \frac{P(X|Z)P(Y|Z)}{P(Y|Z)} = P(X|Z).$$

Therefore, X is conditionally independent of Y given Z.

5. Consider the Bayes network below, defined over four Boolean variables [20 pts].



- How many parameters are needed to define $P(X1, X2, X3, X4)$ for this Bayes Net?
- Give the formula that calculates $P(X1 = 1, X2 = 0, X3 = 1, X4 = 0)$ using only the Bayes net parameters. Use notation like $P(X1 = 0|X2 = 1, X4 = 0)$ to refer to each Bayes net parameter you use in your formula.
- Give the formula that calculates $P(X1 = 1, X4 = 0)$ using only the Bayes net parameters.
- Give the formula that calculates $P(X2 = 1|X3 = 0)$ using only the Bayes net parameters.

★ **SOLUTION:**

(a) 9 parameters: $P(X1 = 1)$, $P(X2 = 1|X1 = 1)$, $P(X2 = 1|X1 = 0)$, $P(X3 = 1|X1 = 1)$, $P(X3 = 1|X1 = 0)$, $P(X4 = 1|X2 = 1, X3 = 1)$, $P(X4 = 1|X2 = 1, X3 = 0)$, $P(X4 = 1|X2 = 0, X3 = 1)$, $P(X4 = 1|X2 = 0, X3 = 0)$. [Note that the parameters $P(Xk = 0|\cdots)$ can be then computed $1 - P(Xk = 1|\cdots)$.]

(b)

$$\begin{aligned} & P(X1 = 1, X2 = 0, X3 = 1, X4 = 0) \\ = \; & P(X1 = 1)P(X2 = 0|X1 = 1)P(X3 = 1|X1 = 1)P(X4 = 0|X2 = 0, X = 1) \\ = \; & P(X1 = 1)\left(1 - P(X2 = 1|X1 = 1)\right)P(X3 = 1|X1 = 1)\left(1 - P(X4 = 1|X2 = 0, X = 1)\right) \end{aligned}$$

(c)

$$\begin{aligned} & P(X1 = 1, X4 = 0) \\ = \; & \sum_{x_2=0}^{1}\sum_{x_3=0}^{1} P(X1 = 1, X2 = x_2, X3 = x_3, X4 = 0) \\ = \; & P(X1 = 1)\sum_{x_2=0}^{1} P(X2 = x_2|X1 = 1)\sum_{x_3=0}^{1} P(X3 = x_3|X1 = 1)P(X4 = 0|X2 = x_2, X3 = x_3) \end{aligned}$$

(d)

$$
\begin{aligned}
& P(X2 = 1 | X3 = 0) \\
= \quad & \frac{P(X2 = 1, X3 = 0)}{P(X3 = 0)} \\
= \quad & \frac{\sum_{x_1=0}^{1} P(X1 = x_1) P(X2 = 1 | X1 = x_1) P(X3 = 0 | X1 = x_1)}{\sum_{x_1=0}^{1} P(X1 = x_1) P(X3 = 0 | X_1 = x_1)}
\end{aligned}
$$

10-701 Machine Learning, Spring 2011: Homework 5 Solution

April 25, 2011

Instructions There are three questions on this assignment. Please submit your writeup as two separate sets of pages according to questions, with your name and userid on each set.

1 Hidden Markov Models [Xi Chen, 30 points]

Andrew lives a simple life. Some days he is Angry and some days he is Happy. But he hides his emotional state, and so all you can observe is whether he smiles, frowns, laughs, or yells. We start on day 1 in the Happy state and there is one transition per day.

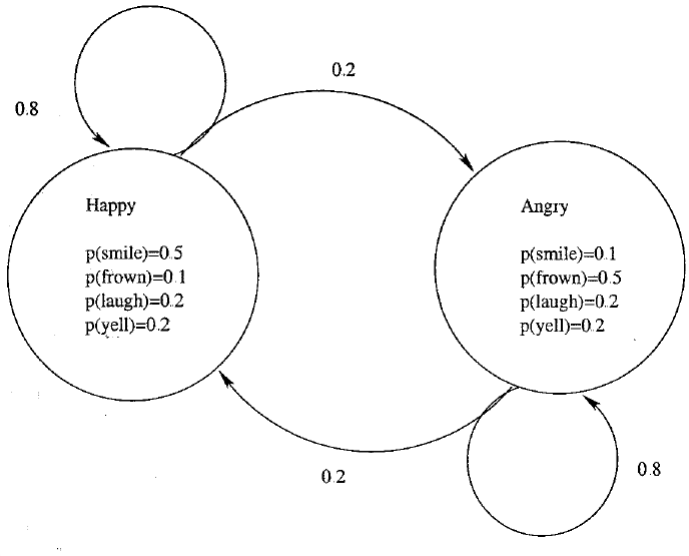


Figure 1: Transition Model for Andrew

We define;

