CSE 512 Machine Learning: Homework 1

Department of Computer Science
Stony Brook University

- There are 6 questions on this assignment. The last question involves coding. Do *not* attach your code to the writeup. Instead, zip all your source files, and upload the .zip file on *Blackboard*. Name your .zip file with your SBU name, e.g. leman.zip
- The assignment is due at 5:30 PM (beginning of class) on **Tue, Feb 25, 2014**.
- Do not forget to put both your name and SBU ID on *each* page of your submission.
- If you have any questions, please direct your question first to the TA, then the instructor.

1 Machine Learning - Problem Setup [10 points]

In online debate forums, people debate issues, express their preferences, and argue why their viewpoint is right. For example, a debate can be "which mobile phone is better: iPhone or Blackberry," or "which OS is better: Windows vs. Linux vs. Mac?" Given a debate forum, machine learning can be applied to:

a. Detect the hot *debate* topics. (Hint: a debate topic is one on which there exist many discussions, with both positive and negative opinions.)

b. For each topic, identify the points of contention within the debate.

c. For a given topic, recognize which stance a person is taking in an online debate posting.

For each of the tasks above: (1) Specify what type of machine learning problem it is (supervised or unsupervised; and regression, classification, or density estimation, etc). (2) Identify what will be (i) the training data,(ii) the features, (iii) the labels (if any), (iv) and what would be the algorithm output. Note that there exist multiple possible answers for this question, depending on how you formulate the problem set up.

2 Probability [10 points]

2.1 Conditional Probability and the Chain Rule [3 points]

Recall the definition of a conditional probability:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

1. Prove that $P(A \cap B \cap C) = P(A|B, C)P(B|C)P(C)$

2.2 Total Probability [7 points]

Suppose that I have two six-sided dice, one is fair and the other one is loaded – having:

$$P(x) = \begin{cases} \frac{1}{2} & x = 6 \\ \frac{1}{10} & x \in \{1, 2, 3, 4, 5\} \end{cases}$$

I will toss a coin to decide which die to roll. If the coin flip is heads I will roll the fair die, otherwise the loaded one. The probability that the coin flip is heads is $p \in (0, 1)$.

1. What is the expectation of the die roll (in terms of p)?
2. What is the variance of the die roll (in terms of p)?

Something commonly used in statistics and machine learning are so called "mixture models" which may be seen as a generalization of the above scenario. For some sample space we have several distributions $P_i(X),\ i = 1 \dots k$ (e.g., the two dice from above). We also have a distribution over these "components" $P(C = i)$ (e.g., the coin toss, where C is a binary random variable).

1. Show the form of $P(X)$ in terms of $P_i(X)$ and $P(C)$.
2. Show the form of $E(X)$ in terms of $E(X|C)$. Make your answer as compact as possible.
3. Show the form of $\text{Var}(X)$ in terms of $\text{Var}(X|C)$ and $E(X|C)$. Make your answer as compact as possible.

3 Parameter Estimation [20 points]

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, packet arrival density is often modeled with the Poisson distribution. That is, if we sit at a computer, count the number of packets arriving in each time interval, say every minute, for 30 minutes, and plot the histogram of how many time intervals had X number of packets, we expect to see something like a Poisson pmf curve.

If X (e.g. packet arrival density) is Poisson distributed, then it has pmf

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

where $\lambda > 0$ is the parameter of the distribution and $X \in \{0, 1, 2, \dots\}$ is the discrete random variable modeling the number of events encountered per unit time.

Note: For the purposes of this problem, everything you need to know about Poisson and Gamma distributions will be provided.

3.1 MLE and MAP estimates [10 points]

It can be shown that the parameter λ is the mean of the Poisson distribution. In this part, we will estimate this parameter from the number of packets observed per unit time $X_1, ..., X_n$ which we assume are drawn i.i.d from $Poisson(\lambda)$.

1. [3 pts] Recall that the *bias* of an estimator of a parameter θ is defined to be the difference between the expected value of the estimator and θ.

   Show that $\hat{\lambda} = \frac{1}{n} \sum_i X_i$ is the maximum likelihood estimate of λ and that it is unbiased (that is, show that $E[\hat{\lambda}] - \lambda = 0$). Recall that $E[a + b] = E[a] + E[b]$ (linearity of expectations).

2. [5 pts] Now let's be Bayesian and put a prior distribution over the parameter λ.

   Your friend in networking hands you a typical plot showing the counts of computers at a university cluster with different average packet arrival densities (Figure 1). Your extensive experience in statistics tells you that the plot resembles a Gamma distribution pdf. So you believe a good prior distribution for λ may be a Gamma distribution.

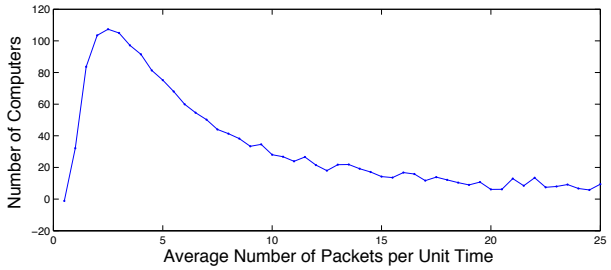


Figure 1: Just giving you some motivation. Don't take it so seriously.

Recall that the Gamma distribution has pdf:

$$p(\lambda|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}, \quad \lambda > 0$$

Also, if $\lambda \sim \Gamma(\alpha,\beta)$, then it has mean α/β and the mode is $(\alpha-1)/\beta$ for $\alpha > 1$.[1]

Assuming that λ is distributed according to $\Gamma(\lambda|\alpha,\beta)$, compute the posterior distribution over λ.

Hint:

$$\lambda^{\sum X_i+\alpha-1}e^{-\lambda(n+\beta)}$$

looks like a Gamma distribution! Is the rest of the expression constant with respect to λ? Working out a messy integral can lead to the answer but shouldn't be necessary.

3. [2 pts] Derive an analytic expression for the maximum a posteriori (MAP) estimate of λ under a $\Gamma(\alpha,\beta)$ prior.

3.2 Estimator Bias/Variance [10 points]

The maximum likelihood estimator is not always unbiased. For example, the maximum likelihood estimator for the variance of a Normal distribution,

$$\hat{\sigma^2}_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i-\hat{\mu})^2$$

is biased - and that an unbiased estimator of variance is:

$$\hat{\sigma^2}_{unbiased} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\hat{\mu})^2$$

For the Normal distribution, these estimators give similar results for large enough N, and it is unclear whether one estimator is preferable to the other. In this problem, we will explore an example in which the maximum likelihood estimate is dramatically superior to any unbiased estimator.

[1] $\Gamma(\alpha)$ refers to the Gamma function, but don't worry if you don't know what this is—it will not be important for this question.

We will again be interested in the Poisson distribution, but instead of estimating the parameter λ, we will estimate a *nonlinear* function of λ, namely $\eta = e^{-2\lambda}$ from a single sample $X \sim Poisson(\lambda)$.

1. [3 pts] Let $\hat{\eta} = e^{-2X}$. Show that $\hat{\eta}$ is the maximum likelihood estimate of η.
2. [4 pts] Show that the bias of $\hat{\eta}$ is $e^{-2\lambda} - e^{\lambda(1/e^2-1)}$.

   The following identity from Taylor expansion may be useful:

$$e^t = \sum_{n=0}^{\infty} \frac{t^n}{n!}$$

3. [3 pts] It turns out that $(-1)^X$ is the *only* unbiased estimate of η. Prove that it is indeed unbiased and briefly explain why this is a bad estimator to use. It may be instructive to plot the values of the MLE and unbiased estimate for $X = 1, ..., 10$.

4 Regression [20 points]

4.1 Linear Models [12 points]

Suppose that you have a software package for linear regression. The linear regression package takes as input a vector of responses (Y) and a matrix of features (X), where the entry $X_{i,j}$ corresponds to the ith data point and the jth feature for that data point and Y_i is the ith response of the function. The linear regression package returns a vector of weights w that minimizes the sum of squared residual errors. The jth entry of the vector, w_j is the weight applied to the jth feature.

For the following functions G_i of the input vector C_i, you should

EITHER

- specify how the response and features (Y_i and $X_{i,j}$) are calculated for the regression software package
- specify how parameters α can be obtained from the values returned by the regression software package w so that α is the maximum likelihood estimate

OR

- provide your reasoning for why the software can not be employed

Example. Given the function $G_i = \sum_{j=0}^{3} \alpha_j C_{i,1}^j + \epsilon_i = \alpha_0 + \alpha_1 C_{i,1} + \alpha_2 C_{i,1}^2 + \alpha_3 C_{i,1}^3 + \epsilon_i$ where $C_{i,1}$ is the first component of C_i and $\epsilon_i \sim N(0, \sigma^2)$, by setting: $X_{i,j} \leftarrow C_{i,1}^j$ for $j = \{0, 1, 2, 3\}$ and $Y_i \leftarrow G_i$ for each i, the software package then returns $w^* = argmin \sum_i (y_i - w_0 - w_1 x_{i,1} - w_2 x_{i,2} - w_3 x_{i,3})^2 = argmin \sum_i (G_i - \sum_{j=0}^{3} w_j C_{i,1}^j)^2$. $\alpha_j \leftarrow w_j$ then is the MLE for each α_j for $j = \{0, 1, 2, 3\}$.

1. [2 pts] $G_i = \alpha_1 C_{i,1}^2 e^{C_{i,2}} + \epsilon_i$ where $C_{i,2}$ is the second component of C_i and $\epsilon_i \sim N(0, \sigma^2)$.
2. [2 pts] $G_i = \alpha_1 C_{i,1}^2 e^{C_{i,2}} + \epsilon_i + \gamma_i$ where $\epsilon_i \sim N(0, \sigma_1^2)$ and $\gamma_i \sim N(\mu, \sigma_2^2)$. Here μ is the unknown bias and must be estimated.
3. [2 pts] $G_i = \sum_j \alpha_j f_j(C_i) + \epsilon_i$ where $f_j(C_i)$ are known basis functions calculated using the input vector C_i and $\epsilon_i \sim N(0, \sigma^2)$
4. [2 pts] $G_i = \sum_j \alpha_{(j\%5)} f_j(C_i) + \epsilon_i$ where "%" is the modulo operator and $\epsilon_i \sim N(0, \sigma^2)$
5. [2 pts] $G_i = \sum_j \alpha_j f_j(C_i|\theta) + \epsilon_i$ where θ is a real valued unknown parameter in the basis functions and $\epsilon_i \sim N(0, \sigma^2)$. You need to estimate both α and θ.

6. [2 pts] $e^{G_i} = \gamma_i[\prod f_j(C_i)^{\alpha_j}]$ where $\gamma_i \sim logNormal(0, \sigma^2)$ and the range of f_j is positive.[2]

4.2 Weighted Least Squares [8 points]

Given instances $< x_i, t_i >$ generated from the linear regression model $t(x) = \sum_i w_i h_i(x_j) + \epsilon_j$, the least squares estimate for the coefficient vector w is given by $w^* = (H^T H)^{-1} H^T t$. If $\epsilon_1, ..., \epsilon_n$ are independent Gaussian with mean 0 and constant standard deviation, the least squares estimate is also the MLE. In the first three questions, assume that $\epsilon_1, ..., \epsilon_n$ are independent Gaussian with mean 0, but the variances are different, i.e. Variance(ϵ_i)= σ_i^2.

1. [1 pts] Give the formulation for calculating the MLE of w.
2. [2 pts] Calculate the MLE of w.
3. [2 pts] Explain why the MLE of w can also be obtained by weighted least squares, i.e. w^* is obtained by minimizing the weighted residual squared error $\sum_j a_j(t_j - \sum_i w_i h_i(x_j))^2$, where a_j is the weights. Give the weights a_j.
4. [2 pts] If $\epsilon_1, ..., \epsilon_n$ are independent Laplace with mean 0 and the same scale parameter b, i.e., the pdf of ϵ_i is $f_{\epsilon_i}(x) = \frac{1}{2b} exp(-\frac{|x|}{b})$, give the formulation for calculating the MLE for w (closed form solution is not required).
5. [1 pts] Sometimes the model in the last question is preferred because its solution tends to be more robust to noise. Explain why this is true.

5 Decision Trees [20 points]

5.1 ID3 and KL Divergence [7 points]

Consider the following set of training examples for the unknown target function $< X_1, X_2 > \rightarrow Y$. Each row indicates the values observed, and how many times that set of values was observed. For example, $(+, T, T)$ was observed 3 times, while $(-, T, T)$ was never observed.

Y	X_1	X_2	Count
+	T	T	3
+	T	F	4
+	F	T	4
+	F	F	1
-	T	T	0
-	T	F	1
-	F	T	3
-	F	F	5

Table 1: Training data

1. [2 pts] Compute the sample entropy $H(Y)$ for this training data (with logarithms base 2)?
2. [3 pts] What are the information gains $IG(X_1) \equiv H(Y) - H(Y|X_1)$ and $IG(X_2) \equiv H(Y) - H(Y|X_2)$ for this sample of training data?

[2]The log-Normal distribution is the distribution of a random variable whose logarithm is normally distributed.

3. [2 pts] Draw the decision tree that would be learned by ID3 (without postpruning) from this sample of training data.

5.2 Information Gain and Entropy [5 points]

When we discussed learning decision trees in class, we chose the next attribute to split on by choosing the one with maximum information gain, which was defined in terms of entropy. To further our understanding of information gain, we will explore its connection to *KL-divergence*, an important concept in information theory and machine learning. For more on these concepts, refer to Section 1.6 in Bishop.

The KL-divergence from a distribution $p(x)$ to a distribution $q(x)$ can be thought of as a measure of dissimilarity from P to Q:

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

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

$$IG(x,y) \equiv KL\left(p(x,y)||p(x)p(y)\right) = -\sum_x \sum_y p(x,y)\ log_2 \left(\frac{p(x)p(y)}{p(x,y)}\right)$$

When the information gain is high, it indicates that adding a split to the decision tree will give a more accurate model.

1. [3 pts] Show that definition of information gain above is equivalent to the one given in class. That is, show that $IG(x,y) = H[x] - H[x|y] = H[y] - H[y|x]$, starting from the definition in terms of KL-divergence.
2. [2 pts] In light of this observation, how can we interpret information gain in terms of dependencies between random variables? A brief answer will suffice.

5.3 Consistent Trees [8 points]

We know that a tree with lower complexity will tend to have better generalization properties. So one (rather simplistic) option to help avoid overfitting is to find the simplest tree that fits the data. This follows the principle known as *Occam's Razor*. One simple way to define "simplest" is based on the *depth* of the tree. Specifically, the depth is the number of nodes along the longest root-to-leaf path. For example, the tree from part 1 would have depth 2. In this problem, we will be interested in learning the tree of least depth that fits the data.

Suppose the training examples are n-dimensional boolean vectors, where $n > 2$ is some constant integer. (For example (T, F, F, T, T) is a 5 dimensional boolean vector). We know that the ID3 decision tree learning algorithm is guaranteed to find a decision tree consistent[3] with any set of (not self-contradicting) training examples, but that doesn't necessarily mean it will find a short tree.

1. [4 pts] For $n = 3$, does ID3 always find a consistent decision tree of depth ≤ 2 if one exists? If so, prove it. If not, provide a counterexample (a set of examples, similar to Table 1 above, but with 3 variables), with an explanation.

[3] A "consistent" tree is one with zero training error.

2. [4 pts] Propose your own learning algorithm that finds a shortest decision tree consistent with any set of training examples (your algorithm can have running time exponential in the depth of the shortest tree). Give the pseudocode and a brief explanation.

6 Naive Bayes vs Logistic Regression [20 points]

In this problem you will implement Naive Bayes and Logistic Regression, then compare their performance on a document classification task. The data for this task is taken from the 20 Newsgroups data set[4], and is available at `http://www.cs.stonybrook.edu/~leman/courses/14CSE512/hws/hw1-data.tar.gz`. The included `README.txt` describes the data set and file format.

Our Naive Bayes model will use the bag-of-words assumption. This model assumes that each word in a document is drawn independently from a multinomial distribution over possible words. (A multinomial distribution is a generalization of a Bernoulli distribution to multiple values.) Although this model ignores the ordering of words in a document, it works surprisingly well for a number of tasks. We number the words in our vocabulary from 1 to m, where m is the total number of distinct words in all of the documents. Documents from class y are drawn from a class-specific multinomial distribution parameterized by θ_y. θ_y is a vector, where $\theta_{y,i}$ is the probability of drawing word i and $\sum_{i=1}^{m} \theta_{y,i} = 1$. Therefore, the class-conditional probability of drawing document x from our Naive Bayes model is $P(X = x|Y = y) = \prod_{i=1}^{m} (\theta_{y,i})^{\text{count}_i(x)}$, where $\text{count}_i(x)$ is the number of times word i appears in x.

1. [5 pts] Provide high-level descriptions of the Naive Bayes and Logistic Regression algorithms. Be sure to describe how to estimate the model parameters and how to classify a new example.

2. [3 pts] Imagine that a certain word is never observed in the training data, but occurs in a test instance. What will happen when our Naive Bayes classifier predicts the probability of the this test instance? Explain why this situation is undesirable. Will logistic regression have a similar problem? Why or why not?

   Add-one smoothing is one way to avoid this problem with our Naive Bayes classifier. This technique pretends that every word occurs one additional time in the training data, which eliminates zero counts in the estimated parameters of the model. For a set of documents $C = x^1, ..., x^n$, the add-one smoothing parameter estimate is $\hat{\theta}_i = \frac{1+\sum_{j=1}^{n} \text{count}_i(x^j)}{D+m}$, where D is the total number of words in C (i.e., $D = \sum_{i=1}^{m} \sum_{j=1}^{n} \text{count}_i(x^j)$). Empirically, add-one smoothing often improves classification performance when data counts are sparse.

3. [10 pts] Implement Logistic Regression and Naive Bayes. Use add-one smoothing when estimating the parameters of your Naive Bayes classifier. For logistic regression, we found that a step size around .0001 worked well. Train both models on the provided training data and predict the labels of the test data. Report the training and test error of both models. Submit your code electronically on *Blackboard* under SafeAssignments. You do *not* need to include a hard copy of your code along with your HW submission.

4. [2 pts] Which model performs better on this task? Why do you think this is the case?

[4]Full version available from http://people.csail.mit.edu/jrennie/20Newsgroups/

CSE 512 Machine Learning: Homework I

Mari Wahl, marina.w4hl at gmail

1 Machine Learning - Problem Setup [10 points]

In online debate forums, people debate issues, express their preferences, and argue why their viewpoint is right. For example, a debate can be "which mobile phone is better: iPhone or Blackberry," or "which OS is better: Windows vs. Linux vs. Mac?" Given a debate forum, machine learning can be applied to:

a. Detect the hot *debate* topics. (Hint: a debate topic is one on which there exist many discussions, with both positive and negative opinions.)

Solutions:

Each post in the forum would be an instance of our data. They could be indexed, for example, by **post message**, by **author**, and by **time**. We also add an index for the **topic of the post**, which would relate groups of posts. If we are using many forums, we can also add the **forum name** to the data, which would not make any difference for the items below (but could give some additional statistical information if desired).

To be able to find the hot topics, we start by adding a grade to the opinion of the post: **positive**, **negative**, or **impartial**. A hot topic is defined as a topic that has many responses which must be either positive and negative (not impartial). These response must be by many authors (although some authors can respond more than once, they will in general maintain the same opinion). A range of time can also be stipulated for the valid posts (some discussions lose their validity or *hotness* after some time).

1. **Type of machine learning: Unsupervised learning**
2. **Algorithm output:** The task of detecting what are the hot topics can be done with **density estimation**.

■

b. For each topic, identify the points of contention within the debate.

Solution:

For each hot topic found in the item above, we find the points of contention by searching for the features that are mentioned within the disagreement though the posts. For example, when debating about phones, if one post says something positive about, for example, the camera of a model, and then another point has a negative comment about this characteristic, this is a contention point. Each post will be indexed by a different author.

1. **Type of machine learning: Unsupervised learning**.
2. **Algorithm output:** task of detecting what are the points of contention for the topics can be done with **density estimation**.

c. For a given topic, recognize which stance a person is taking in an online debate posting.

Solution:

After the results from the previous items, we have a set of what are the point of the contentions, and these become the features. This time we are actually using the labels in each features to be able to recognize the stances people are taking in each post. In other words, for each post, we can find if the opinion for these feature are positive, negative, or impartial, then for each author, we can learn what are their stances (note that not all the features will have a corresponding value for every post, so this should be considered in the algorithm).

1. **Type of machine learning: Supervised learning**.
2. **Training data:** Chose some posts within the hot topics, containing the points of contention.
3. **Features:** Characteristics such as usability, appearance, price, operational system, etc (extracted from the points of contention).
4. **Labels:** Positive, negative, and impartial.
5. **Algorithm output:** Depending on the majority of positive or negative labels for the features, we can define a criteria to recognize the stance of each user as positive, negative, or impartial. For example, the simplest case would be saying that if there are more negatives than positives, the person's stance is negative, or if they amount the same, the person is impartial. We could also attribute weights for the features that seems to be more relevant or more recurring.

2 Probability [10 points]

2.1 Conditional Probability and the Chain Rule [3 points]

Theoretical Introduction:

The expression $P(A)$ denotes the probability that the **event** A is **true**, where $0 \leq P(A) \leq 1$. Given two events, A and B, we define the **probability of** A **or** B, as

$$\boxed{P(A \text{ or } B) = P(A \cup B) = P(A) + P(B) - P(A \text{ and } B) = P(A) + P(B) - P(A \cap B).}$$

If A and B are events that are **impossible** to occur at the same time, they are called **disjoints**,

$$P(A \text{ and } B) = P(A \cap B) = 0.$$

In this case, the probability above reduces to:

$$P(A \text{ or } B) = P(A \cup B) = P(A) + P(B).$$

If A and B are **not disjoints**, the **joint probability** of the joint events A and B, also called **product rule**, is given by

$$\boxed{P(A \text{ and } B) = P(A \cap B) = P(A, B) = P(A)P(B|A) = P(B)P(A|B).}$$

If A and B are **stochastically independent**,

$$P(B|A) = P(B),$$

and the probability above expression reduces to:

$$P(A \text{ and } B) = P(A \cap B) = P(A)P(B).$$

If $P(B) > 0$ we can define the **conditional probability** of event A, given that event B is true, as:

$$\boxed{P(A|B) = \frac{P(A, B)}{P(B)} = \frac{P(A \cap B)}{P(B)}.}$$

Given a **joint distribution** on two events $P(A \text{ and } B) = P(A, B)$, the **marginal distribution** is

$$\boxed{P(A) = \sum_{b \in Im(B)} P(A, B) = \sum_{b} P(A|B = b)P(B = b),}$$

where we are summing over all possible states of B. This is called **the sum rule** or the **rule of total probability**.