- q_t: state on day t
- O_t: observation on day t

(a) What is $\Pr(q_2 = Happy)$? [5pt]

(b) What is $\Pr(O_2 = frown)$? [5pt]

(c) What is $\Pr(q_2 = Happy | O_2 = frown)$? [5pt]

(d) What is $\Pr(O_{100} = yell)$? [5pt]

(e) Assume that $O_1 = O_2 = O_3 = O_4 = O_5 = frown$. What is the most likely sequence of the states. [10pt]

★ **SOLUTION:**

(a)

$$\Pr(q_2 = Happy) = 0.8$$

(b)

$$\Pr(O_2 = frown) = \frac{8}{10}\frac{1}{10} + \frac{2}{10}\frac{1}{2} = \frac{18}{100} = 0.18$$

(c)

$$\Pr(q_2 = Happy|O_2 = frown) = \frac{\Pr(O_2 = frown|q_2 = Happy)\Pr(q_2 = Happy)}{\Pr(O_2 = frown)} = \frac{\frac{1}{10}\frac{8}{10}}{\frac{18}{100}} = \frac{4}{9}$$

(d)

$$\begin{aligned}\Pr(O_{100} = yell) &= \Pr(O_{100} = yell|q_{100} = Happy)\Pr(q_{100} = Happy) + \Pr(O_{100} = yell|q_{100} = Angry)\Pr(q_{100} = Angry \\ &= 0.2\end{aligned}$$

(e) Happy, Angry, Angry, Angry, Angry

2 Dimension Reduction [Yi Zhang, 35 points]

2.1 Principal components analysis vs. Fisher's linear discriminant

Principal components analysis (PCA) reduces the dimensionality of the data by finding projection direction(s) that *minimizes the squared errors in reconstructing the original data* or equivalently *maximizes the variance of the projected data*. On the other hand, Fisher's linear discriminant is a supervised dimension reduction method, which, given labels of the data, finds the projection direction that *maximizes the between-class variance relative to the within-class variance of the projected data*.

[10 points] In the following Figure 2, **draw** the first principal component direction in the left figure, and the first Fisher's linear discriminant direction in the right figure. Note: for PCA, ignore the fact that points are labeled (as round, diamond or square) since PCA does not use label information. For linear discriminant, consider round points as the positive class, and both diamond and square points as the negative class (since in the course lecture we only discuss the two-class case).

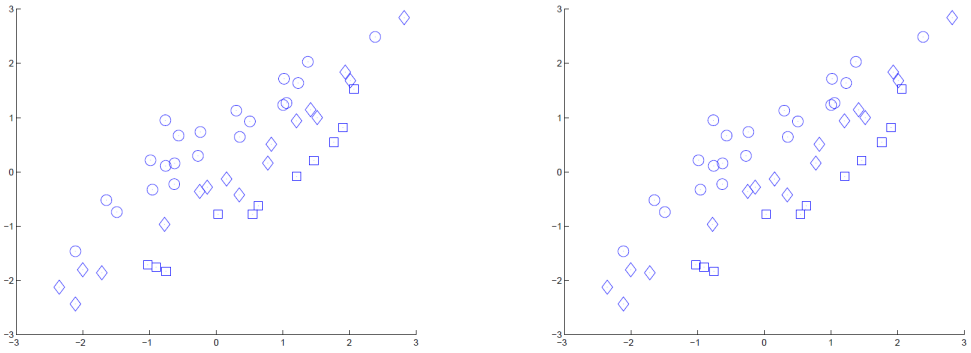


Figure 2: Draw the first principal component and linear discriminant component, respectively

★ **SOLUTION:** The PCA and LDA directions are shown in the following figure.