Solution:

Let A, B, and C be random variables representing three different events[1]. We now are going to proof the equality:

$$P(A \cap B \cap C) = P(A|B,C)P(B|C)P(C). \tag{1}$$

Let us define the random variable $\kappa = B \cap C$. The conditional probability for events B and C in terms of κ is given by:

$$P(B \cap C) = P(B|C)P(C) = P(\kappa).$$

Now, the conditional probability for events A and κ is:

$$P(A \cap \kappa) = P(A \cap B \cap C) = P(A|\kappa)P(\kappa),$$

which is just

$$\begin{aligned} P(A|\kappa)P(\kappa) &= P(A|B \cap C)P(B \cap C) \\ &= P(A|B,C)P(B|C)P(C). \end{aligned}$$

Since probability multiplications are **commutative** (as any real-valued function in a unidimensional space), we complete the proof of Eq. 1. ■

A second way to perform the same proof is by **induction**, using the equation for conditional probability:

$$\begin{aligned} P(A \cap B \cap C) &= P(A \cap B \cap C) \times \frac{P(B \cap C)}{P(B \cap C)} \\ &= P(B \cap C)\frac{P(A \cap B \cap C)}{P(B \cap C)} \\ &= P(B \cap C)P(A|B \cap C) \\ &= P(C)P(B|C)P(A|B,C) \end{aligned}$$

It is possible to generalize this result for N events. ■

2.2 Total Probability [7 points]

Suppose that I have two six-sided dice, one is fair and the other one is loaded. having:

$$P(x) = \begin{cases} \frac{1}{2} & x = 6 \\ \frac{1}{10} & x \in \{1,2,3,4,5\} \end{cases}$$

I will toss a coin to decide which die to roll. If the coin flip is heads I will roll the fair die, otherwise the loaded one. The probability that the coin flip is heads is $p \in (0,1)$.

1. What is the expectation of the die roll (in terms of p)?
2. What is the variance of the die roll (in terms of p)?

[1] A **random variable** is a numerical outcome of the experiment, *i.e.,* a real-valued function whose domain is the **sample space**.

Theoretical Introduction:

In a probabilistic model of an experiment, a **random variable** is a real-valued function of the outcome of the experiment. In this case, we define the outcome of the coin-dice experiment by the random variable X (*i.e.,* it will take the possible outcomes for the dice).

A **discrete** random variable has an associated **probability mass function (pfm)**, which gives the probability of each numerical value that the random variable can take. In this problem, the pfm for the fair dice is:

$$p_{\text{fair dice}}(x) = \frac{1}{2},\ x \in \{1,2,3,4,5,6\}.$$

The pfm for the loaded dice is:

$$p_{\text{loaded}}(x) = \begin{cases} \frac{1}{2}, & x = 6 \\ \frac{1}{10}, & x \in \{1,2,3,4,5\} \end{cases}$$

The pfm for the coin is:

$$p_{coin}(c) = \begin{cases} p, & c = \text{head} \\ (1-p), & c = \text{tail} \end{cases}$$

A function of a discrete random variable defines another discrete random variable, whose pfm can be obtained from the pfm of the original random variables. Therefore, we have:

$$p_{\text{coin}-\text{dice}}(x,c) = \begin{cases} p \times \frac{1}{6}, & \text{if } x = 6 \text{ and } c = \text{head} \\ (1-p) \times \frac{1}{2}, & \text{if } x = 6 \text{ and } c = \text{tail} \\ p \times \frac{1}{6}, & \text{if } x \in \{1,2,3,4,5\} \text{ and } c = \text{head} \\ (1-p) \times \frac{1}{10}, & \text{if } x \in \{1,2,3,4,5\} \text{ and } c = \text{tail} \end{cases}$$

In terms of the random variable X:

$$p_{\text{coin}-\text{dice}}(x) = \begin{cases} \frac{p}{6} + \frac{(1-p)}{2}, & \text{if } x = 6 \\ \frac{p}{6} + \frac{(1-p)}{10}, & \text{if } x \in \{1,2,3,4,5\} \end{cases}$$

Note that

$$\sum_x p_{\text{coin}-\text{dice}}(x) = 6 \times \frac{p}{6} + 5 \times \frac{(1-p)}{10} + \frac{(1-p)}{2} = p + (1-p) = 1.$$

The pfm of the random variable X, given by $p_{\text{coin}-\text{dice}}(x)$, provided us with several numbers, *i.e.,* the probabilities of all its possible outcomes. To summarize this information in a single representative number, we calculate the **expectation value** or **mean** of X which is a weighted (in proportion to probabilities) average of the possible values of X:

$$\boxed{E[X] = \sum_x x p_X(x).}$$

The **variance** of a random variable X is defined as the expected value of the random variable $(X - E[X])^2$, and it is a measure of the **spread** of the distribution, *i.e.,* how much X varies around the expected value:

$$\boxed{\text{var}[X] = E\Big[(X - E[X])^2\Big].}$$

If we open the squares,

$$\text{var}[X] = E[X^2] - E[X]^2.$$

In the discrete case, we can also calculate the variance with

$$\text{var}[X] = \sum_x \Big(x - E[X]\Big)^2 \, p_X(x).$$

Solution:

We are now ready to solve our problem. Using the pmf defined above, the expected value of the dice is:

$$\begin{aligned} E[X] &= \sum_X x \, p_{\text{coin}-\text{dice}}(x) \\ &= 6 \times \Big[\frac{p}{6} + \frac{(1-p)}{2}\Big] + (1+2+3+4+5) \times \Big[\frac{p}{6} + \frac{(1-p)}{10}\Big], \end{aligned}$$

which results on

$$\boxed{E[X] = 4.5 - p.}$$

Let us analyze this result. Suppose both of the dice were fair (*i.e.,* , each of the 6 outcome had 1/6 chance to be seem), or we had only one die. Then the result would not depend on p. The expected value would be:

$$E'[X]_{\text{fair}-\text{dice}} = (1+2+3+4+5+6) \times \frac{1}{6} \times p + (1+2+3+4+5+6) \times \frac{1}{6} \times (1-p) = 3.5.$$

Back to our problem, since one of the dice is loaded, the result now has a weight in p. Now, suppose the coin is fair, *i.e.,* $p = 1/2$. In this case, the expected value for the dice is:

$$E'[X]_{p=0.5} = 4.5 - 0.5 = 4.$$

This makes sense because in 50% of the cases we would deal with the loaded die, which has a larger weight (*i.e.,* it increases the value of the expected value since in 1/4 of the cases we would be seeing 6).

Moreover, the value of p is actually bounded between $p \in \{0, 1\}$, so if the coin is loaded and it takes the boundary values, we would have:

$$E'[X]_{p=0} = 4.5 - 0 = 4.5,$$

in which case, we would only use the loaded die (and 1/2 of the cases we would see 6), and

$$E'[X]_{p=1} = 4.5 - 1 = 3.5,$$

in which case, we would only use the fair die (and we would recover the first result, when both dice were fair). Note that despite the $-p$ in the expectation value, probabilities are never negative, and the expectation values return what the average of the results would be asymptotically.

Now let us plug the previous result into the equation for variance of X:

$$\text{var}[X] = E[X^2] - (4.5 - p)^2,$$

where

$$\begin{aligned} E[X^2] &= 6^2 \times \left[\frac{p}{6} + \frac{(1-p)}{2}\right] + (1^2 + 2^2 + 3^2 + 4^2 + 5^2) \times \left[\frac{p}{6} + \frac{(1-p)}{10}\right] \\ &= 6(3-2p) + \frac{11}{6}(3+2p) \\ &= -\frac{25p}{3} + \frac{47}{2}. \end{aligned}$$

Plugging back in the variance equation,

$$\boxed{\text{var}[X] = -\frac{25p}{3} + \frac{47}{2} - \frac{81}{4} + 9p - p^2 = \frac{2}{3}p - p^2 + \frac{13}{4}.}$$

Let us analyze this result. Supposing that $p = 1$, we would only use the fair die. In this case, $E[X] = 3.5$ (as shown before) and $\text{var}[X] = \frac{2}{3} - 1 + \frac{13}{4} = 2.92$. This matches to the variance found in a fair die. For the other boundary value, when $p = 0$, we find $E[X] = 4.5$ and $\text{var}[X] = 3.25$. For a fair coin, when $p = 1/2$, $E[X] = 4.0$ and $\text{var}[X] = 3.3$. ■

Something commonly used in statistics and machine learning is so called **mixture models** which may be seen as a generalization of the above scenario. For some sample space we have several distributions $P_i(X)$, $i = 1 \ldots k$ (e.g., the two dice from above). We also have a distribution over these "components" $P(C = i)$ (e.g., the coin toss, where C is a binary random variable).

1. Show the form of $P(X)$ in terms of $P_i(X)$ and $P(C)$.

Theoretical Introduction:

Let $\{P_i(x)\}_{i \in A}$ be a collection of distributions for the random variable X, and let C be a **discrete** random variable taking values in A. The **mixture distribution** of the $P_i(x)$ with weights given by the distribution of C is defined as:

$$P(X) = \sum_{i \in A} P_C^i \, P_i(x).$$

In other words, a random variable X having probability $P(X)$ arises first of the random variable C and then, if $C = i$, it gets X from the distribution $P_i(x)$.

Solution:

For our problem, we say that a distribution $P(X)$ is a **mixture** of the two dice's distributions, $P_i(X)$, with mixing proportions p and $1 - p$ (given by the coin). In this case, $A = \{0, 1\}$ and $P(C)$ assumes the values $P_C^0 = 1 - p$ and $P_C^1 = p$:

$$\boxed{P(X) = \sum_{i \in \{0,1\}} P(C = i)P(X|C = i) = \sum_{i \in \{0,1\}} P(C = i)P_i(X).}$$

Note that this distribution will result in a constant when we select one of events $X = x$, $x \in \{0, 1, 2, 3, 4, 5, 6\}$:

$$P(X = x) = \sum_{i \in \{0,1\}} P(C = i)P_i(X = x).$$

■

2. Show the form of $E(X)$ in terms of $E(X|C)$.

Theoretical Introduction:

For this, let us derive the **Theorem of Total Expectation**. Remember that if T is an integer-value random variable, some function $L = h(T)$ is another random variable, with expected value:

$$E(L) = \sum_k h(k)P(T = k).$$

Solution:

For our problem, the random variables X and C take values only in the set $i \in \{1, .., k\}$. For an event, say $C = i$, the quantity $E(X|C = i)$ is the long-run average of X, among the times when $C = i$ occurs. Now, we define a function $g(i) = E(X|C = i)$ (a constant, not a random variable). The quantity $E(X|C)$ is defined to be a new random variable Q, which is a projection in an abstract vector space. Since Q is a function of C, we find its expectation from the distribution of C:

$$\begin{aligned}
E[E[X|C]] &= E(Q) \\
&= \sum_i g(i)P(C = i) \\
&= \sum_i E(X|C = i)P(C = i) \\
&= \sum_i \left[\sum_j jP(X = j|C = i) \right] P(C = i) \\
&= \sum_j j \sum_i P(X = j|C = i)P(C = i) \\
&= \sum_j jP(X = j) \\
&= E(X).
\end{aligned}$$

Resulting in:

$$\boxed{E(X) = E[E[X|C]].}$$

■

3. Show the form of $\mathrm{Var}(X)$ in terms of $\mathrm{Var}(X|C)$ and $E(X|C)$.

Solution:

For this, let us derive the **Law of Total Variance**. First of all, the variance of the conditional expectation of X given C is:

$$\mathrm{var}[X|C] = E[X^2|C] - E[X|C]^2.$$

This has an expectation value of:

$$E[\mathrm{var}[X|C]] = E[E[X^2|C]] - E[E[X|C]^2] = E[X^2] - E[E[X|C]^2].$$

Now, knowing that $E[E(X|C)] = E[C]$, we calculate the variance of $E[X|C]$:

$$\mathrm{var}[E[X|C]] = E[E[X|C]^2] - E[X]^2.$$

Adding all together:

$$\begin{aligned} \mathrm{var}[E[X|C]] + E[\mathrm{var}[X|C]] &= E[E[X|C]^2] - E[X]^2 + E[X^2] - E[E[X|C]^2] \\ &= E[X^2] - E[X]^2, \end{aligned}$$

which is simple, the variance of X:

$$\boxed{\mathrm{var}(X) = E[\mathrm{var}(X|C)] + \mathrm{var}[E(X|C)].}$$

■

3 Parameter Estimation [20 points]

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, packet arrival density is often modeled with the Poisson distribution. That is, if we sit at a computer, count the number of packets arriving in each time interval, say every minute, for 30 minutes, and plot the histogram of how many time intervals had X number of packets, we expect to see something like a Poisson PMF curve.

If X (e.g. packet arrival density) is Poisson distributed, then it has PMF:

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

where $\lambda > 0$ is the parameter of the distribution and $X \in \{0, 1, 2, \ldots\}$ is the discrete random variable modeling the number of events encountered per unit time.

3.1 MLE and MAP estimates [10 points]

It can be shown that the parameter λ is the **mean** of the Poisson distribution. In this part, we will estimate this parameter from the number of packets observed per unit time $X_1, ..., X_n$ which we assume are drawn i.i.d from $Poisson(\lambda)$.

Theoretical Introduction:

We are interested in estimating parametric models of the form

$$y_i \sim f(\theta, y_i),$$

where θ is a vector of parameters and f is some specific functional form (probability density or mass function). In this example we are using the Poisson distribution :

$$y_i \sim f(\lambda, X_i) = \frac{e^{-\lambda}\lambda^{X_i}}{X_i!},$$

which have only one parameter to estimate, $\theta = \lambda$.

In general, we have some observations on X and we want to estimate the mean and the variance from the data. The idea of **maximum likelihood** is to find the **estimate of the parameter(s)** of the chosen distribution that maximize the probability of observing the data we observe.

In other words, given $y_i \sim f(\theta, y_i)$ we would like to make **inferences** about the value of θ. This is the inverse of a typical probability problem, where we want to know something about the distribution of y given the parameters of our model, θ, *i.e.,* $P(y|\theta)$ or $P(data|model)$. Here, we have the data by we want to learn about the model, specifically, the model's parameters, *i.e.,* $P(model|data)$ or $P(\theta|y)$.

From the probability identities we can derive the **Bayes' Theorem**,

$$P(\theta, y) = P(\theta)P(y|\theta) = P(y)P(\theta|y),$$

where the conditional density of $p(\theta|y)$ is given by:

$$\boxed{P(\theta|y) = \frac{P(\theta, y)}{P(y)} = \frac{P(\theta)P(y|\theta)}{P(y).}}$$

The denominator, $P(y)$, is just a function of the data. Since it only makes sense to compare these conditional densities from the same data, we can ignore the denominator (for instance, $1/P(y)$ is called the constant of proportionality). Rewriting the above equation gives:

$$\boxed{P(\theta|y) \propto P(\theta)P(y|\theta),}$$

where $P(\theta)$ is the **prior density of** θ, $P(y|\theta)$ is the **likelihood**, and $P(\theta|y)$ is the **posterior density of** θ. In other words, the likelihood is the sample information that transform the prior to the posterior density of θ. The prior is fixed before the observations.

Without knowing the prior, we would not be able to calculate the inverse probability described above. In this case, we introduce the notion of likelihood, and the **Likelihood Axiom** defines:

$$\mathcal{L}(\theta|y) \propto P(y|\theta).$$

The likelihood is proportional to the probability of observing the data, treating the parameters of the distribution as variables and the data as fixed. The advantage of likelihood is that it can be calculated from a traditional probability, $P(y|\theta)$. We can only compare likelihoods for the same set of data and the same prior. The best **estimator**, $\hat{\theta}$, is the value of θ that maximizes

$$\mathcal{L}(\theta|y) = P(y|\theta),$$

i.e., we are looking for the $\hat{\theta}$ that maximizes the likelihood of observing our sample, and this will maximize $P(\theta|y)$ (the probability of observing the data).

If the y_i are all independent (or conditionally independent), then the likelihood of the whole sample is the product of the individual likelihoods over all the observations:

$$\begin{aligned}
\mathcal{L} &= \mathcal{L}_1 \times \mathcal{L}_2 \times ... \times \mathcal{L}_N \\
&= \prod_{i=1}^{N} \mathcal{L} \\
&= P(y_1|\theta) \times P(y_2|\theta) \times ... \times P(y_N|\theta) \\
&= \prod_{i=1}^{N} P(y_i|\theta).
\end{aligned}$$

The negative log of the likelihood function is called **error function**. Because the negative logarithm is a monotonically decreasing function, maximizing the likelihood is equivalent to minimizing the error:

$$\boxed{\ln \mathcal{L} = \sum_{i=1}^{N} P(y_i|\hat{\theta}).}$$

From the log-likelihood, we find $\hat{\theta}^{ML}$, which can sometimes be obtained analytically by differentiating the function with respect to the parameter vector, and setting the resulting gradient vector to zero. Mostly commonly these calculations are done numerically.

1. [3 pts] Recall that the *bias* of an estimator of a parameter θ is defined to be the difference between the expected value of the estimator and θ.

   (a) Show that $\hat{\lambda} = \frac{1}{n}\sum_i X_i$ is the maximum likelihood estimate of λ.

Solution:

For our network example, we derive $\hat{\theta}^{ML}$ analytically using the Poisson distribution as the underlying distribution for a count model:

$$y_i \sim f(\theta, y_i) = \frac{e^{-\lambda}\lambda^X}{X!}.$$

Since the N samples are i.i.d., the likelihood function is:

$$\begin{aligned}
\mathcal{L} = P(\mathcal{D}|\lambda) &= \prod_{i=1}^{N} \frac{e^{-\lambda}\lambda^{X_i}}{X_i!} \\
&= \frac{\prod_{i=1}^{N} e^{-\lambda} \prod_{i=1}^{N} \lambda^{X_i}}{\prod_{i=1}^{N} X_i!} \\
&= \frac{e^{-N\lambda}\lambda^{\sum_{i=1}^{N} X_i}}{\prod_{i=1}^{N} X_i!}.
\end{aligned}$$

The log-likelihood function is:

$$\begin{aligned}
\ln \mathcal{L} &= \sum_{i=1}^{N} \ln\left(\frac{e^{-\lambda}\lambda^{X_i}}{X_i!}\right) \\
&= \sum_{i=1}^{N} \Big[-\lambda + X_i \ln \lambda - \ln(X_i!) \Big] \\
&= -N\lambda + \ln(\lambda) \sum_{i=1}^{N} X_i - \sum_{i=1}^{N} \ln(X_i!)
\end{aligned}$$

In other words,

$$\hat{\lambda} = \arg\max_{\lambda} P(\mathcal{D}|\lambda) = \arg\max_{\lambda} \ln P(\mathcal{D}|\lambda),$$

which can be found by taking the derivative with respect to $\theta^{\mathrm{ML}} = \hat{\lambda}$:

$$\frac{\partial \ln \mathcal{L}}{\partial \hat{\lambda}} = -N + \frac{1}{\hat{\lambda}} \sum_{i=1}^{N} X_i,$$

and solving for $\hat{\lambda}$:

$$-N + \frac{1}{\hat{\lambda}} \sum_{i=1}^{N} X_i = 0.$$

Resulting in:

$$\boxed{\hat{\lambda} = \frac{\sum_{i=1}^{N} X_i}{N}.}$$

To verify that this is maximum, we take the second derivative:

$$\frac{\partial^2 \ln \mathcal{L}}{\partial \lambda^2} = \frac{\partial}{\partial \lambda}\left(-N + \frac{1}{\lambda} \sum_{i=1}^{N} X_i \right) = -\frac{1}{\lambda^2} \sum_{i=1}^{N} X_i < 0,$$

which is negative, so the result has a local maximum at λ. ■

(b) Show that it is unbiased (that is, show that $E[\hat{\lambda}] - \lambda = 0$). Recall that $E[a + b] = E[a] + E[b]$ (linearity of expectations).

Theoretical Introduction:

Let θ be a parameter of interest and $\hat{\theta}$ be an **estimator** of θ based on a sample of size N. The **bias** of $\hat{\theta}$ is the difference between θ and the long run **average of the estimates** given by $\hat{\theta}$, based on different samples of size N:

$$\boxed{\text{bias}(\hat{\theta}) = E[\hat{\theta}] - \theta.}$$

An estimator is called **unbiased** if $\text{bias}(\hat{\theta}) = 0$. A biased estimator systematically overestimates or underestimates the value of θ.

Solution:

The estimator of the parameter θ is taking over X. We have shown that the **standard estimator** for a Poisson population **mean** is the maximum likelihood estimator:

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i = \hat{\lambda},$$

then

$$\begin{aligned} E[\hat{\lambda}] &= E\Big[\frac{1}{N}\sum_{i=1}^{N} X_i\Big] \\ &= \frac{\sum_{i=1}^{N} E[X_i]}{N} \\ &= \frac{\sum_{i_1}^{N} \lambda}{N} \\ &= \frac{N\lambda}{N} \\ &= \lambda, \end{aligned}$$

where we have used the fact that:

$$\begin{aligned} E[X] &= \sum_{x \in Im(X)} xP(X = x) \\ &= \sum_{i \geq 0}^{N} x_i \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} \\ &= \lambda e^{-\lambda} \sum_{i \geq 1}^{N} \frac{\lambda^{x_i - 1}}{(x_i - 1)!} \\ &= \lambda e^{-\lambda} \sum_{j \geq 0} \frac{\lambda^j}{j!} \\ &= \lambda e^{-\lambda} e^{\lambda} \\ &= \lambda. \end{aligned}$$

Note that in the last step we have used the **Taylor expansion** for the exponential, assuming that the number of samples, N, can be taken asymptotically to be infinity: $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$. Plugging the result of $E[\hat{\lambda}]$ back into the bias equation, we show that the Poisson distribution is unbiased:

$$\boxed{\text{bias}(\hat{\lambda}) = E[\hat{\lambda}] - \lambda = \lambda - \lambda = 0.}$$

2. [5 pts] Now let's be Bayesian and put a prior distribution over the parameter λ.

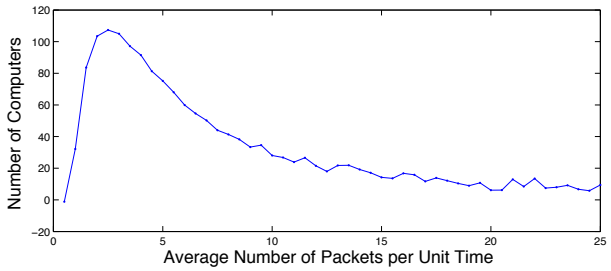


Figure 1: Just giving you some motivation. Don't take it so seriously.

Your friend in networking hands you a typical plot showing the counts of computers at a university cluster with different average packet arrival densities (Figure 1). Your extensive experience in statistics tells you that the plot resembles a Gamma distribution pdf. So you believe a good prior distribution for λ may be a Gamma distribution. Recall that the Gamma distribution has pdf:

$$\boxed{P(\lambda|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}, \quad \lambda > 0}$$

Also, if $\lambda \sim \Gamma(\alpha,\beta)$, then it has mean α/β and the mode is $(\alpha-1)/\beta$ for $\alpha > 1$.[2]

Assuming that λ is distributed according to $\Gamma(\lambda|\alpha,\beta)$, compute the posterior distribution over λ.

Hint:

$$\lambda^{\sum X_i+\alpha-1}e^{-\lambda(n+\beta)}$$

looks like a Gamma distribution! Is the rest of the expression constant with respect to λ?

Solution:

We have shown above that the Bayesian formulation requires:

$$P(\lambda|y) \propto P(\lambda)P(y|\lambda),$$

where $P(\lambda)$ is the **prior density of** λ, $P(y|\lambda)$ is the **likelihood**, and $P(\lambda|y)$ is the **posterior density of** λ. We have also derived the likelihood function for the Poisson distribution:

$$\begin{aligned} P(y|\lambda) &= \prod_{i=1}^{N} \frac{e^\lambda \lambda^{X_1}}{X_i!} \\ &= \frac{\prod_{i=1}^{N} e^\lambda \prod_{i=1}^{N} \lambda^{X_1}}{\prod_{i=1}^{N} X_i!} \\ &= \frac{e^{N\lambda}\lambda^{\sum_{i=1}^{N} X_1}}{\prod_{i=1}^{N} X_i!}. \end{aligned}$$

[2] $\Gamma(\alpha)$ refers to the Gamma function, but don't worry if you don't know what this is—it will not be important for this question.

Now assuming that the prior distribution of λ is a Gamma function,

$$\boxed{P(\lambda|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}, \quad \lambda > 0,}$$

we calculate the posterior distribution over λ

$$\boxed{P(\lambda|y) \propto \left(\frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}\right)\left(\frac{e^{-N\lambda}\lambda^{\sum_{i=1}^N X_1}}{\prod_{i=1}^N X_i!}\right) = \left(\frac{\beta^\alpha}{\Gamma(\alpha)\prod_{i=1}^N X_i!}\right)\left(\lambda^{\alpha-1+\sum_{i=1}^N X_1}e^{-(N+\beta)\lambda}\right).}$$

It is clear that the function inside of the first parenthesis is a constant over λ and the data, and the function inside the second parenthesis is a Gamma distribution with parameters $\hat{\alpha} = \alpha + \sum_i X_i$ and $\hat{\beta} = \beta + N$. We can rewrite this result as:

$$\boxed{P(\lambda|y) \propto \lambda^{\hat{\alpha}-1}e^{-\hat{\beta}\lambda} = \left(\lambda|\alpha + \sum_i X_i, \beta + N\right) = \Gamma(\hat{\alpha},\hat{\beta}).}$$

Finally, from the Law of probability, we can calculate the normalizing constant:

$$\sum_\lambda \text{ constant } \times \lambda^{\alpha+\sum_i X_i}e^{-(\beta+N)\lambda} = 1.$$

■

3. [2 pts] Derive an analytic expression for the maximum a posteriori (MAP) estimate of λ under a $\Gamma(\alpha,\beta)$ prior.

Theoretical Introduction:

The **maximum a posteriori** (MAP) estimate can be computed in several ways. Analytically, it can be calculated when the **mode** of the posterior distribution can be given in a **closed form**[3]. In this case, the **conjugate priors** are used, as for example, for the Poisson distribution, which has the Gamma distribution as its conjugate prior.

The mode is the point where the probability distribution has the highest probability, and in the case of a Gamma function it is given by:

$$\Gamma(\alpha,\beta) = \frac{\alpha-1}{\beta},$$

for $\alpha > 1$.

Solution:

For our problem, we see that if α was greater than 1, $\hat{\alpha} = \alpha + \sum_{x=1}^N X_i$ would also be greater than 1. The MAP estimation is then the mode of the posterior distribution, *i.e.,* the mode of $\Gamma(\sum_i X_i + \alpha, N + \beta)$:

$$\boxed{\text{mode } = \frac{\alpha-1}{\beta} = \frac{\sum_i X_i + \alpha - 1}{N+\beta}.}$$

■

[3] A closed-form expression is an expression that can be written analytically in terms of a finite number of certain *well-known* functions.

3.2 Estimator Bias/Variance [10 points]

The maximum likelihood estimator is not always unbiased. For example, the maximum likelihood estimator for the variance of a Normal distribution,

$$\hat{\sigma^2}_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$$

is biased - and that an unbiased estimator of variance is:

$$\hat{\sigma^2}_{unbiased} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$$

For the Normal distribution, these estimators give similar results for large enough N, and it is unclear whether one estimator is preferable to the other. In this problem, we will explore an example in which the maximum likelihood estimate is dramatically superior to any unbiased estimator.

We will again be interested in the Poisson distribution, but instead of estimating the parameter λ, we will estimate a *nonlinear* function of λ, namely $\eta = e^{-2\lambda}$ from a single sample $X \sim Poisson(\lambda)$.

1. [3 pts] Let $\hat{\eta} = e^{-2X}$. Show that $\hat{\eta}$ is the maximum likelihood estimate of η.

Solution:

We calculate the MLE estimation in the same way we did in the previous item, but now setting X the distribution parameter to $\lambda(\eta)$:

$$\begin{aligned} e^{-2\lambda} &\rightarrow \eta \\ e^{2\lambda} &= \eta^{-1} \\ 2\lambda &= -\ln\eta \\ \lambda(\eta) &= -\frac{1}{2}\ln\eta. \end{aligned}$$

Plugging $\lambda(\eta)$ back to the Poisson distribution equation:

$$\begin{aligned} P(X|\lambda(\eta)) &= \frac{\lambda(\eta)^X e^{-\lambda(\eta)}}{X!} \\ &= \frac{1}{X!}\Big(-\frac{1}{2}\ln\eta\Big)^X e^{\frac{1}{2}\ln\eta}. \end{aligned}$$

We take the log:

$$\ln P(X|\lambda(\eta)) = X\ln(-\frac{1}{2}\ln\eta) + \frac{1}{2}\ln\eta - \ln(X!),$$

and then we find the minimum by setting the first derivative (with respect of η) to zero:

$$\begin{aligned}
\frac{\partial \ln P(X|\hat{\eta})}{\partial \hat{\eta}} &= 0 \\
X \frac{1}{-\frac{1}{2}\ln\hat{\eta}} \frac{-1}{2\hat{\eta}} + \frac{1}{2\hat{\eta}} &= 0 \\
\frac{X}{\hat{\eta}\ln\hat{\eta}} &= -\frac{1}{2\hat{\eta}} \\
-2X &= \ln\hat{\eta} \\
\hat{\eta} &= e^{-2X}.
\end{aligned}$$

This result also can be proved by using the fact that the MLE is **invariant** under functional transformations. That is, if $\hat{\lambda}$ is the MLE of λ and if $\eta(\lambda)$ is a function of λ, then $\eta(\hat{\lambda})$ is the MLE of $\eta(\lambda)$. ■

2. [4 pts] Show that the bias of $\hat{\eta}$ is $e^{-2\lambda} - e^{\lambda(1/e^2-1)}$.

   The following identity from Taylor expansion may be useful:

$$e^t = \sum_{n=0}^{\infty} \frac{t^n}{n!}$$

Solution:

Recalling that $\text{bias}(\hat{\eta}) = E[\hat{\eta}] - \eta$, we calculate $E[\hat{\eta}]$ as we did before(*i.e.,* using Taylor series for the exponential):

$$\begin{aligned}
E[\hat{\eta}] &= \sum_{X\geq 0} \hat{\eta} P(X) \\
&= \sum_{X\geq 0} e^{-2X} \frac{\lambda^X e^{-\lambda}}{X!} \\
&= e^{-\lambda} \sum_{X\geq 0} \frac{(\lambda e^{-2})^X}{X!} \\
&= e^{-\lambda} e^{\lambda e^{-2}} \\
&= e^{(e^{-2}-1)\lambda}.
\end{aligned}$$

The bias is:

$$\boxed{\text{bias}(\hat{\eta}) = E[\hat{\eta}] - \eta = e^{(e^{-2}-1)\lambda} - e^{-2\lambda}.}$$

■

3. [3 pts] It turns out that $(-1)^X$ is the *only* unbiased estimate of η. Prove that it is indeed unbiased and briefly explain why this is a bad estimator to use. It may be instructive to plot the values of the MLE and unbiased estimate for $X = 1, ..., 10$.

Solution:

Performing the same calculations as before, assuming $\hat{\eta} = (-1)^X$:

$$\begin{aligned} E[\hat{\eta}] &= \sum_{X\geq 0} \hat{\eta} P(X) \\ &= \sum_{X\geq 0} (-1)^X \frac{\lambda^X e^{-\lambda}}{X!} \\ &= e^{-\lambda} \sum_{X\geq 0} \frac{(-\lambda)^X}{X!} \\ &= e^{-\lambda} e^{-\lambda} \\ &= e^{-2\lambda}. \end{aligned}$$

The bias is zero:

$$\boxed{\text{bias}(\hat{\eta}) = E[\hat{\eta}] - \eta = e^{-2\lambda} - e^{-2\lambda} = 0.}$$

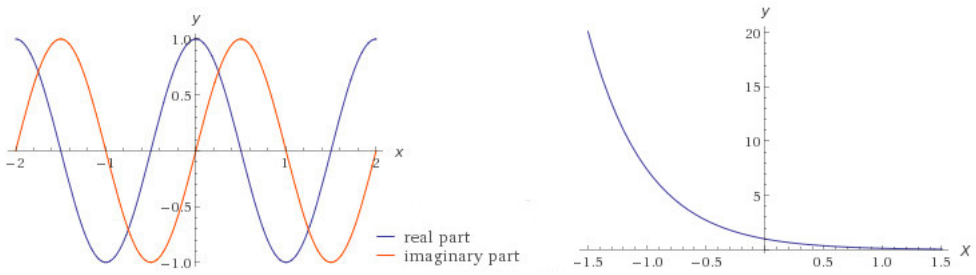


Figure 2: Left: The oscillatory behavior of $\hat{\eta} = (-1)^X$. Right: $\hat{\eta} = e^{-2X}$

The bias is zero: The main characteristic of this estimator is that it will oscillate between negative and positive values, depending on whether X is even or odd, *i.e.,* it is extremely dependent of the data. Second, since we are dealing with probabilities, every time X is odd and $\hat{\eta}$ takes negative values, the probability can be ill-defined. ■

4 Regression [20 points]

4.1 Linear Models [12 points]

Suppose that you have a software package for linear regression. The linear regression package takes as input a vector of responses (Y) and a matrix of features (X), where the entry $X_{i,j}$ corresponds to the ith data point and the jth feature for that data point and Y_i is the ith response of the function. The linear regression package returns a vector of weights w that minimizes the sum of squared residual errors. The jth entry of the vector, w_j is the weight applied to the jth feature.

For the following functions G_i of the input vector C_i, you should

EITHER

- specify how the response and features (Y_i and $X_{i,j}$) are calculated for the regression software package
- specify how parameters α can be obtained from the values returned by the regression software package w so that α is the maximum likelihood estimate

OR

- provide your reasoning for why the software can not be employed

Example. Given the function $G_i = \sum_{j=0}^{3} \alpha_j C_{i,1}^j + \epsilon_i = \alpha_0 + \alpha_1 C_{i,1} + \alpha_2 C_{i,1}^2 + \alpha_3 C_{i,1}^3 + \epsilon_i$ where $C_{i,1}$ is the first component of C_i and $\epsilon_i \sim N(0, \sigma^2)$, by setting: $X_{i,j} \leftarrow C_{i,1}^j$ for $j = \{0, 1, 2, 3\}$ and $Y_i \leftarrow G_i$ for each i, the software package then returns $w^* = argmin \sum_i (y_i - w_0 - w_1 x_{i,1} - w_2 x_{i,2} - w_3 x_{i,3})^2 = argmin \sum_i (G_i - \sum_{j=0}^{3} w_j C_{i,1}^j)^2$. $\alpha_j \leftarrow w_j$ then is the MLE for each α_j for $j = \{0, 1, 2, 3\}$.

Theoretical Introduction:

In **linear regression**, we have a **training set** of N observations, $\mathbf{x} = (x_1, .., x_N)^T$, together with the corresponding values of the **target vector**, $\mathbf{t} = (t_1, ..., t_N)^T$. The goal is to exploit this training set to make predictions of the value $\hat{t}$ of the target variable for some new $\hat{x}$ of the input variable. The data is also corrupted with noise.

We solve the curve fitting problem choosing the value of $\mathbf{w}$ for which $E(\mathbf{w})$ is small as possible. The error function is a quadratic function on $\mathbf{w}$, having derivatives linear in $\mathbf{w}$, so that the minimization of the error function has a unique solution, denoted by $\mathbf{w}^*$ (which can be found in **closed form**). The resulting polynomial is given by $y(x, \mathbf{w}^*)$.

1. [2 pts] $G_i = \alpha_1 C_{i,1}^2 e^{C_{i,2}} + \epsilon_i$ where $C_{i,2}$ is the second component of C_i and $\epsilon_i \sim N(0, \sigma^2)$.

   ### Solution:

   Following the example, we set the response to

   $$\boxed{Y_i \leftarrow G_i},$$

   and the feature to

   $$\boxed{X_{i,1} \leftarrow C_{i,1}^2 e^{C_{i,2}}},$$

for each i, where $C_{i,1}$ and $C_{i,2}$ are the first and second components of the input vector C_i, respectively. The software will return

$$w^* = \text{argmin} \sum_i (y_i - w_1 x_{i,1}),$$

so that the maximum likelihood estimate is

$$\boxed{\alpha_1 \leftarrow w_1}.$$

■

2. [2 pts] $G_i = \alpha_1 C_{i,1}^2 e^{C_{i,2}} + \epsilon_i + \gamma_i$ where $\epsilon_i \sim N(0, \sigma_1^2)$ and $\gamma_i \sim N(\mu, \sigma_2^2)$. Here μ is the unknown bias and must be estimated.

Solution:

We now have the unknown bias to be estimated. Using the theory from the next exercise for Gaussians with different variances, we will use the first weight, w_0, assuming that $X_{i,0} \leftarrow 1$. In this case we set the response to

$$\boxed{Y_i \leftarrow G_i},$$

and the features to

$$\boxed{X_{i,0} \leftarrow 1, \;\; X_{i,1} \leftarrow C_{i,1}^2 e^{C_i,2},}$$

for each i, where $C_{i,1}$ and $C_{i,2}$ are the first and second components of the input vector C_i, respectively. The software will return

$$w^* = \text{argmin} \sum_i (y_i - w_0 x_{i,0} - w_1 x_{i,1}),$$

so that the maximum likelihood estimate are

$$\boxed{\mu \leftarrow w_0,}$$

and

$$\boxed{\alpha_1 \leftarrow w_1.}$$

■

3. [2 pts] $G_i = \sum_j \alpha_j f_j(C_i) + \epsilon_i$ where $f_j(C_i)$ are known basis functions calculated using the input vector C_i and $\epsilon_i \sim N(0, \sigma^2)$

Solution:

This case is similar to the example, except that now $f_j(C_i)$ are *basis functions* and the dimension of the data is not given. In this case, the response is set to

$$\boxed{Y_i \leftarrow G_i},$$

and the feature is set to

$$\boxed{X_{i,1} \leftarrow f_j(C_i)},$$

for each i. The software will return

$$w^* = \text{argmin} \sum_i (y_i - \sum_j w_j x_{i,j}),$$

so the maximum likelihood estimates are

$$\boxed{\alpha_j \leftarrow w_j.}$$

■

4. [2 pts] $G_i = \sum_j \alpha_{(j\%5)} f_j(C_i) + \epsilon_i$ where "%" is the modulo operator and $\epsilon_i \sim N(0, \sigma^2)$

Solution:

In this example we also have basis functions to the input vectors. First, the response is set to

$$\boxed{Y_i \leftarrow G_i}.$$

This time the parameters α_j are divide in five sets, for all possible values returned in the $j\%5$ operation, *i.e.,* $j = 0, 1, 2, 3, 4$. To use them for the maximum likelihood estimates, we incorporate these sets into the basis functions so the features are

$$\boxed{X_{i,j} \leftarrow \sum_j f_{j\%5}(C_i).}$$

The software will return

$$w^* = \text{argmin} \sum_i (y_i - \sum_j w_j x_{i,j}),$$

so the maximum likelihood estimate is

$$\boxed{\alpha_j \leftarrow w_j.}$$

■

5. [2 pts] $G_i = \sum_j \alpha_j f_j(C_i|\theta) + \epsilon_i$ where θ is a real valued unknown parameter in the basis functions and $\epsilon_i \sim N(0, \sigma^2)$. You need to estimate both α and θ.

Solution:

In this case the software should not be employed. Since this is a linear regression software, it only works for models that are a linear combination of their parameters (*i.e.,* linear in θ and α). Since we do not know the form of the basis functions, they could be non-linear in θ. ■

6. [2 pts] $e^{G_i} = \gamma_i[\prod f_j(C_i)^{\alpha_j}]$ where $\gamma_i \sim logNormal(0, \sigma^2)$ and the range of f_j is positive.[4]

Solution:

To solve this problem, we apply the log function for both sides:

$$\begin{aligned} \ln e^{G_i} &= \ln \left[\gamma_i \prod f_j(C_i)^{\alpha_j} \right] \\ G_i &= \ln \gamma_i + \sum_j \alpha_j \ln f_j(C_i), \end{aligned}$$

[4]The log-Normal distribution is the distribution of a random variable whose logarithm is normally distributed.

where $\ln \gamma_i \sim N(0, \sigma^2)$. The response is:

$$\boxed{Y_i \leftarrow G_i}.$$

The features are

$$\boxed{X_{i,j} \leftarrow \ln f_j(C_i).}$$

The software will return

$$w^* = \text{argmin} \sum_i (y_i - \sum_j w_j x_{i,j}),$$

so the maximum likelihood estimate is

$$\boxed{\alpha_j \leftarrow w_j.}$$

■

4.2 Weighted Least Squares [8 points]

Given instances $< x_i, t_i >$ generated from the linear regression model

$$t(x) = \sum_i w_i h_i(x_j) + \epsilon_j,$$

the least squares estimate for the coefficient vector w is given by

$$w^* = (H^T H)^{-1} H^T t.$$

If $\epsilon_1, ..., \epsilon_n$ are independent Gaussian with mean 0 and constant standard deviation, the least squares estimate is also the MLE. In the first three questions, assume that $\epsilon_1, ..., \epsilon_n$ are independent Gaussian with mean 0, but the variances are different, i.e. Variance(ϵ_i)= σ_i^2.

1. [1 pts] Give the formulation for calculating the MLE of w.

Solution:

From class, models with different variances (*i.e.*, different $\epsilon_1, ..., \epsilon_n$), have conditional likelihood for the data is given by:

$$P(\mathcal{D}|\mathbf{w}, \sigma) = \prod_{j=1}^{N} \frac{1}{\sqrt{2\pi} \cdot \sigma_j} \exp\left(- \frac{\left(t_j - \sum_i \mathbf{w}_i h_i(x_j)\right)^2}{2\sigma_j^2} \right).$$

Taking the log-likelihood, we have:

$$\begin{aligned} \ln\Big(P(\mathcal{D}|\mathbf{w}, \sigma)\Big) &= \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N \ln\left(\prod_{j=1}^{N} \exp\Big(- \frac{\left(t_j - \sum_i \mathbf{w}_i h_i(x_j)\right)^2}{2\sigma_j^2}\Big)\right) \\ &= \text{C} - \sum_{j=1}^{R} \frac{\left(t_j - \sum_i \mathbf{w}_i h_i(x_j)\right)^2}{2\sigma_j^2}. \end{aligned}$$

Finally, the MLE of w is:

$$\mathbf{w}^* = \text{argmin}_w \sum_{j=1}^{R} \frac{\left(t_j - \sum_i \mathbf{w}_i h_i(x_j)\right)^2}{2\sigma_j^2}.$$

■

2. [2 pts] Calculate the MLE of w.

Solution:

(a) We set Σ as the diagonal matrix, with diagonal elements $\sigma_1^2, .., \sigma_N^2$.

(b) We set M as the number of basis functions.

(c) We set H as the $N \times M$ matrix, with $H_{ji} = h_i(x_j)$ elements.

(d) We set w as the $M \times 1$ vector, where the i^{th} element is w_i.

(e) We set t as the $N \times 1$ vector, where the j^{th} element is t_j.

(f) Thus (as shown in class):

$$\sum_{j=1}^{N} \frac{\left(t_j - \sum_i w_i h_i(x_j)\right)^2}{2\sigma_j^2} = \frac{1}{2}(t - Hw)^T \Sigma^{-1}(t - Hw).$$

(g) Now, taking the derivative with respect to w gives:

$$-H^T \Sigma^{-1}(t - Hw) = 0,$$

(h) Recovering w^*, the MLE of w:

$$w^* = (H^T \Sigma^{-1} H)^{-1}(H^T \Sigma^{-1} t).$$

■

3. [2 pts] Explain why the MLE of w can also be obtained by weighted least squares, i.e. w^* is obtained by minimizing the weighted residual squared error $\sum_j a_j(t_j - \sum_i w_i h_i(x_j))^2$, where a_j is the weights. Give the weights a_j.

Solution:

With the results in the item (1):

$$w^* = \text{argmin}_w \sum_{j=1}^{N} \frac{\left(t_j - \sum_i w_i h_i(x_j)\right)^2}{2\sigma_j^2},$$

we set $\alpha_j = \frac{1}{2\sigma_j^2}$, so that the MLE is obtained by *minimizing the weighted residual squared error*:

$$\sum_{j=1}^{N} \alpha_j \Big(t_j - \sum_i w_i h_i(x_j) \Big)^2.$$

■

4. [2 pts] If $\epsilon_1, ..., \epsilon_n$ are independent Laplace with mean 0 and the same scale parameter b, i.e., the pdf of ϵ_i is $f_{\epsilon_i}(x) = \frac{1}{2b} exp(-\frac{|x|}{b})$, give the formulation for calculating the MLE for w (closed form solution is not required).