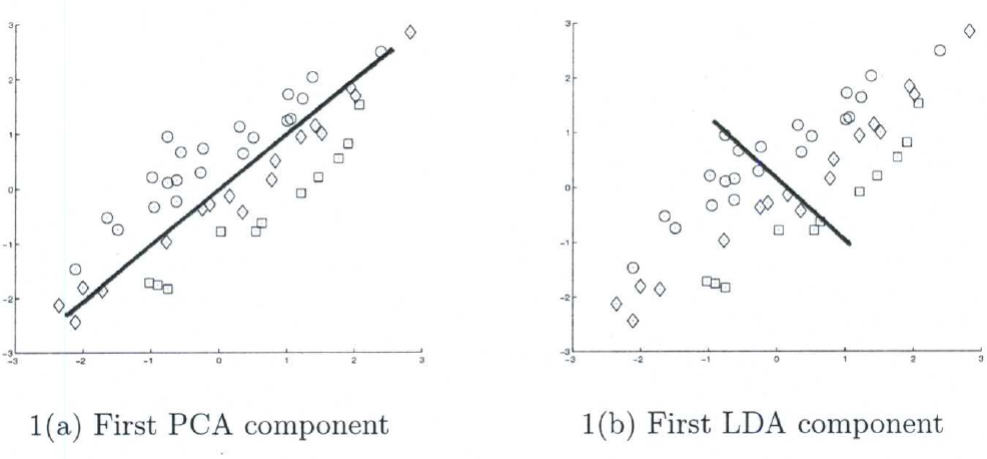


Figure 3: The first principal component and linear discriminant component, respectively

2.2 Canonical correlation analysis

Canonical correlation analysis (CCA) handles the situation that each data point (i.e., each object) has two representations (i.e., two sets of features), e.g., a web page can be represented by the text on that page, and can also be represented by other pages linked to that page. Now suppose each data point has two representations $\mathbf{x}$ and $\mathbf{y}$, each of which is a 2-dimensional feature vector (i.e., $\mathbf{x} = [x_1, x_2]^T$ and $\mathbf{y} = [y_1, y_2]^T$). Given a set of data points, CCA finds a pair of projection directions $(\mathbf{u}, \mathbf{v})$ to maximize the sample correlation $\hat{corr}(\mathbf{u}^T\mathbf{x})(\mathbf{v}^T\mathbf{y})$ along the directions $\mathbf{u}$ and $\mathbf{v}$. In other words, after we project one representation of data points onto $\mathbf{u}$ and the other representation of data points onto $\mathbf{v}$, the two *projected* representations $\mathbf{u}^T\mathbf{x}$ and $\mathbf{v}^T\mathbf{y}$ should be maximally correlated (intuitively, data points with large values in one projected direction should also have large values in the other projected direction).

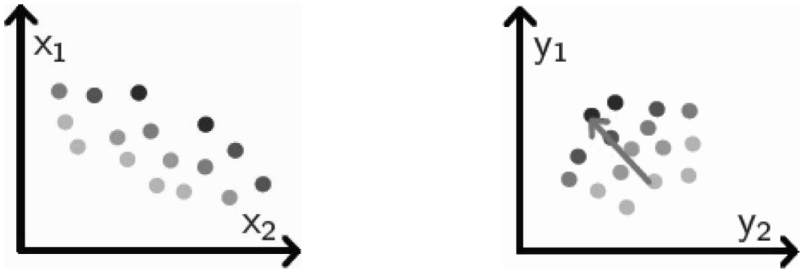


Figure 4: Draw the CCA projection direction in the left figure

[6 points] Now we can see data points shown in the Figure 4, where each data point has two representations $\mathbf{x} = [x_1, x_2]^T$ and $\mathbf{y} = [y_1, y_2]^T$. Note that data are paired: each point in the left figure corresponds to a specific point in the right figure and vice versa, because these two points are two representations of the same object. Different objects are shown in different gray scales in the two figures (so you should be able to approximately figure out how points are paired). In the right figure we've given one CCA projection direction $\mathbf{v}$, **draw** the other CCA projection direction $\mathbf{u}$ in the left figure.

★ **SOLUTION:** The CCA projection direction is shown in the following figure (on the left).

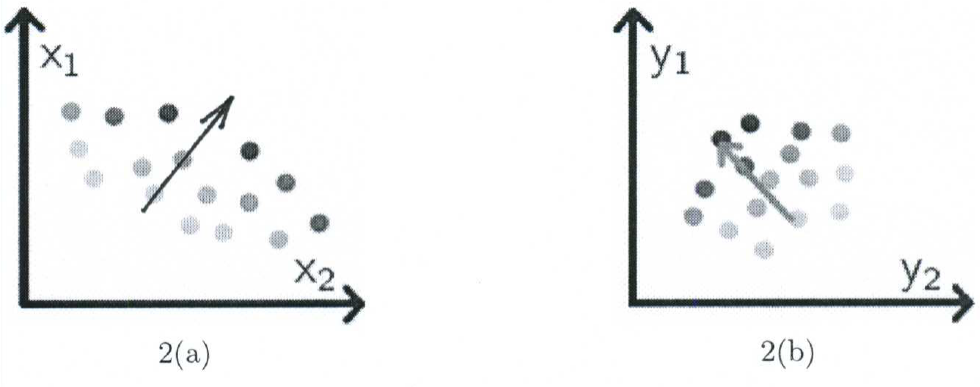


Figure 5: The first principal component and linear discriminant component, respectively

2.3 More Principal Components Analysis

Consider 3 data points in the 2-d space: (-1, -1), (0,0), (1,1).