Solution:

Using the same logic as before, the conditional data likelihood is given by:

$$P(\mathcal{D}|\mathbf{w}, \sigma) = \prod_{j=1}^{N} \frac{1}{2b} \exp\left(\frac{-|t_j - \sum_i w_i h_i(x_j)|}{b} \right).$$

The log of this likelihood is:

$$\ln P(\mathcal{D}|\mathbf{w}, \sigma) = \mathbf{C} - \sum_{j=1}^{N} \frac{|t_j - \sum_i w_i h_i(x_j)|}{b},$$

where $\mathbf{C}$ is a constant. Finally, the MLE of w is given by:

$$\begin{aligned} w^* &= \text{argmax}_w \sum_{j=1}^{N} \frac{|t_j - \sum_i w_i h_i(x_j)|}{b} \\ &= \text{argmax}_w \sum_{j=1}^{N} |t_j - \sum_i w_i h_i(x_j)|. \end{aligned}$$

■

5. [1 pts] Sometimes the model in the last question is preferred because its solution tends to be more robust to noise. Explain why this is true.

Solution:

If the data has noise, this will result in large residuals. In this case, squared models (ϵ_j given by a Gaussian distribution) have larger residuals than linear models (ϵ_j with Laplace distribution), therefore, the former would be less robust. ■

5 Decision Trees [20 points]

5.1 ID3 and KL Divergence [7 points]

Consider the following set of training examples for the unknown target function $< X_1, X_2 > \rightarrow Y$. Each row indicates the values observed, and how many times that set of values was observed. For example, $(+, T, T)$ was observed 3 times, while $(-, T, T)$ was never observed.

Y	X_1	X_2	Count
+	T	T	3
+	T	F	4
+	F	T	4
+	F	F	1
-	T	T	0
-	T	F	1
-	F	T	3
-	F	F	5

Table 1: Training data

1. [2 pts] Compute the sample entropy $H(Y)$ for this training data (with logarithms base 2)?

Theoretical Introduction:

A measure of information content called **self-information** of a message **m** is given by:

$$I(m) = \lg\left(\frac{1}{p(m)}\right) = -lg(p(m)),$$

where $p(m) = Pr(M = m)$ is the probability that the message m is chosen from all possible choices in the message space M. The **entropy** of a discrete message space M is a measure of the amount of uncertainty one has about which message will be chosen, and it is defined as the average self-information of a message m from that message space. In probability language, the entropy of a random variable x is then defined as[5]

$$\boxed{H[x] = -\sum_x p(x) \lg p(x).}$$

Solution:

The sample entropy is then:

$$\begin{aligned} H(Y) &= -\sum_{i=1}^{k} p(Y = y_i) \lg p(Y = y_i) \\ &= -p_- \lg p_- - p_+ \lg p_+ \\ &= -\frac{4}{7} \lg \frac{4}{7} - \frac{3}{7} \lg \frac{3}{7} \\ &\approx 0.98523 \end{aligned}$$

[5] We could express the entropy in any logarithm basis, they would be up to a constant. We choose logarithm base 2, represented by lg.

2. [3 pts] What are the information gains $IG(X_1)$ and $IG(X_2)$ for this sample of training data?

Theoretical Introduction:

Given entropy as a measure of the impurity in a collection of training examples, we define a measure of the effectiveness of an attribute in classifying the training data. The measure is called **information gain** and is the expected reduction in entropy caused by partitioning the examples according to this attribute:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in \{A\}} \frac{S_v}{S} \; Entropy(S_v).$$

For this case we can express the information gain based on X_1 by computing the entropy of Y after a split on X_1, given by $H(Y|X_1)$:

$$\boxed{IG(X_1) \equiv H(Y) - H(Y|X_1)}.$$

The same thing for X_2:

$$\boxed{IG(X_2) \equiv H(Y) - H(Y|X_2).}$$

Solution:

Since $H(Y)$ was already calculated in the item above, we need to find $H(Y|X_1)$ and $H(Y|X_2)$. For this, we first calculate all the probabilities:

$$\begin{aligned}
p(X_1 = T) &= \frac{8}{21} \\
p(X_1 = F) &= \frac{13}{21} \\
p(Y = +|X_1 = T) &= \frac{7}{8} \\
p(Y = +|X_1 = F) &= \frac{5}{13} \\
p(Y = -|X_1 = T) &= \frac{1}{8} \\
p(Y = -|X_1 = F) &= \frac{8}{13}
\end{aligned}$$

and

$$\begin{aligned} p(X_2 = T) &= \frac{10}{21} \\ p(X_2 = F) &= \frac{11}{21} \\ p(Y = +|X_2 = T) &= \frac{1}{10} \\ p(Y = +|X_2 = F) &= \frac{5}{11} \\ p(Y = -|X_2 = T) &= \frac{3}{10} \\ p(Y = -|X_2 = F) &= \frac{6}{11} \end{aligned}$$

Now we are able to calculate the conditional entropies:

$$\begin{aligned} H(Y|X_1) &= -\sum_{j=1}^{v} p(X_1 = x_j) \sum_{i=1}^{k} p(Y = y_i|X_1 = x_j) \lg p(Y = y_i|X_1 = x_j) \\ &= -p(X_1 = T)\Bigg(p(Y = +|X_1 = T) \lg p(Y = +|X_1 = T) + \\ &\quad +p(Y = -|X_1 = T) \lg p(Y = -|X_1 = T)\Bigg) - \\ &\quad -p(X_1 = F)\Bigg((p(Y = +|X_1 = F) \lg p(Y = +|X_1 = F) + \\ &\quad +p(Y = -|X_1 = F) \lg p(Y = -|X_1 = F)\Bigg) \\ &= -8/21\left(\frac{7}{8} \lg \frac{7}{8} + \frac{1}{8} \lg \frac{1}{8}\right) - \frac{13}{21}\left(\frac{5}{13} \lg \frac{5}{13} + \frac{8}{13} \lg \frac{8}{13}\right) \\ &\approx 0.802123 \end{aligned}$$

and

$$\begin{aligned} H(Y|X_2) &= -\sum_{j=1}^{v} p(X_2 = x_j) \sum_{i=1}^{k} p(Y = y_i|X_2 = x_j) \lg(Y = y_i|X_2 = x_j) \\ &= -p(X_2 = T)\Bigg(p(Y = +|X_2 = T) \lg p(Y = +|X_2 = T) \\ &\quad +p(Y = -|X_2 = T) \lg p(Y = -|X_2 = T)\Bigg) \\ &\quad -p(X_2 = F)\Bigg((p(Y = +|X_2 = F) \lg p(Y = +|X_2 = F) \\ &\quad +p(Y = -|X_2 = F) \lg p(Y = -|X_2 = F)\Bigg) \\ &= -10/21\left(\frac{7}{10} \lg \frac{7}{10} + \frac{3}{10} \lg \frac{3}{10}\right) - \frac{11}{21}\left(\frac{5}{11} \lg \frac{5}{11} + \frac{6}{11} \lg \frac{1}{13}\right) \\ &\approx 0.9403448 \end{aligned}$$

Resulting in:

$$\begin{aligned} IG(X_1) &= H(Y) - H(Y|X_1) \\ &\approx 0.98523 - 0.802123 \\ &\approx 0.183107 \end{aligned}$$

and

$$\begin{aligned} IG(X_2) &= H(Y) - H(Y|X_2) \\ &\approx 0.98523 - 0.9403448 \\ &\approx 0.044885 \end{aligned}$$

Therefore

$$\boxed{IG(X_2) < IG(X_1).}$$

■

3. [2 pts] Draw the decision tree that would be learned by ID3 (without postpruning) from this sample of training data.

Solution:

To build the decision tree we start by finding the attribute X with highest information gain. In this case, it is $IG(X_1) \approx 0.1831$, so our first split will be on feature X_1.

Now we need to look to the left and right branches. Since splitting on X_2 results in two nodes with non-zero values, we split both branches. The final decision tree can bee seen in Fig. 3. ■

5.2 Information Gain and Entropy [5 points]

When we discussed learning decision trees in class, we chose the next attribute to split on by choosing the one with maximum information gain, which was defined in terms of entropy. To further our understanding of information gain, we will explore its connection to *KL-divergence*, an important concept in information theory and machine learning. For more on these concepts, refer to Section 1.6 in Bishop.

The KL-divergence from a distribution $p(x)$ to a distribution $q(x)$ can be thought of as a measure of dissimilarity from P to Q:

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

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

$$IG(x,y) \equiv KL\left(p(x,y)||p(x)p(y)\right) = -\sum_x \sum_y p(x,y)\ log_2 \left(\frac{p(x)p(y)}{p(x,y)}\right)$$

When the information gain is high, it indicates that adding a split to the decision tree will give a more accurate model.

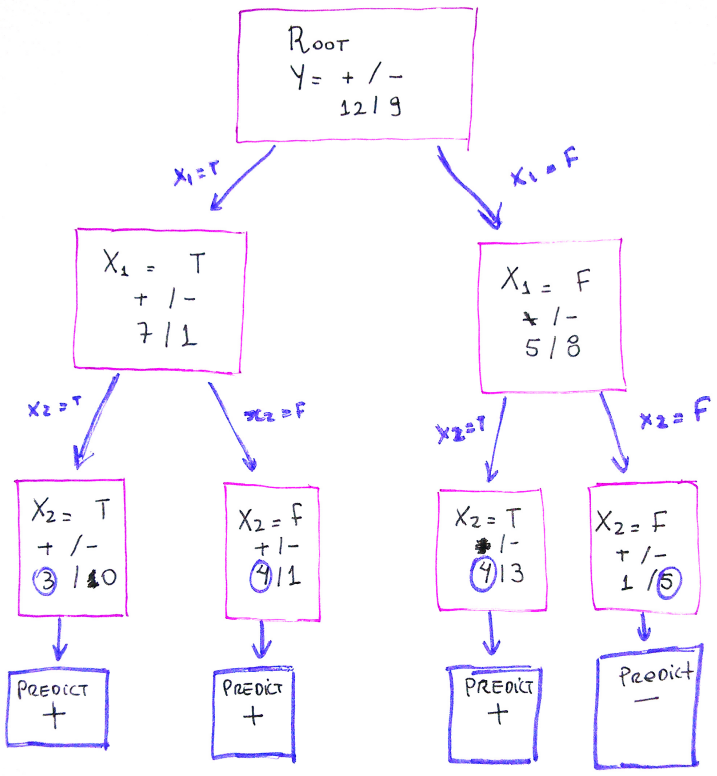


Figure 3: Learned decision tree.

1. [3 pts] Show that definition of information gain above is equivalent to the one given in class. That is, show that $IG(x, y) = H[x] - H[x|y] = H[y] - H[y|x]$, starting from the definition in terms of KL-divergence.

Solution:

Using the definitions above, the formula for conditional probability, and the fact that the sum of the probabilities of all the outcomes in a sample space is equal to one, we have:

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

The same derivation can be used to prove that $H(x) - H(x|y)$, using the fact that $p(x)p(y|x) = p(y)p(x|y)$. ■

2. [2 pts] In light of this observation, how can we interpret information gain in terms of dependencies between random variables? A brief answer will suffice.

Solution:

The **relative entropy** or **Kullback-Leibler divergence** between two distributions is a measure of the distance between the two distributions. The joint distribution between two sets of variables, X and Y is given by $p(x, y)$. If the sets of variables are independent, their joint distribution will factorize to the product of their marginals $p(X, Y) = p(X)p(Y)$. If the variable are not independent, we gain some idea of they are closer to being independent by considering the KL distance between the joint distribution and the product of marginals.

This **mutual information** is related to the conditional entropy as shown above, so we can see the mutual information as the reduction in the uncertainty about X by knowing the value of Y (and vice versa). In other words, the information gain measures the distance between the joint distribution of X and Y and the product distribution, resulting on how far X and Y are from being independent (and resulting in the extreme value 0 if they are independent). ■

5.3 Consistent Trees [8 points]

We know that a tree with lower complexity will tend to have better generalization properties. So one (rather simplistic) option to help avoid overfitting is to find the simplest tree that fits the data. This follows the principle known as *Occam's Razor*. One simple way to define "simplest" is based on the *depth* of the tree. Specifically, the depth is the number of nodes along the longest root-to-leaf path. For example, the tree from part 1 would have depth 2. In this problem, we will be interested in learning the tree of least depth that fits the data.

Suppose the training examples are n-dimensional boolean vectors, where $n > 2$ is some constant integer. (For example (T, F, F, T, T) is a 5 dimensional boolean vector). We know that the ID3 decision tree learning algorithm is guaranteed to find a decision tree consistent[6] with any set of (not self-contradicting) training examples, but that doesn't necessarily mean it will find a short tree.

1. [4 pts] For $n = 3$, does ID3 always find a consistent decision tree of depth ≤ 2 if one exists? If so, prove it. If not, provide a counterexample (a set of examples, similar to Table 1 above, but with 3 variables), with an explanation.

Solution:

We do not always find a consistent decision tree of smaller depth. For example, let's look to the table below, which adds a third feature to the Table 2:

We show two possible decision trees in the Figs. 4. In the first case, the tree splits on X_3 as the root, and since we need both X_1 and X_2, the tree has depth 3. In the second case, the tree splits on X_1 as the root split, and since X_3 does not add any additional information (the predictions do not change if we add a rule on X_3), we only need X_2 to be have a consistent tree. ■

[6] A "consistent" tree is one with zero training error.

Y	X_1	X_2	X_3	Count
+	T	T	T	2
+	T	T	F	0
+	T	F	T	1
+	T	F	F	1
+	F	T	T	1
+	F	T	F	1
+	F	F	T	2
+	F	F	F	0
-	T	T	T	4
-	T	T	F	0
-	T	F	T	0
-	T	F	F	0
-	F	T	T	0
-	F	T	F	0
-	F	F	T	4
-	F	F	F	0

Table 2: Training data with three features.

2. [4 pts] Propose your own learning algorithm that finds a shortest decision tree consistent with any set of training examples (your algorithm can have running time exponential in the depth of the shortest tree). Give the pseudocode and a brief explanation.

Solution:

We suppose the data is given as an array of the rows of the above training data:

$$data = [[+, T, T, T, 2], [+, T, T, F, 0]....]],$$

i.e., data[0] is the first row, data[1] is the second row, data[15] is the last row. Also, data[0][0] = +, data[0][1] = T (X_1 in the first row), and data[0][4] = 2 (the number of counts). The algorithm bellow has two recursive calls and only two loops, so the recurrence is bellow super-exponential (in case we had to search for all the depths of the tree):

```
main(data):
  # find number of attributes
  nX = len(data[0])-2
  m = 0
  while m < = nX:
    tree = findShortest(data, m, nX)
    if != tree:
      m += 1
  return tree or False


findShortest(data, m, nX):
  # Base case:
  if m == 0:
    Inspect all data subsets, if labels are equal, return leaf with label
  else:
    return False
```

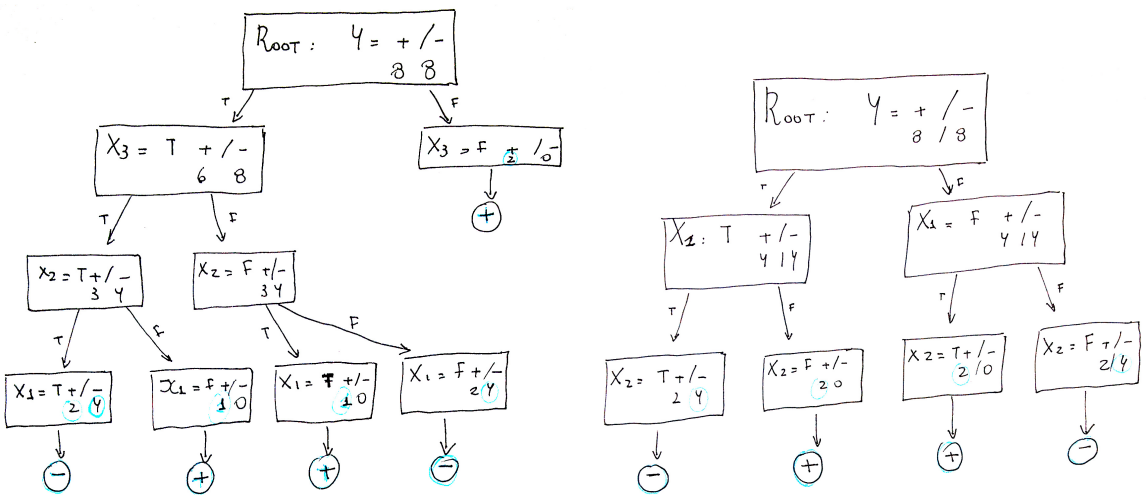


Figure 4: An ID3 tree with three features. (Left) The training data in a depth 3 ID3, with X_3 as the root. (Right) Shows the same training data, in a consistent ID3 tree, with depth 2 by splitting on X_1.

```
  for att in (nX):
    # find the subsets the attributes are T, F and have counts:
    data-T =[data[:][att + 1], data[:][nX-1] where data[:][att + 1] == True and
data[:][nX-1] != 0]
    data-F = [data[:][att + 1], data[:][nX-1] where data[:][att + 1] == False and
data[:][nX-1] != 0]
    tree-left = findShortest(data-T, m-1, nX)
    tree-right = findShortest(data-F, m-1, nX)
    if tree-left and tree-right:
      # return tree with split in this attribute, and subroots found above:
      return tree(attribute, tree-left, tree-right)
    # In case the above failed:
  return False
```

6 Naive Bayes vs Logistic Regression [20 points]

In this problem you will implement Naive Bayes and Logistic Regression, then compare their performance on a document classification task. The data for this task is taken from the 20 Newsgroups data set[7], and is available at `http://www.cs.stonybrook.edu/~leman/courses/14CSE512/hws/hw1-data.tar.gz`. The included `README.txt` describes the data set and file format.

Our Naive Bayes model will use the bag-of-words assumption. This model assumes that each word in a document is drawn independently from a multinomial distribution over possible words. (A multinomial distribution is a generalization of a Bernoulli distribution to multiple values.) Although this model ignores the ordering of words in a document, it works surprisingly well for a number of tasks. We number the words in our vocabulary from 1 to m, where m is the total number of distinct words in all of the documents. Documents from class y are drawn from a class-specific multinomial distribution parameterized by θ_y. θ_y is a vector, where $\theta_{y,i}$ is the probability of drawing word i and $\sum_{i=1}^{m} \theta_{y,i} = 1$. Therefore, the class-conditional probability of drawing document x from our Naive Bayes model is $P(X = x|Y = y) = \prod_{i=1}^{m} (\theta_{y,i})^{\text{count}_i(x)}$, where $\text{count}_i(x)$ is the number of times word i appears in x.

1. [5 pts] Provide high-level descriptions of the Naive Bayes and Logistic Regression algorithms. Be sure to describe how to estimate the model parameters and how to classify a new example.

Solution:

Naive Bayes

The **Naive Bayes algorithm** is a classification algorithm based on the Bayes rule, that assume the attributes $X_1 ... X_n$ to be all conditionally independent of each other given Y. Assuming that Y is any discrete-valued variable, and the attributes $X_1 ... X_n$ are any discrete or real-valued attributes, the algorithm to train a classifier that will output the probability distribution over possible values of Y, for each new instance X, is:

$$P(Y = y_k|X_1 ... X_n) = \frac{P(Y = y_k) \prod_i P(X_i|Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i|Y = y_j)}.$$

In other words, given a new X^{new}, this equation will calculate the probability that Y will take on any given value, given the observed attribute values of X^{new} and the given distributions $P(Y)$ and $P(X_i|Y)$.

The **most probable value** of Y is given by:

$$Y \leftarrow \text{argmax}_{y_k} P(Y = y_k) \prod_i P(X_i|Y = y_k).$$

We can summarize the Naive Bayes learning algorithm by describing the parameters to be estimated. When the n input attributes X_i take on J possible values, and Y is a discrete variable taking on K possible values, then the learning correspond to estimate **two set of parameters**:

$$\theta_{ijk} = P(X_i = x_{ijk}|Y = y_k),$$

for each input X_i, each of its possible values x_{ij}, and each of possible values y_k of Y; and:

$$\pi_k = P(Y = y_k),$$

[7] Full version available from http://people.csail.mit.edu/jrennie/20Newsgroups/

the prior probability over Y. We can estimate these parameters using **maximum likelihood estimates** for θ_{ijk} given a set of training examples D:

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij}|Y = y_k) = \frac{\#D(X_i = x_{ij} \wedge Y = y_k)}{\#D(Y = y_k)},$$

where $\#D$ returns the number of elements in set D with property x (without smoothing). The **maximum likelihood estimate** for $\hat{\pi}_k$ is (without smoothing):

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D(Y = y_k)}{|D|}.$$

Logistic Regression

Logistic regression is an approach to learning functions of the form $P(Y|X)$, in the case where Y is discrete-valued, and X is a vector containing discrete or continuous values. Logistic Regression assumes a **parametric form** for the distribution $P(Y|X)$, then directly estimates its parameters from the training data. The parametric model in case where Y is boolean is:

$$P(Y = 0|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)},$$

and

$$P(Y = 1|X) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}.$$

To classify any given X we assign the value y_k that maximizes $P(Y = y_k|X)$. In other words, we assign the label $Y = 1$ if the following conditions holds:

$$\begin{aligned} 1 &< \frac{P(Y = 0|X)}{P(T = 1|X)} \\ 1 &< \exp(w_0 + \sum_{i=1}^{N} w_i X_i) \\ 0 &< w_0 + \sum_{i=1}^{N} w_i X_i, \end{aligned}$$

where we took the log in the last line. We assign $Y = 0$ otherwise.

One approach to training Logistic Regression is to choose parameter values that maximize the **conditional data likelihood**, which is the probability of the observed Y values in the training data, conditioned on their corresponding X values. We choose parameters W that satisfy

$$W \leftarrow \arg\max_w \prod_l P(Y^l|X^l, W),$$

where W is the vector of parameters to be estimated; Y^l denotes the observed value of Y in the lth training example; and X^l is the observed value of X in the lth training example. The **conditional data log likelihood** is then:

$$\begin{aligned} l(W) &= \sum_l Y^l \ln P(Y_l = 0|X^l, W) + (1 - Y^l) \ln P(Y^l = 1|X^l, W), \\ &= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + \exp(w_0 + \sum_i^n w_i X_i^l)). \end{aligned}$$

where Y can only be 0 or 1.

Since there is no closed form solution to maximizing $l(W)$ with respect to W, we use the **gradient ascent** (vector of partial derivatives). Beginning with initial weights of zero, we can repeatedly update the weights in the direction of the gradient, on each iteration changing every weight w_i according to:

$$w_i \leftarrow w_i + \eta \sum_l X_i^l \Big(Y^l - \hat{P}(Y^l = 1 | X^l, W) \Big),$$

where η is a small constant for the step size.

Naive Bayes vs. Logistic Regression

Naive Bayes, a **generative classifier**, directly estimates parameters for $P(Y)$ and $P(X|Y)$, *i.e.,* it optimizes the joint data likelihood $P(X|Y)$ with respect to the conditional independence assumptions. **Logistic Regression**, a **discriminative classifier**, directly estimates the parameters of $P(Y|X)$, *i.e.,* the conditional data likelihood $P(Y$ given $X)$. Other differences are:

- Naive Bayes makes more restrictive assumptions and has higher asymptotic error, but converge faster than Logistic Regression ($\mathcal{O}(\ln n)$ vs. $\mathcal{O}(n)$).
- Naive Bayes is a learning algorithm with greater bias, but lower variance than Logistic Regression.
- Logistic Regression is consistent with the Naive Bayes assumption that the input features X_i are conditionally independent given Y. However, the data can disobey this assumption. The conditional likelihood maximization algorithm for Logistic Regression will adjust its parameters to maximize the fit to the data.

■

2. [3 pts] Imagine that a certain word is never observed in the training data, but occurs in a test instance. What will happen when our Naive Bayes classifier predicts the probability of the this test instance? Explain why this situation is undesirable. Will logistic regression have a similar problem? Why or why not?

Solution:

In Naive Bayes, if the training data does not contain a certain word, the maximum likelihood estimates defined above can result in θ estimates of zero. These features will be assigned zero probability, since they were not present in the training data but occur in the test data. However, zero probability should not be assigned to any event in the feature space. To prevent this, we employ Laplace smoothing. Furthermore, Logistic regression will not have the same problem because it is directly parametrized by the logit functions, not generated by the estimates parameters for $P(Y)$ and $P(X|Y)$.

Add-one smoothing is one way to avoid this problem with our Naive Bayes classifier. This technique pretends that every word occurs one additional time in the training data, which eliminates zero counts in the estimated parameters of the model. For a set of documents $C = x^1, ..., x^n$, the add-one smoothing parameter estimate is $\hat{\theta}_i = \frac{1 + \sum_{j=1}^n \text{count}_i(x^j)}{D+m}$, where D is the total number of words in C (i.e., $D = \sum_{i=1}^m \sum_{j=1}^n \text{count}_i(x^j)$). Empirically, add-one smoothing often improves classification performance when data counts are sparse.

3. [10 pts] Implement Logistic Regression and Naive Bayes. Use add-one smoothing when estimating the parameters of your Naive Bayes classifier. For logistic regression, we found that a step size around .0001 worked well. Train both models on the provided training data and predict the labels of the test data. Report the training and test error of both models.

Solution:

	Naive Bayes	Logistic Regression
Training Accuracy	0.98	1.00
Test Accuracy	0.75	0.70

4. [2 pts] Which model performs better on this task? Why do you think this is the case?

Solution:

Naive Bayes performed slightly better. Logistic Regression outperforms Naive Bayes when many training examples are available, but Naive Bayes outperforms Logistic Regression when training data is scarce (which is the case of this example). ■

CSE 512 Machine Learning: Homework 2

Department of Computer Science
Stony Brook University

- There are 4 questions on this assignment. The first question involves coding. Do *not* attach your code to the writeup. Instead, zip and submit your code electronically on Blackboard (Bb). Name your .zip file as **[your_SBU_name].zip**, e.g. sbillah.zip
- The assignment is due at 5:30 PM (beginning of class) on **Tuesday Mar 11, 2014**.
- Do not forget to put both your name and SBU ID on *each* page of your submission.
- If you have any questions, please direct your question first to the TA, then the instructor.

1 Boosting [45 points]

The details of Adaboost are in *Robert E. Schapire. The boosting approach to machine learning: An overview.* In Nonlinear Estimation and Classification. Springer, 2003. `http://www.cs.princeton.edu/~schapire/uncompress-papers.cgi/msri.ps`

1.1 [10 Points] Combination of Weak Classifiers

In this problem, we are going to show that any linear combination of classifiers can be good in maintaining a constant exponential loss on the input data.

Suppose we have a Boosting algorithm where $h_t \in H$ from $t = 1, \ldots, T$ be any sequence of classifiers. Let $\{x_i, y_i\}_{i=1}^m$ be a training set of m observations. Starting with $f_0 = 0$, f_t is defined as $f_t = f_{t-1} + \alpha_t h_t$ and $\alpha_t = \beta \log \frac{1-\epsilon_t}{\epsilon_t}$. Here, ϵ is the weighted training error of the classifier h_t which is defined as follows:

$$\epsilon_t = \sum_{i=1}^{m} D_{t-1}(i) I(y_i \neq h_t(x_i))$$

where $I(.)$ is an indicator function, $D_{t-1}(i)$ is the weight of data point i at $t-1$, and

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\frac{1}{\beta}\alpha_t y_i h_t(x_i)\right)}{Z_t}$$

Now, for all T, prove that :

$$\sum_{i=1}^{m} \frac{1}{m} \exp(-\frac{1}{\beta} y_i f_T(x_i)) = 1$$

1.2 [5 points] Loss Function

By this time, we know that the goal of boosting is to solve the following optimization problem:

$$\min_f \sum_{i=1}^{N} L(y_i, f(x_i))$$

and $L(y, \hat{y})$ is some loss function, and f is assumed to be an adaptive basis function model (ABM), which is a model of the form

$$f(x) = w_0 + \sum_{m=1}^{M} w_m \phi_m(x)$$

Common choices for the loss functions are given as follows: (i) squared error, $(y_i - f(x_i))^2$ (ii) absolute error, $|y_i - f(x_i)|$, (iii) exponential loss, $\exp(-(\tilde{y}_i f(x_i)))$, and (iv) logloss, $\log(1 + \exp(-\tilde{y}_i f(x_i)))$.

1. (2 point) Mention whether it is a good idea or a bad idea: for binary classification, absolute error or 0/1 loss is the perfect choice. Justify your answer.

2. (3 points) Consider the exponential error function E defined as follows:

$$E = \sum_{i=1}^{m} \exp(-(\tilde{y}_i f_T(x_i)))$$

where $f_T(x)$ is a classifier defined in terms of a linear combination of base classifiers $y_t(x)$ of the form

$$f_T(x) = \frac{1}{2} \sum_{t=1}^{T} \alpha_t y_t(x)$$

and $\tilde{y}_i \in \{-1, +1\}$ are the training set target values. Show that E, which is minimized by the AdaBoost algorithm, does not correspond to the log-likelihood of any well-behaved probabilistic model.
Hint: show that the corresponding conditional distribution $p(\tilde{y}|x)$ cannot be correctly normalized.

1.3 [5 Points] Adaboost on a Toy Dataset

Now we will apply Adaboost to classify a toy dataset. Consider the following dataset in Figure 1a). The dataset consists of 4 points , $(X_1 : 0, -1, -)$, $(X_2 : 1, 0, +)$, $(X_3 : -1, 0, +)$ and $(X_4 : 0, 1, -)$.

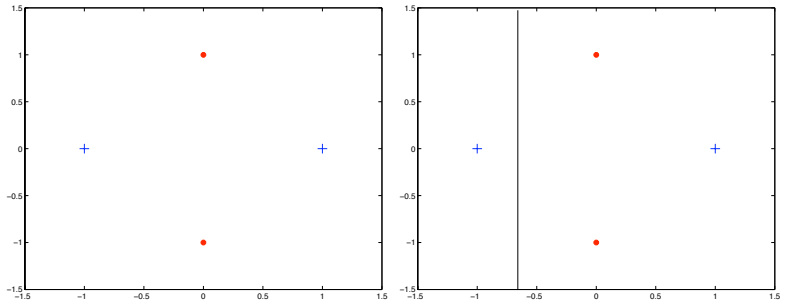


Figure 1: a) Toy data in Question 1. b) h_1 in Question 1

1. Use simple decision stumps as weak classifiers. (For description of decision stumps, refer to Problem 1.4) Now for $T = 4$, show how Adaboost works for this dataset. For each timestep remember to compute the following numbers:

$$\epsilon_t, \alpha_t, Z_t, D_t(i) \ \forall i,$$

Also for each timestep draw your weak classifier. For example h_1 can be as shown in 1b).

2. What is the training error of Adaboost?

3. Is the above dataset linearly separable? Explain why Adaboost does better than a decision stump in the above dataset.

1.4 [25 Points] Implementation

Implement the AdaBoost algorithm (page 658 in the Bishop's book) using a decision stump as the weak classifier.

AdaBoost trains a sequence of classifiers. Each classifier is trained on the same set of training data $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, m$, but with the significance $D_t(i)$ of each example $\{\mathbf{x}_i, y_i\}$ weighted differently. At each iteration, a classifier, $h_t(\mathbf{x}) \rightarrow \{-1, 1\}$, is trained to minimize the weighted classification error, $\sum_{i=1}^{m} D_t(i) \cdot I(h_t(\mathbf{x}_i) \neq y_i)$, where I is the indicator function (0 if the predicted and actual labels match, and 1 otherwise). The overall prediction of the AdaBoost algorithm is a linear combination of these classifiers, $H_T(\mathbf{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}))$. *Note:* The textbook uses $w_i \equiv D_t(i)$.

A decision stump is a decision tree with a single node. It corresponds to a single threshold in one of the features, and predicts the class for examples falling above and below the threshold respectively, $h_t(\mathbf{x}) = C_1 I(x^j \geq c) + C_2 I(x^j < c)$, where x^j is the j^{th} component of the feature vector $\mathbf{x}$. Unlike in class, where we split on Information Gain, for this algorithm split the data based on the weighted classification accuracy described above, and find the class assignments $C_1, C_2 \in \{-1, 1\}$, threshold c, and feature choice j that maximizes this accuracy.

1. *(15 points)* Submit your (zipped) code electronically on Bb. Name your file **[your_SBU_name].zip**, e.g. sbillah.zip. You do **not** need to include a hard copy of your code along with your HW submission.
2. Evaluate your AdaBoost implementation on the Bupa Liver Disorder dataset that is available for download from `http://www.cs.stonybrook.edu/~leman/courses/14CSE512/hws/hw2-data.tar.gz`. The classification problem is to predict whether an individual has a liver disorder (indicated by the selector feature) based on the results of a number of blood tests and levels of alcohol consumption. Use 90% of the dataset for training and 10% for testing. Average your results over 50 random splits of the data into training sets and test sets. Limit the number of boosting iterations to 100. In a single plot show:
   - average training error after each boosting iteration
   - average test error after each boosting iteration
3. *(5 points)* Using all of the data for training, display the selected feature component j, threshold c, and class label C_1 of the decision stump $h_t(\mathbf{x})$ used in each of the first 10 boosting iterations ($t = 1, 2, ..., 10$)
4. *(5 points)* Using all of the data for training, in a single plot, show the empirical cumulative distribution functions of the margins $y_i f_T(\mathbf{x}_i)$ after 10, 50 and 100 iterations respectively, where $f_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$. Notice that in this problem, before calculating $f_T(\mathbf{x})$, you should normalize the α_ts so that $\sum_{t=1}^{T} \alpha_t = 1$. This is to ensure that the margins are between -1 and 1.

   Hint: The empirical cumulative distribution function of a random variable X at x is the proportion of times $X \leq x$.

2 Model Selection and Cross-Validation [20 points]

2.1 [13 points] Bias-Variance Trade-off

Suppose, we define a term named True Risk as the MSE (Mean Squared Error) between predicted model and the true model as:

$$R(f) = E[(f(X) - Y)^2]$$

If we assume zero noise variance, it is shown that this risk can be expressed in terms of Bias-Variance Trade-off, i.e:

$$R(f) = E[(f(X) - Y)^2] = Variance + Bias^2$$

where $Y = f^*(X)$.

We can also define risk in terms of our estimated parameter $\hat{\theta}$ (obtained using MLE or MAP or density estimator, etc) and the true parameter θ as:

$$R(\theta, \hat{\theta}) = E_\theta[(\hat{\theta} - \theta)^2] = Var_\theta(\hat{\theta}) + bias^2$$

where $bias = E_\theta[\hat{\theta}] - \theta$

Let $X_1, \ldots, X_n \sim Bernoulli(p)$, be our coin-flip example where each X_i is an independent flip, $X_i = 1$ indicates flipping a head, and $X_i = 0$ indicates flipping a tail, with p as the probability of getting a head.

Consider two estimators for p: $\hat{p}_1 = \frac{1}{n}\sum_i X_i$ (the MLE estimate) and $\hat{p}_2 = \frac{\sum_i X_i + \alpha}{\alpha + \beta + n}$ (the mean of the posterior Beta distribution $P(p|D)$ when we use $Beta(\alpha, \beta)$ as prior).

1. (1 point) Compute the risk of $\hat{p}_1$, i.e. $R(p, \hat{p}_1)$
2. (4 points) Compute the risk of $\hat{p}_2$, i.e. $R(p, \hat{p}_2)$
3. (2 points) Which estimator $\hat{p}_1$ or $\hat{p}_2$ you would prefer when there is less data and which would you prefer when there is more data? (Hint: consider bias-variance tradeoff)
4. (3 points) Given a particular n, find the value of α and β that will make the risk of $\hat{p}_2$ constant.
5. (3 points) Using Hoeffding's inequality, and knowing that $P(0 \leq X_i \leq 1) = 1$, find an upper bound of $|\hat{p}_1 - p|$ with a probability of at least $1 - \gamma$.

2.2 [7 points] Model Selection

Let $x \in \{0, 1\}$ denote the result of a coin toss (x = 0 for tails, x = 1 for heads). The coin is potentially biased, so that heads occurs with probability θ_1. Suppose that someone else observes the coin flip and reports to you the outcome, y. But this person is unreliable and only reports the result correctly with probability θ_2; i.e., $p(y|x, \theta_2)$ is given by

	$y = 0$	$y = 1$
$x = 0$	θ_2	$1 - \theta_2$
$x = 1$	$1 - \theta_2$	θ_2

Assume that θ_2 is independent of θ_1 and x.

1. (1 point) Write down the joint probability distribution $p(x, y|\theta)$ as a 2×2 table, in terms of $\theta = (\theta_1, \theta_2)$. *Hint: write down how the likelihood function $p(x, y|\theta)$ factorizes.*

2. (1 point) Suppose we have the following dataset: $x = (1, 1, 0, 1, 1, 0, 0)$, $y = (1, 0, 0, 0, 1, 0, 1)$. What are the MLEs for θ_1 and θ_2? Justify your answer. What is $p(\mathcal{D}|\hat{\theta}, M_2)$ where M_2 denotes this 2-parameter model?

3. (1 point) Now consider a model with 4 parameters, $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$, representing $p(x, y|\theta) = \theta_{x,y}$. (Only 3 of these parameters are free to vary, since they must sum to one.)

x	y	θ
0	0	$\theta_{0,0} \equiv \theta_1$
0	1	$\theta_{0,1} \equiv \theta_2$
1	0	$\theta_{1,0} \equiv \theta_3$
1	1	$\theta_{1,1} \equiv \theta_4$

   What is the MLE of θ? What is $p(\mathcal{D}|\hat{\theta}, M_4)$ where M_4 denotes this 4-parameter model?

4. (4 points) Suppose we are not sure which model is correct. We compute the *Leave-One-Out Cross Validation* (LOOCV) log likelihood of the 2-parameter model M_2 and the 4-parameter model M_4 as follows:

$$L(M) = \sum_{i=1}^{m} \log p(x_i, y_i|M, \hat{\theta}(D_{-i}))$$

   and $\hat{\theta}(D_{-i})$ denotes the MLE computed on $\mathcal{D}$ excluding row i. Which model will LOOCV pick and why? *Hint: notice how the table of counts changes when you omit each training case one at a time.*

3 Neural Networks [15 points]

3.1 [15 points] Network Understanding

1. (2 point) Suppose my training data has lots of noise because it involves responses from human participants (and humans are "noisy" :-)). My Neural Network gave non-zero training error with 3 hidden layers, so I will try 4 hidden layers instead to reduce the error. Is this a good idea or bad idea? Justify/discuss your answer.

2. (2 points) When using back-propagation, it is important to choose a good range of random values for the weights. What problems arise if the weights are too small? What happens if they are too large?

3. (3 points) Consider the following neural network in Figure 2 in which the hidden unit nonlinear activation functions $g()$ are given by logistic sigmoid functions of the form $\sigma(a) = \frac{1}{1+\exp(-a)}$.

   Also recall that the overall network function takes the form

$$y_k(x, w) = \sigma(\sum_{j=1}^{M} w_{kj}^{(2)} g(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}) + w_{k0}^{(2)})$$

   Show that there exists an equivalent network, which computes exactly the same function, but with hidden unit activation functions given by $tanh(a)$ where the tanh function is defined as $tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$. *Hint: first find the relation between $\sigma(a)$ and $tanh(a)$, and then show that the parameters of the two networks can be obtained from one another by linear transformations.*

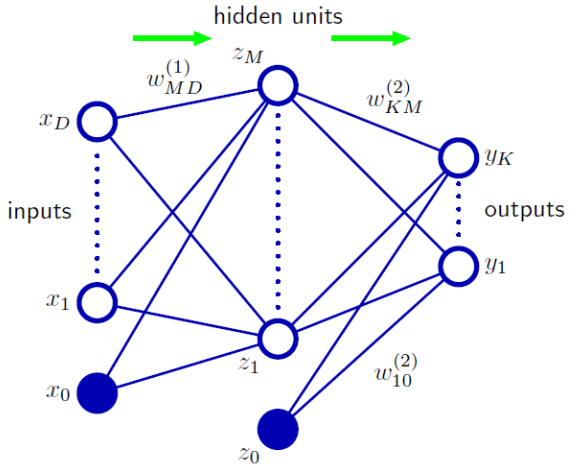


Figure 2: Diagram of a 2-layer neural network

4. (4 points) Given the above transformation, now show that a general linear combination of logistic sigmoid functions of the form

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_j \sigma\left(\frac{x - \mu_j}{s}\right)$$

is equivalent to a linear combination of *tanh* functions of the form

$$y(x, \mathbf{u}) = u_0 + \sum_{j=1}^{M} u_j tanh\left(\frac{x - \mu_j}{2s}\right)$$

and find expressions to relate the new parameters $\{u_1, \ldots, u_M\}$ to the original parameters $\{w_1, \ldots, w_M\}$.

5. (4 points) Suppose that you have two types of activation functions at hand:

$$g_I(x) = x,$$

$$g_s(x) = \begin{cases} 1 & if x \geq 0, \\ 0 & otherwise. \end{cases}$$

Now consider the indicator function $I_{[a,b)}(x)$:

$$I_{[a,b)}(x) = \begin{cases} 1 & if x \in [a, b); \\ 0 & otherwise. \end{cases}$$

Construct a neural network with one input x and one hidden layer whose response is $yI_{[a,b)}(x)$, for given real values y, a, and b; that is, its output is y if $x \in [a, b)$, and 0 otherwise. Draw the structure of the neural network, specify the activation function for each unit (either g_I or g_s), and specify the values for all weights (in terms of y, a, and b).

4 Instance-based Learning [20 points]

4.1 [10 Points] Kernel Regression

In this question we are given a set of n examples, (x_i, y_i) (where x_i is the input, and y_i is output), $i = 1, \ldots, n$, and $x_i \in R^d$ and $y_i \in R$.

Kernel regression is a widely used non-parametric regression method that applies a kernel function $K(.)$ to smooth the data y_i:

$$\hat{r}(x) = \frac{\sum_{i=1}^{n} K(x - x_i) y_i}{\sum_{i=1}^{n} K(x - x_i)} = \sum_{i=1}^{n} w_i(x, x_i) y_i$$

where $w_i(x, x_i) = \frac{K(x-x_i)}{\sum_{j=1}^{n} K(x-x_j)}$, and $\hat{r}(x)$ is the estimator of y (a scalar) at the point x (which is d-dimensional). There are many choices for the function $K(.)$; the most common one being the Gaussian kernel $K(x) = e^{-\|x\|^2/\sigma^2}$.

In the following questions we are going to look at 1-dimensional (d = 1) classification data with binary classes, i.e., $y_i \in \{0, 1\}$.

1. (2 points) Suppose that $x_i = i$ for all $i = 1, \ldots, n$, and to predict which class any new point $x_0 \in [0, n]$ belongs to, we use the decision rule

   $$\hat{y_0} = I(\hat{r}(x_0) > 0.5)$$

   where I is the indicator function that equals to 1 if the expression within I is true, and 0 if not true. Can you think of a kernel function K such that this decision rule gives the same answer as the k-nearest neighbor algorithm? (*Hint: the function $K(x)$ should be defined at some fixed range like $[a, b)$; you can ignore the case where x_0 is near the margins, i.e. close to 0 or n.*)

2. In general, if training data is drawn from some marginal distribution $p(x)$, we are unable to find a nice-looking kernel to simulate k-NN. One way to solve this problem is to use "locally weighted regression (LWR)" instead of kernel regression. The LWR is very similar to kernel regression except that we only calculate the weighted sum of k points near the given new point x:

   $$\hat{r}(x) = \sum_{i | x_i \in \text{k-NN of } x} w_i(x, x_i) y_i$$

   Now consider the case of weighted k-nearest neighbor (WKNN). Weighted k-NN penalizes the vote of every point z within the k-NN range by $d(x, z)$, the distance from x to z, meaning that having a point in class j in the k-neighbor will increase the vote for this class by $1/d(x, z)$.

   (a) (2.5 points) What should we put in for $w_i(x, x_i)$ so that the decision rule $\hat{y} = I(\hat{r}(x_0) > 0.5)$ gives the same answer as (unweighed) k-NN?.

   (b) (2.5 points) What type of k-NN does kernel regression with an arbitrary kernel function $K(x)$ simulate exactly? (Specify the value of k, whether the k-NN is weighted or non-weighted, and, if weighed, what the weights are.)

3. (3 points) How do you modify locally weighted regression method if we want to use it to simulate the weighted k-NN algorithm with more than 2 classes?