[6 points] What is the first principal component (write down the actual vector)?

★ **SOLUTION:** The first principal component is $\mathbf{v} = [\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]^T$ (you shouldn't really need to solve any SVD or eigenproblem to see this). Note that the principal component should be normalized to have unit length. (The negation $\mathbf{v} = [-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}]^T$ is also correct.)

[7 points] If we project the original data points into the 1-d subspace by the principal component you choose, what are their coordinates in the 1-d subspace? And what is the variance of the projected data?

★ **SOLUTION:** The coordinates of three points after projection should be $z_1 = \mathbf{x}_1^T\mathbf{v} = [-1, -1][\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]^T = -\sqrt{2},\ z_2 = \mathbf{x}_2^T\mathbf{v} = 0,\ z_3 = \mathbf{x}_3^T\mathbf{v} = \sqrt{2}$. Note that the sample mean is 0, and thus the variance is $\frac{1}{3}\sum_{i=1}^{3}(z_i - 0)^2 = \frac{4}{3}$ (or you can also choose to use the unbiased estimation $\frac{1}{3-1}\sum_{i=1}^{3}(z_i - 0)^2 = 2$).

[6 points] For the projected data you just obtained above, now if we represent them in the original 2-d space and consider them as the reconstruction of the original data points, what is the reconstruction error?

★ **SOLUTION:** The reconstruction error is 0, since all three points are perfectly located on the direction of the first principal component. Or, you can actually calculate the reconstruction: $\hat{\mathbf{x}}_1 = z_1 \cdot \mathbf{v} = -\sqrt{2} \cdot [\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]^T = [-1, -1]^T,\ \hat{\mathbf{x}}_2 = [0, 0]^T,\ \hat{\mathbf{x}}_3 = [1, 1]^T$, which are exactly $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$.

3 Neural Nets [Carl, 35 points]

The neural networks shown in class used logistic units: that is, for a given unit U, if A is the vector of activations of units that send their output to U, and W is the weight vector corresponding to these outputs, then the activation of U will be $(1 + exp(W^T A))^{-1}$. However, activation functions could be anything. In this exercise we will explore some others. Consider the following neural network, consisting of two input units, a single hidden layer containing two units, and one output unit:

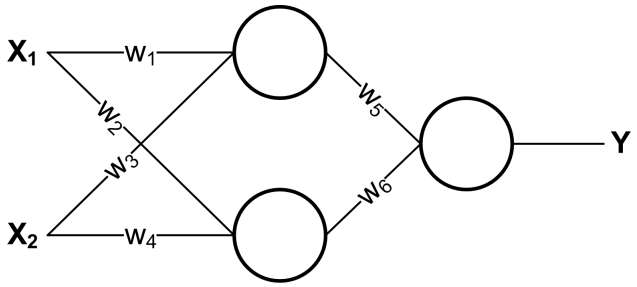


Figure 6: Neural network for question 3

1. **[9 points]** Say that the network is using linear units: that is, defining W and and A as above, the output of a unit is $C * W^T A$ for some fixed constant C. Let the weight values w_i be fixed. Re-design the neural network to compute the same function without using any hidden units. Express the new weights in terms of the old weights and the constant C.

   ★ **SOLUTION:** Connect the input for X_1 to the output unit with a weight $C * (w_5 * w_1 + w_6 * w_2)$, and connect the input for X_2 to the output unit with weight $C(w_5 * w_3 + w_6 * w_4)$. Then the output unit can use the same activation function it used originally.

2. **[4 points]** Is it always possible to express a neural network made up of only linear units without a hidden layer? Give a one-sentence justification.

   ★ **SOLUTION:** This is true. Each layer can be thought of as performing a matrix multiply to find its representation given the representation on the layer that it receives input from. Thus the entire netowrk just performs a chain of matrix multiplies, and therefore we can simply multiply the matrices together to find the weights to represent the function with a single layer.

3. **[9 points]** Another common activation function is a theshold, where the activation is $t(W^T A)$ where $t(x)$ is 1 if $x > 0$ and 0 otherwise. Let the hidden units use sigmoid activation functions and let the output unit use a threshold activation function. Find weights which cause this network to compute the XOR of X_1 and X_2 for binary-valued X_1 and X_2. Keep in mind that there is no bias term for these units.

   ★ **SOLUTION:** One solution: $w_1 = w_3 = -10$,$w_2 = w_4 = -1$,$w_5 = 5$, and $w_6 = -6$. The intuition here is that we can decompose `A XOR B` into `(A OR B) AND NOT (A AND B)`. We make the upper hidden unit behave like an OR by making it saturate when either of the input units are 1. It isn't possible to make a hidden unit that behaves exactly like AND, but we can at least make the lower hidden unit continue to increase in activation after the upper one has saturated.