4.2 [10 Points] k-NN Classifier and Decision Surfaces

1. (**5 points**) **Decision Surfaces**

   Suppose, you have the following classifiers: (a) logistic regression, (b) Gaussian Naive Bayes, (c) 1-Nearest-Neighbor, and (d) 10-Nearest-Neighbor. Now, if you apply each of these classifiers on the following two datasets displayed in Figure 3, what would be the possible decision surface/boundaries. A hand-drawn decision boundary would be enough for each of the cases. But make sure that you briefly describe the important features of the boundary, e.g., (i) what made you choose that boundary, (ii) how good of a classifier it is, (iii) and anything else noteworthy.

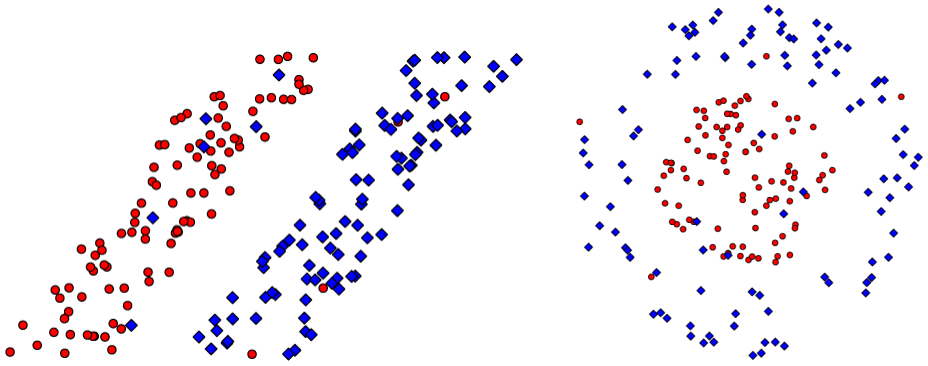


Figure 3: a) Toy data for Question 4.2. 1. b) Toy data for Question 4.2. 1

   Each image contains 200 points (2-dimensional), 100 from each of the two well-separated clusters; however, a few labels have been flipped due to noisy observation. For your convenience, these two images are made available in the document section of the Blackboard.

2. **[5 points] Limitation of k-NN** Consider n sample points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ independently and uniformly drawn from a p-dimensional zero-centered unit ball $B := \{\mathbf{x} \mid \sqrt{\mathbf{x}^\top \mathbf{x}} \leq 1, \mathbf{x} \in \mathcal{R}^p\}$. In this problem you will study the size of the 1-nearest neighborhood of the origin $\mathbf{0}$ and how it changes with respect to the dimension p, thereby gain intuition about the downside of k-NN in high dimension. More precisely, consider the distance from $\mathbf{0}$ to its nearest neighbor in the sample:

$$d^* := \min_{1 \leq i \leq n} \sqrt{\mathbf{x}_i^\top \mathbf{x}_i},$$

   which is a random variable since the sample is random.

   (a) (1.5 point) In the special case $p = 1$, what is the cumulative distribution function (cdf) of d^*, i.e., $P(d^* \leq t)$ for $0 \leq t \leq 1$?

   (b) (1.5 point) In the general case $p \in \{1, 2, 3, \ldots\}$, what is the cdf of d^*? (*Hint: You may find the following fact useful: the volume of a p-dimensional ball with radius r is* $\frac{(r\sqrt{\pi})^p}{\Gamma(p/2+1)}$, *where* $\Gamma(\cdot)$ *is the Gamma function.*)

   (c) (2 point) With the cdf you derived in Problem 2.2b, answer the following question: How large should the sample size n be such that with probability at least 0.9, the distance d^* from $\mathbf{0}$ to its nearest neighbor is less than 1/2, i.e., half way from $\mathbf{0}$ to the boundary of the ball? Your answer should be a function of p. From this function, what can you infer? Can you identify the downside of k-NN in terms of n and p?

CSE 512 Machine Learning: Homework 2

Marina von Steinkirch
SBU ID: 107185255

1 Boosting

1.1 Combination of Weak Classifiers

(Please check attached page in the end - page 1.)

■

1.2 Loss Function

The goal of boosting is to solve the following optimization problem:

$$\min_f \sum_{i=1}^{N} L(y_i, f(x_i)),$$

and $L(y, \hat{y})$ is some loss function, with f assumed to be an adaptative basis function model in the form

$$f(x) = w_0 + \sum_{m=1}^{M} w_m \phi_m(x).$$

1. Is it correct that for the binary classification, absolute error or 0/1 loss is the perfect choice?

   Solution:

   No. The 0/1 loss function is non-differentiable and non-convex. There is no guarantees about minimization with no-convex functions, *i.e.*, it is not guaranteed to find a global minimum. An upper differentiable and convex approximation of the 0/1 loss function is the exponential loss function.

   ■

2. Consider the exponential error function:

   $$E = \sum_{i=1}^{m} \exp\Big(- (\tilde{y}_i f_T(x_i))\Big),$$

   where $f_T(x)$ is a classifier denied in terms of a linear combination of base classifiers $y_t(x)$ of the form

   $$f_T(x) = \frac{1}{2} \sum_{t=1}^{T} \alpha_t y_t(x)$$

   and $\tilde{y}_i \in \{-1, +1\}$ are the training set target values. Show that E, which is minimized by the AdaBoost algorithm, does not correspond to the log-likelihood of any well-behaved probabilistic model.

Solution:

Adaboost does not have a simple interpretation in terms of the maximum likelihood. It minimizes empirical version of

$$E_{x,y} = P(y = +1|x)e^{-f(x)} + P(y = -1|x)e^{f(x)},$$

over all f, minimized when

$$f(x) = \frac{1}{2} \ln \frac{P(y = +1|x)}{P(y = -1|x)},$$

or

$$P(y = +1|x) = \frac{1}{1 + e^{-2f(x)}}.$$

One way to see this is that if we consider the maximum likelihood of logistic regression, it tries to minimize the sample average of

$$\phi(\alpha) = \ln(1 + e^{-2\alpha}),$$

which is close to Adaboost, that minimizes the sample average of

$$\phi(\alpha) = e^{-\alpha}.$$

We can prove this looking to the Taylor expansion around zero,

$$\ln(1 + e^{-2\alpha}) + 1 - \ln 2 \sim 1 - \alpha + \frac{\alpha^2}{2} ... = e^{-\alpha}.$$

The two functions are very similar around zero but asymptotically they are very different. While the logarithm will grows linearly, the exponential will grow exponentially (being non-normalized).

■

1.3 Adaboost on a Toy Model

(Please check attached page in the end - pages 2-3.)

■

1.4 Implement the AdaBoost algorithm.

AdaBoost trains a sequence of classifiers. Each classifier is trained on the same set of training data

$$(\mathbf{x}_i, y_i),\ i = 1, \ldots, m,$$

but with the significance $D_t(i)$ of each example

$$\{\mathbf{x}_i, y_i\}$$

weighted differently.

At each iteration, a classifier,

$$h_t(\mathbf{x}) \rightarrow \{-1, 1\},$$

is trained to minimize the weighted classification error,

$$\sum_{i=1}^{m} D_t(i) \cdot I(h_t(\mathbf{x}_i) \neq y_i),$$

where I is the indicator function (0 if the predicted and actual labels match, and 1 otherwise). The overall prediction of the AdaBoost algorithm is a linear combination of these classifiers,

$$H_T(\mathbf{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})).$$

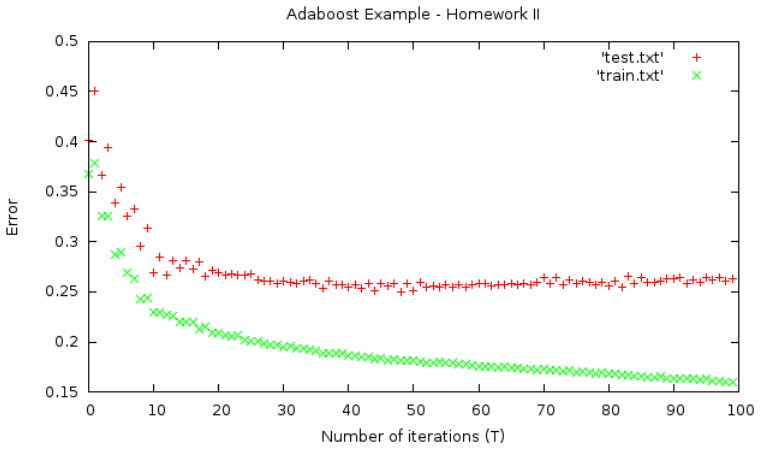
A decision stump is a decision tree with a single node. It corresponds to a single threshold in one of the features, and predicts the class for examples falling above and below the threshold respectively,

$$h_t(\mathbf{x}) = C_1 I(x^j \geq c) + C_2 I(x^j < c),$$

where x^j is the j^{th} component of the feature vector $\mathbf{x}$. Unlike in class, where we split on Information Gain, for this algorithm split the data based on the weighted classification accuracy described above, and find the class assignments $C_1, C_2 \in \{-1, 1\}$, threshold c, and feature choice j that maximizes this accuracy.

1. Submit your source code. `SUBMITTED!`

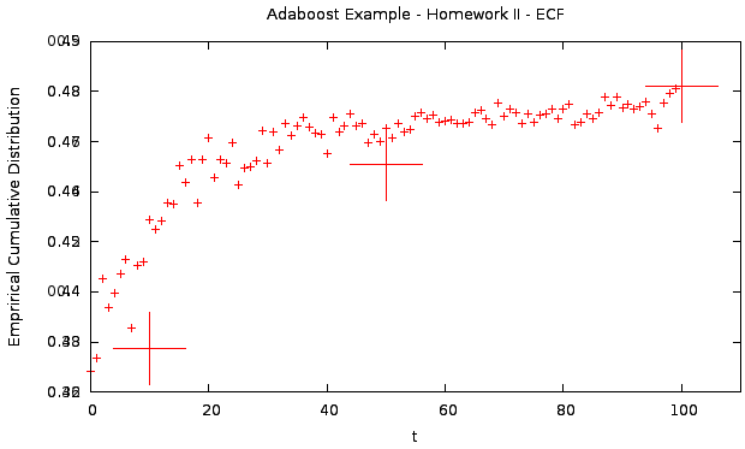
2. Evaluate your AdaBoost implementation on the Bupa Liver Disorder dataset. The classification problem is to predict whether an individual has a liver disorder (indicated by the selector feature) based on the results of a number of blood tests and levels of alcohol consumption. Use 90% of the dataset for training and 10% for testing. Average your results over 50 random splits of the data into training sets and test sets. Limit the number of boosting iterations to 100. In a single plot show:
   - average training error after each boosting iteration
   - average test error after each boosting iteration



3. Using all of the data for training, display the selected feature component j, threshold c, and class label C_1 of the decision stump $h_t(\mathbf{x})$ used in each of the first 10 boosting iterations ($t = 1, 2, ..., 10$)

t (iteration)	j (feature)	c (threshold)	C_1 (label)
1	4	20.5	1
2	2	26.5	1
3	3	19.5	1
4	1	59.5	-1
5	5	35.5	-1
6	5	3.5	-1
7	0	88.5	-1
8	4	12.5	1
9	2	19.5	-1
10	3	22.5	1

4. Using all of the data for training, in a single plot, show the empirical cumulative distribution functions of the margins $y_i f_T(\mathbf{x}_i)$ after 10, 50 and 100 iterations respectively, where $f_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$. Notice that in this problem, before calculating $f_T(\mathbf{x})$, you should normalize the α_ts so that $\sum_{t=1}^{T} \alpha_t = 1$. This is to ensure that the margins are between -1 and 1.



2 Neural Networks

2.1 Networking Understanding

1. Suppose my training data has lots of noise because it involves responses from human participants (and humans are noisy). My Neural Network gave non-zero training error with 3 hidden layers, so I will try 4 hidden layers instead to reduce the error. Is this a good idea or bad idea? Justify/discuss your answer.

   Solution:

   A network that is too complex that fits the noise, not just the signal, leads to **overfitting**, which is especially dangerous because it can lead to predictions that are far beyond the range of the training data. Therefore, adding another hidden layer is not the best strategy, because the noise will also be reproduced.

   The best approach is performing **cross verification**. Some of the training cases are reserved, and not actually used for training in the back propagation algorithm. Instead, they are used to keep an independent check. Usually, the initial performance of the network on training and verification sets is the same. As training progresses, the training error drops, minimizing the true error function, so the verification error drops too. However, if the verification error stops dropping, or starts to rise, this indicates that the network is starting to overfit the data, and training should cease (over-learning).

   ■

2. When using back-propagation, it is important to choose a good range of random values for the weights. What problems arise if the weights are too small? What happens if they are too large?

   Solution:

   If the weights are too small, the sigmoidal activation function will be approximately linear, which will lead to slow convergence in the network training. Setting the initial weights too large will result in poor fit or in oscillation between wrong values of weight, so that the network may take long time to learn.

   ■

3. Show that there exists an equivalent network, which computes exactly the same function, with hidden unit activation functions given by $\sigma(a) = \frac{1}{1+\exp(-a)}$ and $\sigma'(a)\tanh(a)$.

Solution:

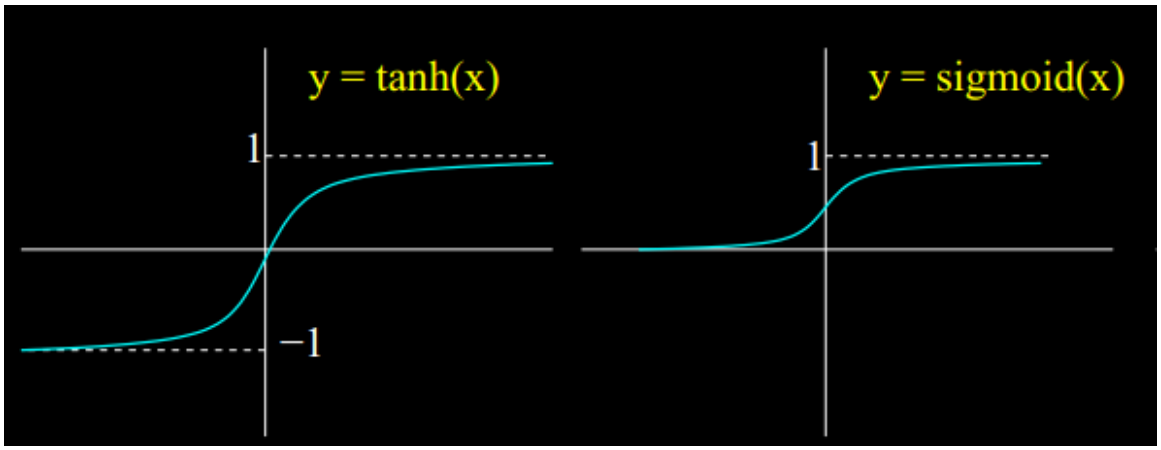
$$\begin{aligned}\tanh(b) &= \frac{e^b - e^{-b}}{e^b + e^{-b}} \\ &= \frac{e^{2b} - 1}{e^{2b} + 1} \\ &= \frac{1 - e^{-2b}}{1 + e^{-2b}} \\ &= \sigma(a = 2b) - \sigma(-a = -2b),\end{aligned}$$

but:

$$\begin{aligned}2 \times \sigma(a) - 1 &= \frac{2 - 1 - e^{-a}}{1 + e^{-1}} \\ &= \frac{1 - e^{-a}}{1 + e^{-a}}.\end{aligned}$$

Thus, the relation between then is given by (also illustrated in the figure below):

$$\boxed{\tanh(b) = 2\sigma(2b) - 1.}$$



■

4. Given the above transformation, show that a general linear combination of logistic sigmoid functions of the form

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_j \sigma\Big(\frac{x - \mu_j}{s}\Big)$$

is a combination linear of tanh functions of the form:

$$y(x, \mathbf{w}) = u_0 + \sum_{j=1}^{M} u_j \tanh\Big(\frac{x - \mu_j}{2s}\Big).$$

Solution:

Using $\frac{1+\tanh(a/2)}{2} = \sigma(a)$,

$$\begin{aligned} y(x, \mathbf{w}) &= w_0 + \sum_{j=1}^{M} w_j \sigma\Big(\frac{x-\mu_j}{s}\Big) \\ &= w_0 + \sum_{j=1}^{M} w_j \left(\frac{1+\tanh(\frac{x-\mu_j}{2s})}{2}\right) \\ &= w_0 + \sum_{j=1}^{M} \frac{w_j}{2} + \sum_{j=1}^{M} \frac{w_j}{2} \tanh\Big(\frac{x-\mu_j}{2s}\Big) \\ y(x, \mathbf{w}) &= u_0 + \sum_{j=1}^{M} u_j \tanh\Big(\frac{x-\mu_j}{2s}\Big) \end{aligned}$$

where

$$\boxed{u_0 = w_0 + \sum_{j=1}^{M} \frac{w_j}{2},}$$

and

$$\boxed{u_j = \frac{w_j}{2} \;\; j \in \{1, ..., M\}.}$$

■

5. Consider two types of activation function:

$$g_I(x) = x$$

and

$$g_s = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Consider the indicator function:

$$I_{[a,b)}(x) = \begin{cases} 1 & \text{if } x \geq [a, b) \\ 0 & \text{otherwise} \end{cases}$$

Construct a neural network with one input x and one hidden layer with response $yI_{[a,b)}(x)$e, for given values y, a and b.

Solution:

For the two activation functions, the general form of the output can be written as:

$$out(x) = g_I\Big(w_0 + \sum_i w_i g_s(w_0^{(i)} + w_1^{(i)} x)\Big).$$

Now, let us think about the three things that have to happen:

$$out(x) \to \begin{cases} 0 & \text{if } x < a \\ y & \text{if } x > a \text{ but } x < b \\ 0 & \text{if } x > b \end{cases}$$

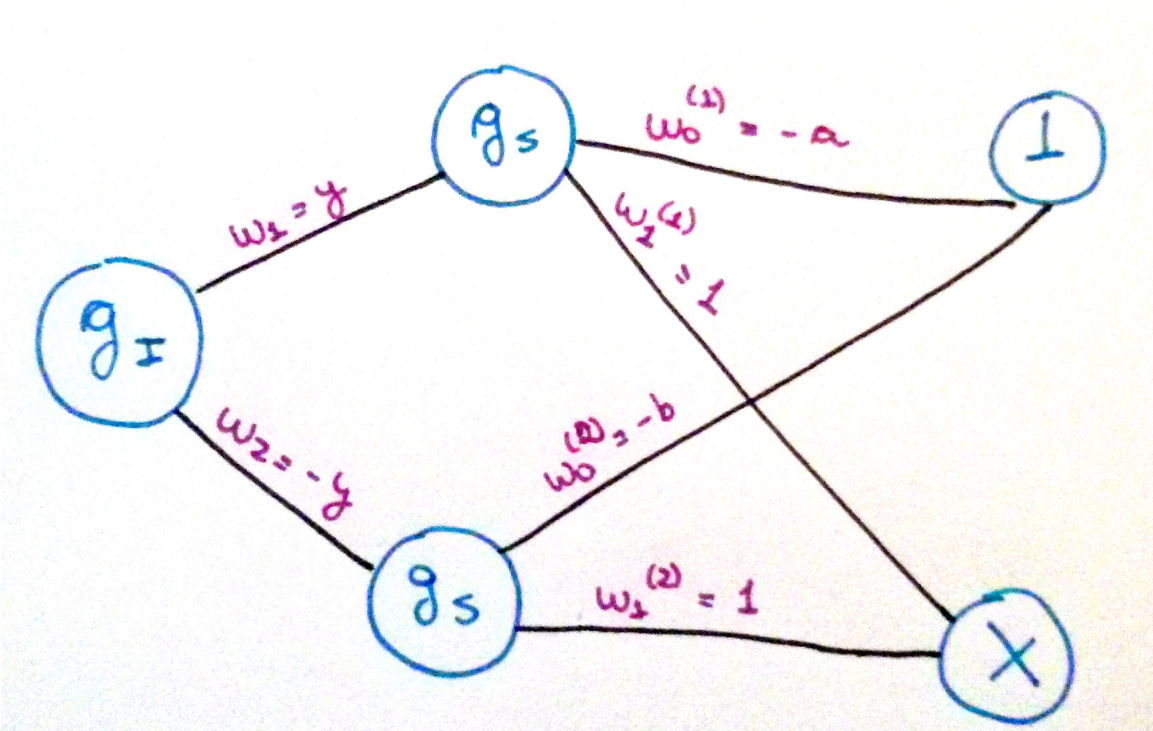
We can build this neural network with two step activation, one comparing x to a and one comparing x to b. Since the output has no constant independent of the constant y, $w_0 = 0$ in the first activation function. We set $w_0^{(1)} = -a$, $w_0^{(2)} = -b$, $w_1^{(1)} = 1$, $w_1^{(2)} = -1$ $w_1 = y$, and $w_2 = -y$. The result is:

$$\boxed{out(x) = g_I\Big(yg_s(x-a) - yg_s(x-b)\Big)}.$$

This is proved to be correct since:

$$out(x) \to \begin{cases} g_I\Big(yg_s(<0) - yg_s(<0)\Big) = 0 & \text{if } x < a \\ g_I\Big(yg_s(x>0) - yg_s(x<0)\Big) = y & \text{if } x > a \text{ but } x < b \\ g_I\Big(yg_s(x>0) - yg_s(x>0)\Big) = 1 - 1 = 0 & \text{if } x > b \end{cases}$$

$$\boxed{\tanh(b) = 2\sigma(2b) - 1.}$$



■

3 Model Selection

SOLUTION ATTACHED IN THE END (PAGES 4-7).

4 Instance-Based Learning

SOLUTION ATTACHED IN THE END (STARTING ON PAGE 8).

.1) COMBINATION OF WEAK CLASSIFIERS

- SUPPOSE WE HAVE A BOOSTING ALGORITHM WHERE $h_t \in H$ FROM $t=1-T$ ARE THE CLASSIFIERS.
- LET $\{x_i, y_i\}_{i=1}^m$ BE A TRAINING SET OF m OBSERVATIONS.
- STARTING WITH $f_0=0$, f_t IS DEFINED AS $\boxed{f_t = f_{t-1} + \alpha_t h_t}$ WHERE $\boxed{\alpha_t = \beta \log \frac{1-\epsilon_t}{\epsilon_t}}$
- THE WHEIGHTED TRAINING ERROR OF THE CLASSIFIER IS:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i)\, I(y_i \neq h_t(x_i))$$

↳ INDICATOR FUNCTION

↳ WEIGHT OF DATA POINT i AT $t-1$

AND:

$$D_{t+1} = \frac{D_t(i) \exp\left(-\frac{1}{\beta} \alpha_t y_i h_t(x_i)\right)}{Z_t}$$

- PROVE $\forall T$: $\sum_{i=1}^{m} \frac{1}{m} \exp\left(-\frac{1}{\beta} y_i f_T(x_i)\right) = 1$

FROM THE DEFINITION OF THE ADABOOST ALGORITHM, WE BEGIN WITH A SET OF WEIGHTS

$$D_0(i) = \frac{1}{m} \quad \text{(EQUALLY DISTRIBUTED)}$$

THE NEXT WEIGHTS WILL DEPENDING ON THE CHOICE OF THE WEAK CLASSIFIERS ITERATIVELY:

$$D_1(i) = \frac{\frac{1}{m} \cdot e^{-\alpha_1 \frac{1}{\beta} y_i h_1(x_i)}}{Z_1}$$

→ DATA WEIGHTS CAN BE CALCULATED RECURSIVELY!

$$D_2(i) = \frac{\frac{1}{m} \cdot e^{-\alpha_1 \frac{1}{\beta} y_i h_1(x_i)} \cdot e^{-\alpha_2 \frac{1}{\beta} y_i h_2(x_i)}}{Z_1 \cdot Z_2}$$

$$D_3(i) = \frac{\frac{1}{m} \cdot e^{-\alpha_1 \frac{1}{\beta} y_i h_1(x_i)} \cdot e^{-\alpha_2 \frac{1}{\beta} y_i h_2(x_i)} \cdot e^{-\alpha_3 \frac{1}{\beta} y_i h_3(x_i)}}{Z_1 \cdot Z_2 \cdot Z_3}$$

$\vdots$

$$D_T(i) = \frac{\frac{1}{m} e^{-\frac{1}{\beta} y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)}}{\prod_{t=1}^{T} Z_t} \longrightarrow \text{SINCE THE FACTORS ARE COMMUTATIVE}$$

WHERE THE (NEW) NORMALIZATION IS GIVEN BY:

$$Z_t = \sum_{i=1}^{m} D_{t-1}(i)\, e^{-\alpha_t \frac{1}{\beta} y_i h_t(x_i)}$$

AND HERE WE USED THE FACT THAT

$$\sum_{i=1}^{m} D_t(i) = 1$$

→ THE WEIGHTS MUST SUM TO 1!

SO THAT

$$\sum_{i=1}^{m} \frac{1}{m} e^{-\frac{1}{\beta} y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)} = \prod_{t=1}^{T} Z_t$$

Now we work on

$$\star \quad \sum_{i=1}^{m} \frac{1}{m} e^{-\frac{1}{\beta} y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)} = \prod_{t=1}^{T} Z_t \quad \star$$

Let's open Z_t:

$$Z_t = \sum_{i=1}^{m} D_{t-1}(i) \cdot e^{-\alpha_t \beta y_i h_t(x_i)}$$

$$= \sum_{i=1}^{m} D_{t-1}(i) \cdot e^{\alpha_t \beta \, \mathbb{I}\{y_i \neq h_t(x_i)\}} + \sum_{i=1}^{m} D_{t-1}(i) e^{-\alpha_t \beta} \{1 - \mathbb{I}\{y_i \neq h_t(x_i)\}\}$$

$$= e^{\alpha_t \beta} \sum_{i=1}^{m} D_{t-1}(i) \, \mathbb{I}\{y_i \neq h_t(x_i)\} + e^{-\alpha_t \beta} \sum_{i=1}^{m} D_{t-1}(i) \{1 - \mathbb{I}\{y_i \neq h_t(x_i)\}\}$$

Now, we use the coefficients $\alpha_t = \beta \log \frac{1-\epsilon_t}{\epsilon_t}$ and the fact that $b^{\log_b(y)} = y$:

$$\therefore = \frac{1-\epsilon_t}{\epsilon_t} \cdot \sum_{i=1}^{m} D_{t-1}(i) \, \mathbb{I}\{y_i \neq h_t(x_i)\} + \frac{\epsilon_t}{1-\epsilon_t} \sum_{i=1}^{m} D_{t-1}(i) \{1 - \mathbb{I}\{y_i \neq h_t(x_i)\}\}$$

Now, using the definition of the weighted training error: $\epsilon_t = \sum_{i=1}^{m} D_{t-1}(i) \, \mathbb{I}\{y_i \neq h_t(x_i)\}$

$$\therefore = \frac{1-\epsilon_t}{\epsilon_t} \cdot \epsilon_t + \frac{\epsilon_t}{1-\epsilon_t} (1 - \epsilon_t) \qquad \sum_{i=1}^{m} D_{t-1}(i) = 1$$

$$= 1 - \epsilon_t + \epsilon_t$$

$$= 1$$

QED.

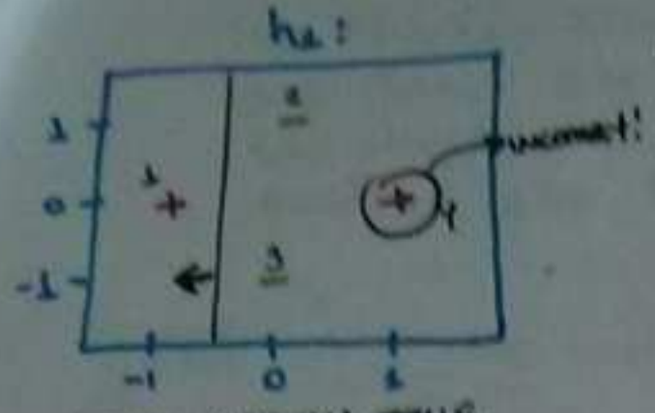
Now we just plug back to $\star$ to complete the proof

$$\sum_{i=1}^{m} \frac{1}{m} e^{-y_i \beta f_T(x_i)} = \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} 1 = 1$$

$\forall T$

■

1.3) ADABOOST ON A TOY DATASET



(THE ARROW TELLS WHICH SIDE WE CLASSIFY POSITIVE)

WE BEGIN WITH A SET OF WEIGHTS IN THE ALGORITHM SUCH THAT:

$$D_1(i) = \frac{1}{m} = \frac{1}{4}$$

THE WEIGHTED TRAINING ERROR IS $\epsilon_1 = 1/4$ SINCE ONE EXAMPLE IS MISCLASSIFIED.

WE THEN HAVE:

$$\alpha_1 = \frac{1}{2}\log\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = \frac{1}{2}\log\left(\frac{3/4}{1/4}\right) = \frac{\log 3}{2}$$

NOW, WE UPDATE THE D_2, RECALLING: $D_{t+1}(i) = \frac{D_t(i)\, e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$.

THEY ARE CORRECTLY CLASSIFIED SO $h_1(x_i) y_i = 1$

$$\begin{cases} D_2(1) = \dfrac{D_1(1)\cdot \exp(-\alpha_1 \times 1)}{Z_1} = \dfrac{\frac{1}{4}\exp(-1/2\log 3)}{Z_1} = \dfrac{1}{\sqrt{3}}\cdot\dfrac{1}{4}\cdot\dfrac{1}{Z_1} \\ D_2(2) = D_2(3) = \dfrac{D_1\cdot \exp(-\alpha_1 \times 1)}{Z_1} = \dfrac{1}{\sqrt{3}}\cdot\dfrac{1}{4}\cdot\dfrac{1}{Z_1} \end{cases}$$

$y_i h_1(x_i) = -1$ HENCE

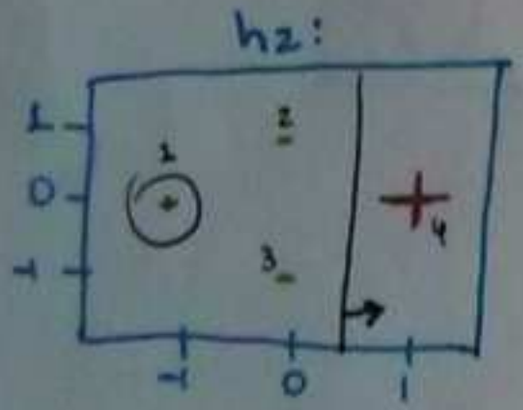
$$D_2(4) = \frac{D_1(4)\cdot \exp(-\alpha_1 \times (-1))}{Z_1} = \frac{\sqrt{3}}{4}\frac{1}{Z_1}$$

FINALLY, THE NORMALIZATION IS: GIVEN BY $Z_t = \sum_{i=1}^{m} D_t(i)\exp(-\alpha_t h_t^{(i)})$, SO THAT:

$$Z_1 = \frac{1}{\sqrt{3}}\cdot\frac{1}{4}\cdot 3 + \frac{1}{4}\sqrt{3} = \frac{\sqrt{3}}{2}$$

WHICH GIVES:

$$D_2(1) = D_2(2) = D_2(3) = 1/6 \quad \text{AND} \quad D_2(4) = 1/2$$



NOW, SINCE THE ONLY MISCLASSIFIED WAS x_4, WE HAVE

$$\epsilon_2 = 1/6$$

SO THAT: $\alpha_2 = \frac{1}{2}\log\left(\frac{1-1/6}{1/6}\right) = 1/2\log 5$

CALCULATING THE NEW D_3:

$y_i h_2(x_i) = -1$

$$D_3(1) = \frac{D_2(1)\cdot \exp(-\alpha_2 \times -1)}{Z_2} = \frac{1/6\cdot \exp(+1/2\log 5)}{Z_2} = \sqrt{5}\cdot\frac{1}{6}\cdot\frac{1}{Z_2}$$

$y_i h_2(x_i) = 1$

$$\begin{cases} D_3(2) = \dfrac{D_2(2)\cdot \exp(-\alpha_2 \times 1)}{Z_2} = \dfrac{1/6\cdot \exp(-1/2\log 5)}{Z_2} = \dfrac{1}{\sqrt{5}}\cdot\dfrac{1}{6}\cdot\dfrac{1}{Z_2} \\ D_3(3) = \dfrac{D_2(3)\cdot \exp(-\alpha_2 \times 1)}{Z_2} = \dfrac{1}{\sqrt{5}}\cdot\dfrac{1}{6}\cdot\dfrac{1}{Z_2} \\ D_3(4) = \dfrac{D_2(4)\cdot \exp(-\alpha_2 \times 1)}{Z_2} = 1/2\cdot 1/\sqrt{6}\cdot 1/Z_2 \end{cases}$$

THE NORMALIZATION CONSTANT IS:

$$Z_2 = 2 \times \frac{1}{6}\frac{1}{\sqrt{5}} + \frac{1}{2\sqrt{5}} + \frac{\sqrt{5}}{6} = \frac{\sqrt{5}}{3}$$

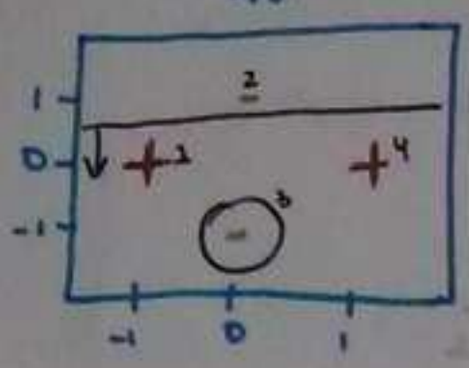
SO THAT:

$$D_3(1) = \frac{\sqrt{5}}{6}\left(\frac{\sqrt{5}}{3}\right)^{-1} = \frac{1}{2}$$

$$D_3(2) = D_3(3) = \frac{1}{\sqrt{5}}\frac{1}{6}\cdot\frac{3}{\sqrt{5}} = \frac{1}{10}$$

$$D_3(4) = \frac{1}{2}\frac{1}{\sqrt{5}}\cdot\frac{3}{\sqrt{5}} = \frac{3}{10}$$

h3:



$$\epsilon_3 = \frac{1}{10}$$

$$\alpha_3 = \frac{1}{2}\log\left(\frac{1-1/10}{1/10}\right) = \frac{1}{2}\log 9$$

bi h3(xi) = 1

$$\begin{cases} D_4(1) = \dfrac{D_3(1)\cdot \exp(-\alpha_3 \times 1)}{Z_3} = 1/2 \cdot \dfrac{\exp(-\frac{1}{2}\log 9)}{Z_3} = \frac{1}{2}\frac{1}{Z_3}\cdot\frac{1}{\sqrt{9}} = \frac{1}{6}\frac{1}{Z_3} \\ D_4(2) = \dfrac{D_3(2)\cdot \exp(-\alpha_3 \times 1)}{Z_3} = 1/10 \cdot \frac{1}{Z_3}\cdot\frac{1}{\sqrt{9}} = \frac{1}{30 Z_3} \end{cases}$$

bi h3(xi) = -1

$$\begin{cases} D_4(3) = \dfrac{D_3(3)\cdot \exp(-\alpha_3 \times (-1))}{Z_3} = 1/10 \cdot \frac{1}{Z_3}\cdot\sqrt{9} = \frac{3}{10 Z_3} \end{cases}$$

bi h3(xi) = 1

$$\begin{cases} D_4(4) = \dfrac{D_3(4)\cdot \exp(-\alpha_3 \times 1)}{Z_3} = 3/10 \cdot \frac{1}{Z_3}\cdot\frac{1}{\sqrt{9}} = \frac{1}{10 Z_3} \end{cases}$$

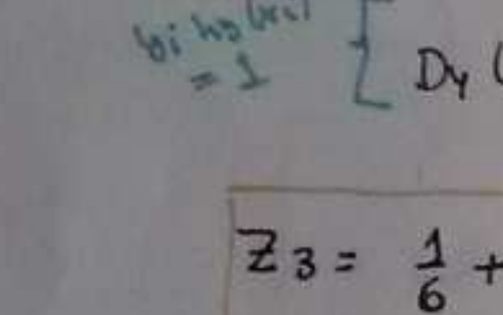
$$Z_3 = \frac{1}{6} + \frac{1}{30} + \frac{3}{10} + \frac{1}{10} = \frac{3}{5}$$

SO THAT:

$$D_4(1) = \frac{5}{3}\cdot\frac{1}{6} = \frac{5}{18} \qquad D_4(3) = \frac{3}{10}\cdot\frac{5}{3} = \frac{1}{2}$$

$$D_4(2) = \frac{1}{30}\cdot\frac{5}{3} = \frac{1}{18} \qquad D_4(4) = \frac{1}{10}\cdot\frac{5}{3} = \frac{1}{6}$$

h4:



$$\epsilon_4 = \frac{1}{18}$$

$$\alpha_4 = \frac{1}{2}\log\left(\frac{1-1/18}{1/18}\right) = \frac{1}{2}\log(17)$$

$$D_5(1) = \frac{D_4(1)}{Z_4}\cdot\frac{1}{\sqrt{17}} = \frac{5}{18}\frac{1}{\sqrt{17}}\cdot\frac{1}{Z_4} = \frac{5}{18}\frac{1}{\sqrt{17}}\cdot\frac{\sqrt{17}}{3} = \frac{5}{34}$$

$$D_5(2) = \frac{D_4(2)}{Z_4}\sqrt{17} = \frac{1}{18}\cdot\sqrt{17}\cdot\frac{1}{Z_4} = \frac{\sqrt{17}}{18}\cdot\frac{3}{\sqrt{17}} = \frac{1}{2}$$

$$D_5(3) = \frac{D_4(3)}{Z_4}\frac{1}{\sqrt{17}} = \frac{1}{2}\frac{1}{\sqrt{17}}\cdot\frac{1}{Z_4} = \frac{1}{2}\frac{1}{\sqrt{17}}\cdot\frac{3}{\sqrt{17}} = \frac{3}{34}$$

$$D_5(4) = \frac{D_4(4)}{Z_4}\frac{1}{17} = \frac{1}{6}\cdot\frac{1}{\sqrt{17}}\cdot\frac{1}{Z_4} = \frac{1}{6}\frac{1}{\sqrt{17}}\frac{3}{\sqrt{17}} = \frac{3}{34}$$

$$Z = \frac{5}{18}\frac{1}{\sqrt{17}} + \frac{1}{2}\frac{1}{\sqrt{17}} + \frac{1}{6}\frac{1}{\sqrt{17}} + \frac{\sqrt{17}}{18} = \frac{\sqrt{17}}{9}$$

THE TRAINING ERROR FOR ADABOOST

THE TRAINING ERROR OF THE FINAL CLASSIFIER IS BOUNDED BY $\prod_t Z_t$. SO, IF $Z_t < 1$, THE TRAIND ERROR DECREASES EXPONENTIALLY. IN OTHE WORDS, IF THE BOOSTING CAN MAKE THE UPPER BOUND → 0, THE TRAINING ERROR → 0, AND ADABOOST WILL ACHIEVE ZERO TRAINING ERROR EXPONENTIALLY FAST.

IN OUR CASE $\prod_t Z_t = Z_1 \cdot Z_2 \cdot Z_3 \cdot Z_4 \approx 0.18$, SO THE TRAINING ERROR IS 0.

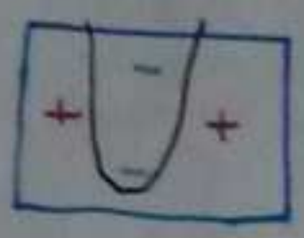
OPTIONAL: AN EXAMPLE WHERE DECISION STUMPS ARE NOT ENOUGH:

THIS EXAMPLE WILL NOT GET TRAINING ERROR ZERO:

BECAUSE ANY DECISION STUMP THAT YOU CHOOSE WILL ALWAYS GIVE ERROR OF 0.5, SO IT WILL BE ALWAYS 0, AND WE GET A TRIVIAL CLASSIFIER W/ ERROR 0.5.
(IN THIS CASE WE SHOULD CHANGE THE CLASSIFIER TO GENERAL LINEAR CLASSIFIER)

(3) IS THE ABOVE DATASE SEPARABLE?

NO, IT IS NOT. THIS DATA IS ONLY SEPARABLE IN HIGHER DIMENSIONAL SPACES (AND FOR THIS REASON BOOSTING IS EFFICIENT.



EXPLAIN WHY ADABOOST IS BETTER HERE THAN A DECISION STUMP:

FIRST, COMBINING PREDICTIONS OF MULTIPLE CLASSIFIERS IS MORE ACCURATE THAN A SINGLE CLASSIFIERS. THIS CAN HAPPEN FOR MANY REASONS: IT'S HARD TO FIND A SINGLE HIGHLY ACCURATE PREDICTION RULE (SPECIALLY TRUE FOR DECISION STUMPS), THE TRAINING DATASET IS SMALL AND THE HYPOTHESIS SPACE IS LARGE (WE ONLY HAVE 4 POINTS HERE), THE HYPOTHESIS SPACE DOES NOT CONTAIN THE TRUE FUNCTION (BUT IT'S SEVERAL GOOD APPROXIMATIONS), EXTENSIVE GLOBAL SEARCH IN THE HYPOTHESIS SPACE IS EXPENSIVE SO WE CAN COMBINE THE PREDICTIONS OF SEVERAL LOCALLY ACCURATE CLASSIFIERS.

DECISION STUMPS ARE VERY SIMPLE RULES (1-DEPTH DECISION TREES), SO THEY ARE TRULY WEAK. BOOSTING WITH DECISION STUMP SHOWS BETTER RESULTS WHEN COMPARED TO JUST DECISIONS TREES.

(2.1) The Bias-Variance Trade-off

Suppose we define a term named true risk as the MSE (mean squared error) between predicted model and the true model:

$$R(\hat{f}) = E\left[(\hat{f}(x) - y)^2\right]$$

If we assume zero noise variance, it is shown that this risk can be expressed in terms of bias-variance trade-off

$$R(\hat{f}) = E\left[(\hat{f}(x) - y)^2\right] = \text{variance} + \text{Bias}^2$$

where $y = f^*(x)$.

We can also define risk in terms of our estimated parameter $\hat{\theta}$:

$$\boxed{R(\theta, \hat{\theta}) = E_\theta\left[(\hat{\theta} - \theta)^2\right] = \text{Var}_\theta(\hat{\theta}) + \text{Bias}^2, \quad \text{Bias} = E_\theta[\hat{\theta}] - \theta}$$

(1) Compute the risk of $\hat{p}_1$:

Solution:

Using the last relation above:

$$R(p, \hat{p}_1) = \text{Var}[\hat{p}_1] + \text{Bias}[\hat{p}_1]^2$$

$$= \text{Var}\left[\frac{1}{N}\sum_i^N x_i\right] + \text{Bias}\left[\frac{1}{N}\sum_i^N x_i\right]^2$$

$$= \frac{1}{N}p(1-p) + \left[E\left[\frac{1}{N}\sum x_i\right] - p\right]^2$$

$$= \frac{1}{N}p(1-p) + (p-p)^2$$

$$= \boxed{\frac{p(1-p)}{n}} \quad \blacksquare$$

$\text{VAR}\left[\sum_i^N x_i\right] = np(1-p)$

$E\left[\frac{1}{n}\sum x_i\right] = p$

(2) Compute the risk of $\hat{p}_2$:

$$R(p, \hat{p}_2) = \text{VAR}[\hat{p}_2] + \text{BIAS}[\hat{p}_2]^2$$

$$= \text{VAR}\left[\frac{\sum_i^n x_i + 2}{2 + p + N}\right] + \text{BIAS}\left[\frac{\sum_i^n x_i + 2}{2 + p + N}\right]^2$$

$$= \frac{\text{VAR}\left[\sum_i^n x_i\right]}{(2+p+n)^2} + \left[E\left[\frac{\sum_i^n x_i + 2}{2+p+n}\right] - p\right]^2$$

$E\left[\sum_i^N x_i\right] = E\left[\sum_i^N x_i \cdot \frac{N}{N}\right]$

$= np$

$$= \frac{np(1-p)}{(2+p+n)^2} + \left(\frac{2 + E\left[\sum x_i\right]}{2+p+n} - p\right)^2$$

$$= \frac{np(1-p)}{(2+p+n)^2} + \left(\frac{2+np}{2+p+n} - p\right)^2$$

$$\therefore = \frac{mp(1-p)}{(\alpha+\beta+m)^2} + \frac{(\alpha + mp - p(\alpha+\beta+m))^2}{(\alpha+\beta+m)^2}$$

$$= \frac{mp(1-p) + (\alpha - p(\alpha+\beta))^2}{(\alpha+\beta+m)^2}$$ ■

(3) WHICH ESTIMATOR $\hat{p}_1$ OR $\hat{p}_2$ YOU PREFER WHEN THERE IS LESS DATA?

SOLUTION:

LESS DATA ⇒ THE MLE APPROACH IS WORSE THAN THE BAYESIAN APPROACH.
WE THEN CAN RELY ON THE PRIOR AND THIS WILL ALLOW LESS VARIANCE.
THEREFORE, WE USE $\hat{p}_2$.