4. **[4 points]** Why are threshold activation functions generally inconvenient for training? Explain in one sentence.

★ **SOLUTION:** Although many people wrote that the non-differentiability of the threshold function makes computing gradients impossible, this isn't exactly correct. In fact, it is quite common to perform gradient descent on the absolute value function for the sake of finding sparse representations (google the lasso if you're interested). A bit harder to deal with is the fact that the threshold function has a discontinuity; there are optimization techniques to handle some forms of discontinuity, but I don't know of one that would work well with a threshold. Probably the worst problem is that the gradient is zero almost everywhere; therefore, all of the weight updates computed by backpropagation would be zero.

5. [**9 points**] Using the same architecture as in figure 6, choose an activation function for each unit in the network which will cause this network to learn the same function that logistic regression would learn. Each unit must use a logistic, linear, or threshold activation function, with no constraints on the weights. You may assume either gradient-descent learning, or you may assume that there is an oracle which can set the weights optimally in terms of squared-error.

★ **SOLUTION:** Most of the class correctly assumed what was implicitly assumed in the problem: that the goal was to create a network that simulated logistic regression with no bias term. A handful of people tried to create a network with a bias, for example, using a threshold unit, but it isn't possible to do this without allowing the network to represent nonlinear decision surfaces. However, I tried not to take off points for this if your argument was well-reasoned.

The intended solution was to create a hidden layer with linear units and a logistic output unit. As long as the weight matrix between the first and second layers is non-singular (i.e. it is possible to reconstruct the input layer from the hidden layers), it will always be possible for the final layer to learn a pair of weights which will achieve the optimal squared-error, since the entire optimization is linear up until the final sigmoid function is applied on the last hidden layer.

Due to a technicality, though, this isn't quite correct. The objective functions for logistic regression and neural networks are different: neural networks optimize squared error, whereas logistic regression optimizes probabilities. In general, the optima of these two objective functions will not be exactly the same, although they will usually be quite similar.

Despite its flaws, nearly the entire class understood the spirit of this question. Thus I still graded it, though I tried to be forgiving of misinterpretations, and therefore rarely took off points.

10-701 Machine Learning, Spring 2011: Homework 6

Instructions This homework is completely optional, and will NOT be collected or graded. We provide it to help you review the final exam. Feel free to work on this together with other students as you study for the exam.

1 Kernel and SVM

1. **Kernel**: Kernel functions implicitly define some mapping function $\phi(\cdot)$ that transforms an input instance $\mathbf{x} \in \mathbb{R}^d$ to high dimensional space Q by giving the form of dot product in Q: $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \langle \phi(\mathbf{x}_i) \ , \ \phi(\mathbf{x}_j) \rangle$.

   (a) Prove that the kernel is symmetric, i.e. $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_j, \mathbf{x}_i)$.

   ★ **SOLUTION:** We have $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle = K(\mathbf{x}_j, \mathbf{x}_i)$

   (b) Assume we use radial basis kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$. Thus there is some implicit unknown mapping function $\phi(\mathbf{x})$. Prove that for any two input instances $\mathbf{x}_i$ and $\mathbf{x}_j$, the squared Euclidean distance of their corresponding points in the feature space Q is less than 2, i.e. prove that $\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 \leq 2$.

   ★ **SOLUTION:**

$$\begin{aligned} & \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 \\ = \ & \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_i) \rangle + \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_j) \rangle - 2 \cdot \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \\ = \ & K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2 \cdot K(\mathbf{x}_i, \mathbf{x}_j) \\ = \ & 1 + 1 - 2\exp(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2) \\ < \ & 2 \end{aligned}$$

2. **SVM**: With the help of a kernel function, SVM attempts to construct a hyper-plane in the feature space Q that maximizes the margin between two classes. The classification decision of any $\mathbf{x}$ is made on the basis of the sign of

$$\langle \hat{\mathbf{w}} \ , \ \phi(\mathbf{x}) \rangle + \hat{w}_0 = \sum_{i \in SV} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + \hat{w}_0 = f(\mathbf{x}; \alpha, \hat{w}_0),$$

   where $\hat{\mathbf{w}}$ and $\hat{w}_0$ are parameters for the classification hyper-plane in the feature space Q, SV is the set of support vectors, and α_i is the coefficient for the i-th support vector. Again we use the radial basis kernel function. Assume that the training instances are linearly separable in the feature space Q, and assume that the SVM finds a margin that perfectly separates the points.

   If we choose a test point $\mathbf{x}_{far}$ which is far away from any training instance $\mathbf{x}_i$ (distance here is measured in the original space $\mathbb{R}^d$), prove that $f(\mathbf{x}_{far}; \alpha, \hat{w}_0) \approx \hat{w}_0$.

★ **SOLUTION:**

$$
\begin{aligned}
& \|\mathbf{x}_{far} - \mathbf{x}_i\| \gg 0 \quad \forall i \in SV \\
\Rightarrow \quad & K(\mathbf{x}_{far}, \mathbf{x}_i) \approx 0 \quad \forall i \in SV \\
\Rightarrow \quad & \sum_{i \in SV} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) \approx 0 \\
\Rightarrow \quad & f(\mathbf{x}; \alpha, \hat{w}_0) \approx \hat{w}_0
\end{aligned}
$$

2 Active Learning

This simple question tests our understanding on uncertainty sampling and version space reduction as two popular active learning strategies. Recall in the class that Burr used the example of learning the threshold to classify between poisonous and safe fruits, as shown in the following figure.

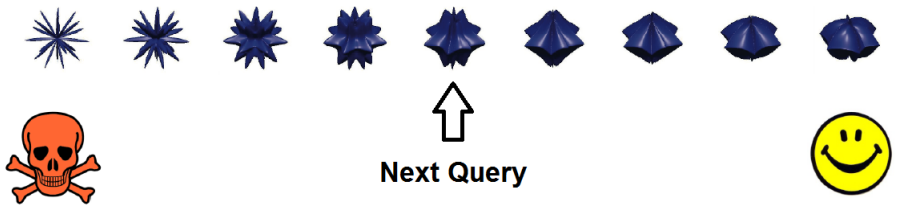


Figure 1: Learning the threshold between poisonous and safe fruits, with two labeled examples.

In the figure we already have two labeled examples: the leftmost one is poisonous ($y = +$) and the rightmost one is safe ($y = -$). Now we want to decide which fruit (among the pool of seven unlabeled fruits in the figure) to query next. According to binary search, we should query the label of the fruit in the middle.

2.1 Uncertainty sampling

Justify this choice from the perspective of uncertainty sampling. To do this, we need: 1) choose a class of hypothesis (i.e., a type of classifiers) we want to learn; 2) point out what is the classifier we will learn given the two labeled examples; 3) select a proper measure of uncertainty (three taught in the class); 4) given the learned classifier and the uncertainty measure, argue why we should choose the fruit in the middle.

★ **SOLUTION:** Let's choose logistic regression as the classifier type (since we need a probabilistic classifier to measure the uncertainty on unlabeled examples). Given the two labeled examples, we will learn a logistic regression whose linear decision boundary is right in the middle of the span. Now we choose maximum entropy as a measure of uncertainty. For the fruit x in the middle, it's clear that the learned logistic regression will predict $p(y = +|x) = p(y = -|x) = 0.5$ (since it is located exactly on the decision boundary). Since the entropy of $p(y|x)$ is maximized when $p(y = +|x) = p(y = -|x) = 0.5$, we will select the fruit in the middle as the next sample to query.

2.2 Version space reduction

Justify the choice of querying the fruit in the middle from the perspective of version space reduction.

Hint 1. We can assume the entire hypothesis space contains only a finite number of classifiers: since there are 9 fruits in total, there are 10 possible classifiers (each classifier is a threshold such that all fruits on the left side are poisonous and all fruits on the right side are safe).

Hint 2. When we choose a fruit to query, its label is unknown. However, the reduction of the version space depends on the actual label of the queried fruit. As a result, we can choose an unlabeled fruit to maximize the "expected" reduction of the version space (expectation taken over the distribution of this fruit's label), to maximize the "best-case" reduction of the version space (i.e., when the fruit's label is the one that maximizes the version space reduction), or to maximize the "worst-case" reduction of the version space (i.e., when the fruit's label is the one that minimizes the version space reduction). We need to choose this notion properly to justify the choice of querying the fruit in the middle.

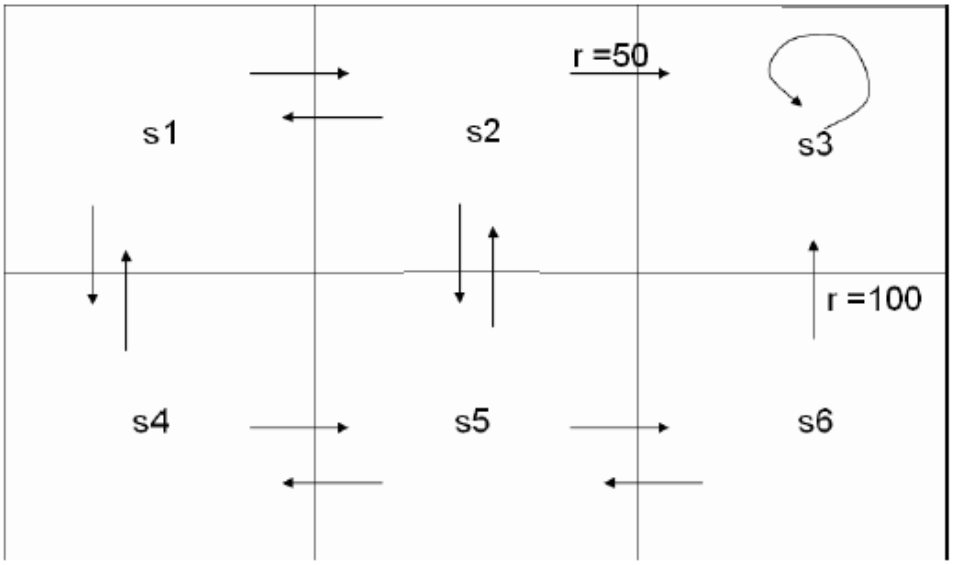
★ **SOLUTION:** We will consider maximizing the "worst-case" reduction of the version space. The hypothesis space contains 10 classifiers, and the current version space contains 8 classifiers that are consistent with the two labeled examples. If we query the unlabeled fruit in the middle, we are guaranteed to reduce the version space by

4, no matter whether the queried fruit is poisonous or safe. On the other hand, by querying any other unlabeled fruit, the "worst-case" reduction of the version space is smaller than 4 (e.g., if we query the second fruit from the left and it turns out to be poisonous, we will only reduce the version space by 1).

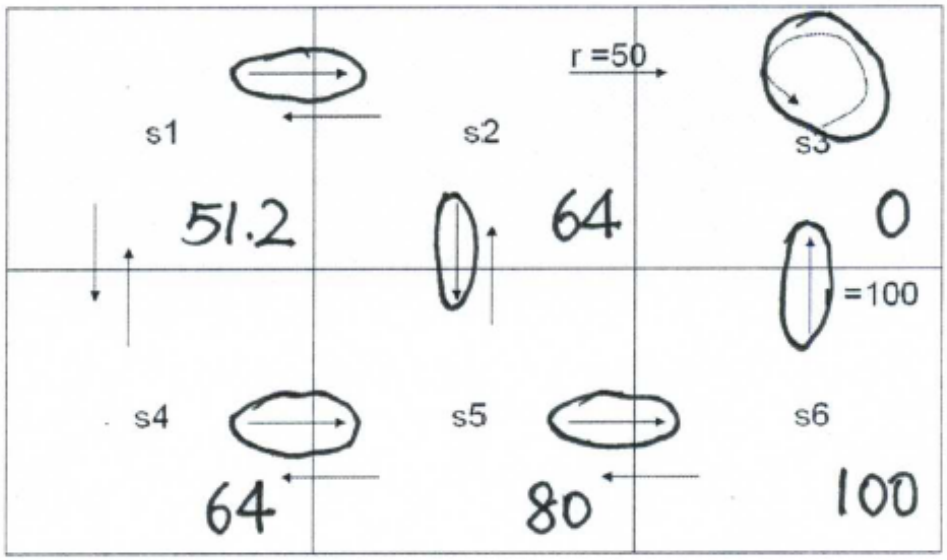
3 MDPs and Reinforcement Learning

Part A.:

Consider the following deterministic Markov Decision Process (MDP), describing a simple robot grid world. Notice the values of the immediate rewards are written next to transitions. Transitions with no value have an immediate reward of 0. **Assume the discount factor** $\gamma = 0.8$.



1. For each state s, write the value for $V^*(s)$ inside the corresponding square in the diagram.
2. Mark the state-action transition arrows that correspond to one optimal policy. If there is a tie, always choose the state with the smallest index.



3. Give a different value forγ which results in a different optimal policy and the number of changed policy actions should be minimal. Give your new value for γ and describe the resulting policy by indicating which $\pi(s)$ values (i.e., which policy actions) change.

   New value for γ:

   Changed policy actions:

   ★ **SOLUTION:** New value for γ: 0.7

   Changed policy actions: $\pi(S_2) = S_3$

   For the remainder of this question, assume again that $\gamma = 0.8$.

4. How many complete loops (iterations) of value iteration are sucient to guarantee finding the optimal policy for this MDP? Assume that values are initialized to zero, and that states are considered in an arbitrary order on each iteration.

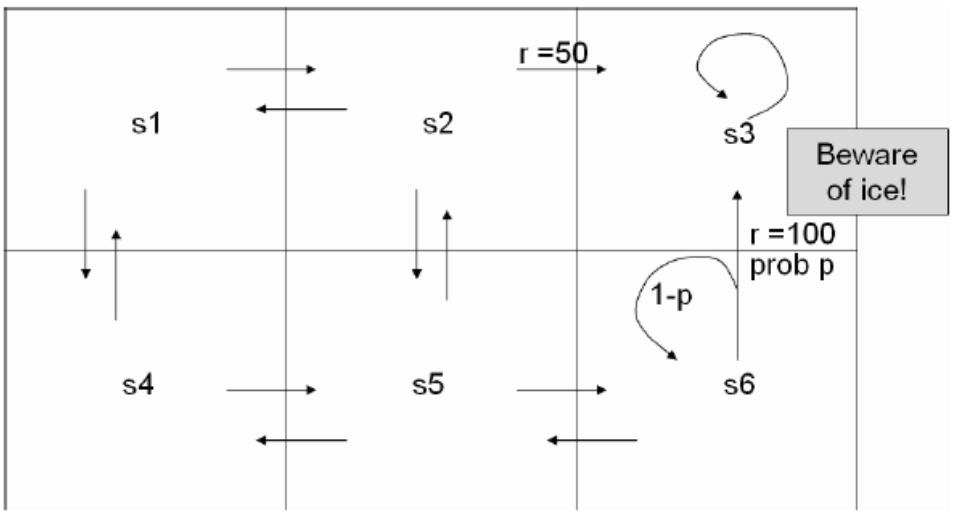
   ★ **SOLUTION:** 4 iterations

5. Is it possible to change the immediate reward function so that V^* changes but the optimal policy π^* remains unchanged? If yes, give such a change, and describe the resulting change to V^*. Otherwise, explain in at most 2 sentences why this is impossible.

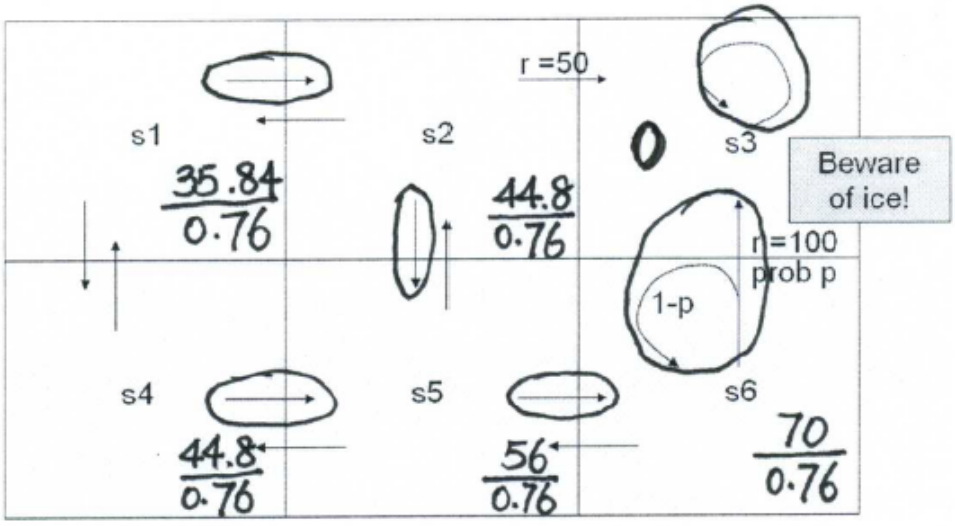
   ★ **SOLUTION:** Yes. Double each each immediate reward. Then V^* is also doubled and π^* remains unchanged

Part B.:

It is December. Unfortunately for our robot, a patch of ice has appeared in its world, making one of its actions non-deterministic. The resulting MDP is shown below. Note that now the result of the action "go north" from state $s6$ results in one of two outcomes. With probability p the robot succeeds in transitioning to state $s3$ and receives immediate reward 100. However, with probability $(1 - p)$, it slips on the ice, and remains in state $s6$ with zero immediate reward. **Assume the discount factor** $\gamma = 0.8$.



1. Assume $p = 0.7$. Write in the values of V^* for each state, and circle the actions in the optimal policy.



★ **SOLUTION:**

$$V_6^* = 100p + \gamma(1-p)V_6^* \Rightarrow V_6^* = 92.1053$$

2. How bad does the ice have to get before the robot will prefer to completely avoid it? Answer this question by giving a value for p below which the optimal policy chooses actions that completely avoid the ice, even choosing the action "go west" over "go north" when the robot is in state $s6$.

★ **SOLUTION:**

$$50\gamma^2 = V_6^* = \frac{100p}{1-\gamma(1-p)}$$
$$p = \frac{\gamma^2 - \gamma^3}{2-\gamma^3} = \frac{8}{93}$$