MORE DATA ⇒ BOTH $\hat{p}_1$ AND $\hat{p}_2$ CAN BE USED SINCE:

$m \to \infty$:

$$R(p, \hat{p}_2) = \frac{mp(1-p) + (\alpha + p(\alpha+\beta))^2}{(\alpha+\beta+m)^2} \longrightarrow \frac{mp(1-p)}{m^2} \longrightarrow R(p, \hat{p}_1)$$

(THE PRIOR IS WASHED OUT!)

IF WE DON'T TRUST OUR PRIOR, WE MIGHT PREFER TO USE $\hat{p}_1$ (WHICH HAS SMALLER BIAS).

(4) GIVEN A PARTICULAR m, FIND α AND β TO MAKE THE RISK OF $\hat{p}_2$ CONSTANT:

SOLUTION:

LET'S WORK OUT THE SOLUTION FOR $\hat{p}_2$:

$$R(p, \hat{p}_2) = \frac{mp(1-p) + (\alpha - p(\alpha+\beta))^2}{(\alpha+\beta+m)^2}$$

$$= \frac{mp(1-p) + \alpha^2 + p^2(\alpha+\beta)^2 - 2\alpha p(\alpha+\beta)}{(\alpha+\beta+m)^2}$$

$$= \frac{p^2\left((-m + (\alpha+\beta)^2\right) + p\left(m - 2\alpha(\alpha+\beta)\right) + (\alpha^2)}{(\alpha+\beta+m)^2}$$

TO MAKE IT CONSTANT, WE NEED TO MAKE EVERY THING THAT DEPENDS ON p TO BE ZERO:

① $m = (\alpha+\beta)^2$

② $m = 2\alpha(\alpha+\beta)$

↓

(1) $\therefore \alpha + \beta = \sqrt{m}$

(2) $\therefore m = 2\alpha(\sqrt{m})$

$m^2 = 4\alpha^2 m$

$m^2 - 4\alpha^2 m = 0$

$m(m - 4\alpha^2) = 0$

$\therefore \boxed{\alpha = \frac{\sqrt{m}}{2}}$ or ~~$\alpha \neq 0$~~

$\therefore \boxed{\beta = \frac{\sqrt{m}}{2}}$ ■

(5) USING HOEFFDING'S INEQUALITY AND KNOWING $P(0 \leq X_i \leq 1) = 1$, FIND THE UPPER BOUND OF $|\hat{p}_1 - p|$ WITH PROBABILITY OF AT LEAST $1-\gamma$.

IN PROBABILITY THEORY, HOEFFDING'S INEQUALITY PROVIDES AN UPPER BOUND ON THE PROB. THAT THE SUM OF RANDOM VARIABLES DEVIATES FROM ITS EXPECTED VALUES

Caution:

(1) GIVEN $X_1, X_2, X_3, \dots X_N$ I.I.D. OBSERVATIONS SUCH THAT $E[X_i] = p$ AND $a \leq X_i \leq b$:

$$\boxed{P\left(\left|\frac{1}{N}\sum_i^N X_i - p\right| \geq \varepsilon\right) \leq 2\exp\left(-2N\varepsilon^2/(b-a)^2\right)}$$

FOR ANY $\varepsilon \geq 0$.

(2) FOR THE PROBABILITY OF AT LEAST $1-\gamma$:

$$P\left(\left|\frac{1}{N}\sum_i^N X_i - p\right| \geq \varepsilon\right) \leq 2\exp\left(-2N\varepsilon^2\right)$$

WHICH IS THE SAME OF:

$$P\left(\left|\frac{1}{N}\sum_i X_i - p\right| \leq \varepsilon\right) > 1 - 2e^{-2N\varepsilon^2}$$

THAT IS EXACTLY THE UPPER BOUND OF $|\hat{p}_1 - p|$

$$\therefore P\left(\left|\frac{1}{N}\sum_i X_i - p\right| \leq \varepsilon\right) > 1 - \gamma$$

$$\therefore P(|\hat{p}_1 - p| \leq \varepsilon) > 1 - \gamma$$

(3) NOW WE SEE THAT THE UPPER BOUND IS GIVEN BY:

$$2e^{-2N\varepsilon^2} \equiv \gamma$$

SO WE CAN WRITE:

$$\log\left(\frac{\gamma}{2}\right) = -2N\varepsilon^2$$

$$\therefore \boxed{\varepsilon = \sqrt{\frac{1}{2N}\log\frac{\gamma}{2}}}$$ ■

(2.2) Model Selection

LET $x \in \{0,1\}$, DENOTE THE RESULT OF A COIN TOSS ($x=0$ FOR TAILS, $x=1$ FOR HEADS) THE COIN IS BIASED SO HEADS OCCUR W/ PROB θ_1. SOMEONE ELSE REPORTS THE RESULT y WITH PROBABILITY θ_2:

$P(y|x,\theta_2)$ ⇒ GIVEN BY:

	$y=0$	$y=1$
$x=0$	θ_2	$1-\theta_2$
$x=1$	$1-\theta_2$	θ_2

(ASSUME θ_1 INDEPENDENT OF θ_2 AND x).

① WRITE DOWN THE JOINT PROBABILITY DIST. $P(x,y|\theta)$:

Solution:

$P(y=0|x=0,\theta_2)=\theta_2$
$P(y=0|x=1,\theta_2)=1-\theta_2$
$P(y=1|x=0,\theta_2)=1-\theta_2$
$P(y=1|x=1,\theta_2)=\theta_2$

$P(y=0|\theta_1)=1-\theta_1$
$P(y=1|\theta_1)=\theta_1$

Resulting:

	$y=0$	$y=1$
$x=0$	$\theta_2(1-\theta_1)$	$(1-\theta_2)(1-\theta_1)$
$x=1$	$(1-\theta_2)\cdot\theta_1$	$\theta_2\cdot\theta_1$

OBS: NOTE THAT THE PROBABILITY SUM TO 1.

② SUPPOSE WE HAVE $\begin{cases} x=(1,1,0,1,1,0,0) \\ y=(1,0,0,0,1,0,1) \end{cases}$ CALCULATE MLE FOR θ_1 AND θ_2.

Solution:

FROM THE RESULTS ABOVE, WE SEE THAT $\boxed{P(x,y|\theta)=P(y|x,\theta_2)\cdot P(x|\theta_1)}$, i.e. THE PARAMETERS CAN BE SEPARATED IN COMPONENTS.

THE $\theta=(\theta_1,\theta_2)$ THAT MINIMIZES THE LOG-LIKELIHOOD OF THE SAMPLES $\{(x_i,y_i), i=1,\dots m\}$ IS WRITTEN AS:

$$\begin{aligned} L(\theta_1,\theta_2) &= \sum_i \log P(x_i,y_i) \\ &= \sum_i (\log P(y_i|x_i) + \log P(x_i)) \\ &= \left(\sum_i \log P(y_i|x_i)\right) + \left(\sum_i \log P(x_i)\right) \\ &= L_2(\theta_2) + L_1(\theta_1) \end{aligned}$$

↓

SO WE CHOOSE TO MINIMIZE $L_1(\theta_1) = \sum_i \log P(x_i)$ FOR θ_1
AND $L_2(\theta_2) = \sum \log P(y_i|x_i)$ FOR θ_2.

① MINIMIZING L_1 FOR θ_1:

$$L_1(\theta_1) = \sum_i^n \log P(y_i)$$

$$= \left(\sum_i^n x_i\right) \log \theta_1 + \left(m - \sum_i^n x_i\right) \log (1-\theta_1)$$

DIFFERENTING:

$$\frac{\partial L_1}{\partial \theta_1} = \frac{\sum_i x_i}{\theta_1} - \frac{m - \sum_i x_i}{1-\theta_1}$$

SETTING TO ZERO:

$$\frac{\partial L_1}{\partial \theta_1} = 0 \Rightarrow \boxed{\hat{\theta}_1 = \frac{\sum_i^n x_i}{m}} \qquad (x = 1)$$

② MINIMIZING L_2 FOR θ_2:

$$L_2(\theta_2) = \sum_i^n \log P(y_i|x_i)$$

$$= \left[\sum_i^n x_i y_i + \sum_i (1-x_i)(1-y_i)\right] \theta_2 + \left(m - \left[\sum_i^n x_i y_i + \sum_i^n (1-x_i)(1-y_i)\right]\right)$$

$$= \bar{X}\,\theta_2 + (m - \bar{X})(1-\theta_2)$$

WHERE $\bar{X} = \sum_i^n (x_i y_i + (1-x_i)(1-y_i))$

DIFFERENTIATING AND SETTING TO ZERO:

$$\frac{\partial L_2}{\partial \theta_2} = 0 \Rightarrow \boxed{\hat{\theta}_2 = \frac{\sum_i^n (x_i y_i + (1-x_i)(1-y_i))}{m}} \qquad (x = y)$$

NOW, WE CALCULATE FOR THE DATA:

$$\hat{\theta}_1 = \frac{\sum_i^n x_i}{m} = \frac{4}{7}$$

$$\hat{\theta}_2 = \frac{\sum_i^n (x_i y_i + (1-x_i)(1-y_i))}{m} = \frac{4}{7}$$

THE MAXIMUM LIKELIHOOD OF THE DATA FOR THIS MODEL IS:

$$\prod_i^n P(y_i|x_i) P(x_i) = \left(\frac{4}{7}\right)^8 \left(\frac{3}{7}\right)^6$$

$$= 0.00007$$

(3) NOW CONSIDER A MODEL WITH 4 PARAMETERS $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$ REPRESENTING $p(x, y|\theta) = \theta_{x,y}$:

x	y	θ
0	0	$\theta_{0,0} = \theta_1$
0	1	$\theta_{0,1} = \theta_2$
1	0	$\theta_{1,0} = \theta_3$
1	1	$\theta_{1,1} = \theta_4$

WHAT'S THE MLE OF θ?

WHAT'S $P(D|\hat{\theta}, M_4)$?

SOLUTION:

SINCE ONLY 3 PARAMETERS ARE FREE TO VARY, WE REWRITE:

$$\theta_{0,0} = 1 - \sum_{i=1}^{3} \theta_i = \theta_1$$

$$\theta_{0,1} = \theta_2$$
$$\theta_{1,0} = \theta_3$$
$$\theta_{1,1} = \theta_4$$

FOR THE SAMPLES $\{(x_i, y_i), i = 1, \dots m\}$, WE CALCULATE THE LOG-LIKELIHOOD:

$$L(\theta) = \sum_i \log P(x_i, y_i)$$

$$= \underbrace{N_{1=0}}_{\#\text{times}=0} \cdot \log\left(1 - \sum_{i=1}^{3} \theta_i\right) + \sum_{i=1}^{3} \underbrace{N_i}_{\#\text{times}=i} \log \theta_i$$

DIFFERENTING EACH θ_i:

$$\frac{\partial L}{\partial \theta_i} = \frac{N_{i=0}}{1 - \sum_{i=1}^{3} \theta_i} + \frac{N_{i=i}}{\theta_i}$$

SETTING THE DERIVATIVES TO ZERO:

$$\frac{\partial L}{\partial \theta_i} = 0 \Rightarrow \theta_i = \frac{N_i}{N_{i=0} \Big/ 1 - \sum_{i=1}^{3} \theta_i}$$

$$= \frac{N_i}{N_{i=0}} \left[1 - \frac{1}{\sum_i \theta_i} (N_1 + N_2 + N_3)\right]$$

$$\therefore N = N_0 + N_1 + N_2 + N_3$$

SO WE GET THAT THE MLE OF EACH i IS $\hat{P}(i) = N_i / N$.
THE MLE OF THE JOINT IS $\hat{P}(x, y) = \frac{N_{x,y}}{N}$. → # (x, y) occurs

NOW, FOR OUR DATA:

$$\hat{P}_{x,y} = \begin{bmatrix} \hat{P}(0,0) & \hat{P}(0,1) \\ \hat{P}(1,0) & \hat{P}(1,1) \end{bmatrix} = \frac{1}{7} \begin{bmatrix} 2 & 1 \\ 2 & 2 \end{bmatrix}$$

↓

THE MLE UNDER THIS MODEL:

$$\prod_i \hat{P}(x_i, y_i) = \left(\frac{1}{7}\right)\left(\frac{2}{7}\right)^6 = 0.00008$$

WHICH IS LARGER THAN WITH 2 PARAMETERS !!!

(4) COMPUTE LOOCV LOG LIKELIHOOD OF M_2 AND M_4 AS:

$$L(M) = \sum_{i=1}^{7} \log p\,(x_i, y_i \mid M, \hat{\theta}(D_{-i}))$$

WHERE $\hat{\theta}(D_{-i})$ DENOTES MLE COMPUTED ON D EXCLUDING ROW i.
WHICH MODEL WILL LOOCV PICK?

SOLUTIONS:

* WE COMPUTE THE CROSS-VALIDATION LOG-LIKELIHOOD UNDER BOTH MODELS * AND CHOOSE THE ONE THAT IS LARGER.

2-PARAMETERS:

$$L = \log\left(\frac{1}{4}\cdot\frac{1}{6}\cdot\frac{1}{6}\cdot\frac{1}{6}\cdot\frac{1}{4}\cdot\frac{1}{3}\cdot\frac{4}{5}\right)$$

$$= \ln(0.000423)$$

$$= -10.06$$

3-PARAMETERS:

LET'S PICK A CASE WHERE ONLY HAPPENS ONCE. FOR OUR DATA WE NOTICE THAT $(x_7, y_7) = (0, 1)$ IS AN EXAMPLE.
WE CALCULATE $\hat{\theta}(D_{-1}) = \hat{\theta}(D_{-7}) = 0$, SO:

$$L = \log 0 = -\infty$$

THIS MODEL OVERFITS THE DATA!!!

SO WE PICK THE 2-PARAMETERS MODEL!

4.1 KERNEL REGRESSION

WE ARE GIVEN A SET OF n EXAMPLES (x_i, y_i) WHERE x_i IS THE INPUT AND y_i OUTPUT, $i = 1, \dots, n$ AND $x_i \in \mathbb{R}^d$ AND $y_i \in \mathbb{R}$.

KERNEL REGRESSION IS NON-PARAMETRIC REGRESSION METHOD THAT APPLIES A KERNEL FUNCTION $K(\cdot)$ TO SMOOTH THE DATA y_i:

$$\hat{r}(x) = \frac{\sum_{i=1}^{n} K(x - x_i)\, y_i}{\sum_{i=1}^{n} k(x - x_i)} = \sum_{i=1}^{n} w_i(x, x_i)\, y_i$$

WHERE $w_i(x, x_i) = \dfrac{K(x - x_i)}{\sum_{i=1}^{n} k(x - x_i)}$ AND $\hat{r}(x)$ IS THE ESTIMATOR OF y AT x.

WE LOOK TO 1-DIM. CLASSIFICATION W/ BINARY CLASSES.

(1) SUPPOSE $x_i = 1$, $i = 1, \dots, m$, TO PREDICT WHICH CLASS ANY POINT $x_0 \in [0, m]$, USING THE DECISION RULE:

$$\hat{y}_0 = I(\hat{r}(x_0) > 0.5)$$

KERNEL FUNCTION K TO GIVE THE SAME ANSWER AS KNN?

WE CAN USE $K(x, x_i) = I\left(-\frac{k}{2} \leq |x - x_i| < \frac{k}{2}\right)$ ■ ✓

(2) TRAINING DATA IS DRAWN FROM SOME MARGINAL DIST. $p(x)$. WE USE LWR:

$$\hat{r}(x) = \sum_{i \text{ in cell of } x} w_i(x, x_i)\, y$$

(a) WHAT SHOULD WE PUT FOR $w_i(x, x_i)$ SO $\hat{y} = I(\hat{r}(x_0) > 0.5)$ GIVES SAME ANSWER?

$$w_i(x, x_i) = 1/k$$ ■

(b) WHAT TYPE OF k-NN DOES K. R. W/ ARBITRARY FUNCTION $k(r)$ SIMULATE?

KERNEL REGRESSION HAS THE SAME SHAPE AS m-NN WEIGHTED REGRESSION WITH THE WEIGHT:

$$w_i(x, x_i) = \frac{d^{-1}(x, x_i)}{\sum_{j=1}^{N} d^{-1}(x, x_j)}$$ ■

explain?

(3) How do you modify locally weighted R. Method if want to use it to evaluate weighted k-NN Algorithms w/ more 2 classes?

We can do the following:

(1) $y \rightarrow$ replaced w/ a vector of dummy variables ($\in \{0,1\}$)

(2) We suppose $y \in \{0, 1, \ldots m-1\}$ and define

$$y^{(i)} = \langle I(y_i^{(1)} = 1), I(y_i^{(2)} = 1) \ldots I(y_i^{(m-1)} = 1) \rangle$$

$y^{(i)}$: factorized model; $y_i^{(1)}$: value of y_i

(3) Now we use kernel Regression on x:

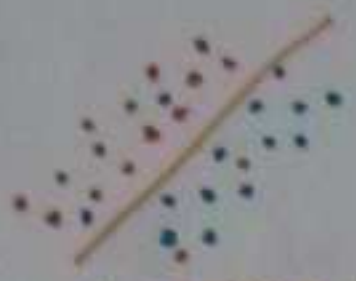
$\hat{\pi}(x)$ → values $\in [0,1]$

(4) Find the first element that < 0.5, assign the index to y

(5) Finally, if all elements in $\hat{\pi}(x) > 0.5$, assign 0 to y

(4.2) Decision Surfaces

(a) Logistic Regression:

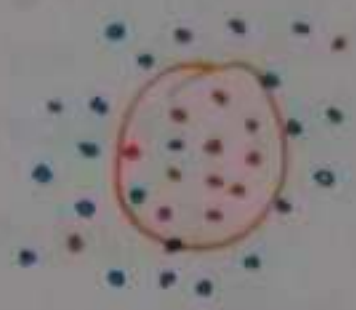


CLASSIFIES! ✓
(GOOD!)

CAN'T CLASSIFY BECAUSE THERE IS NO LINEAR DECISION BOUNDARY. ✗
(BAD!)

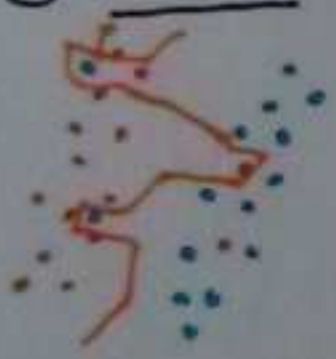
(b) Gaussian Naive Bayes:

NB ASSUMES UNCORRELATED FEATURES, AND HERE THIS ASSUMPTION DOESN'T WORK. ✗
(BAD!)



CLASSIFIES! ✓
(GOOD!)

(c) 1-NN:

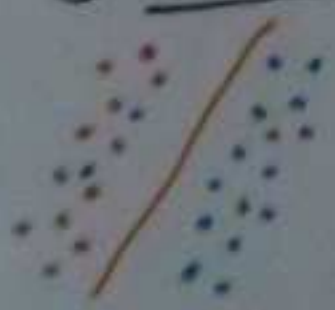


→ CHOOSE ONE POINT DISTANCE

FOR SMALLER K, THE PREDICTED LABEL IS MORE ACCURATE.

THE 1-NN ALGORITHM CAN CLASSIFY BY DISTANCE. A CONVEX POLYGON IS CREATED INDICATING THE REGION OF INSTANCE SPACE CLOSEST TO EACH POINT.

(d) 10-NN:



INCREMENTING K SIMPLIFIES THE DECISION BOUNDARY (MAJORITY MEANS LESS SENSITIVE TO INDIVIDUAL POINTS)

FOR LARGE K, THE PREDICTED LABEL IS MORE STABLE
(BETTER!!!)

K-MEANS LOOKS ALSO NICE ROUND CLUSTERS SO IT SHOULD CLASSIFY REALLY WELL. ✓

4.2.2 LIMITATION OF k-NN

CONSIDER m SAMPLE POINTS $\{x_1, x_2, \cdots x_m\}$, DRAWN FROM A p-DIMENSIONAL ZERO-CENTERED UNIT BALL $B = \{x \mid \sqrt{x^T x} \leq 1 \; x \in R^p\}$.

CONSIDER THE DISTANCE 0 TO ITS NN:

$$d^* = \min_{1 \leq i \leq m} \sqrt{x_i^T x_i}$$

(1) IN THE CASE $p = 1$, WHAT'S THE CDF OF d^*?

SOLUTION:

IN THIS CASE THE UNIT BALL IS $[-1, 1]$:

$$\therefore P(d^* \underset{\text{ONLY}}{\leq} t) = 1 - P(d^* \geq t)$$

$$= 1 - P(|x_i| \geq t), \; (i = 1, 2 \cdots, m)$$

$$= 1 - \prod_{i=1}^{m} P(|x_i| \geq t)$$

$$= 1 - (1-t)^N$$

(2) IN THE GENERAL CASE $p \in \{1, 2, 3 \cdots\}$ WHAT'S THE CDF OF d^*?

SOLUTION:

$$\therefore P(d^* \leq t) = 1 - P(d^* \geq t)$$ → USING THE GENERAL CASE DEFINITION

$$= 1 - P(\sqrt{x_i^T x_i} \geq t), \; (i = 1, 2, \cdots m)$$

$$= 1 - \prod_{i=1}^{N} P(\sqrt{x_i^T x_i} \geq t)$$

$$= 1 - \left(\frac{\sqrt{\pi}^p - (t\sqrt{\pi})^p}{\sqrt{\pi}^p} \right)^N$$ → VOLUME OF A p-DIMENSIONAL BALL W/ r IS $\frac{(r\sqrt{\pi})^p}{\Gamma(p/2+1)}$

$$= 1 - (1 - t^p)^N$$

(3) HOW LARGE SHOULD BE THE SAMPLE SIZE m TO HAVE PROB 0.9, d^* FROM $0 < 1/2$, HALF WAY FROM 0 TO THE BALL?
CAN YOU INFER? DOWNSIDES OF k-NN IN TERMS OF m AND p?

SOLUTION:

$\therefore$ WE WANT $P(d^* \leq 0.5) \geq 0.9$

CSE 512 Machine Learning: Homework 3

Department of Computer Science
Stony Brook University

- There are 3 questions on this assignment. The first question involves coding. Do *not* attach your code to the writeup. Instead, zip and submit your code electronically on Blackboard (Bb). Name your .zip file as **[your_SBU_name].zip**, e.g. sbillah.zip
- The assignment is due at 5:30 PM (beginning of class) on **April 8, 2014**.
- Do not forget to put both your name and SBU ID on *each* page of your submission.
- If you have any questions, please direct your question first to the TA, then the instructor.

1 k-NN, SVM, Classification and Cross-Validation [50 points]

In this question, you will explore how cross-validation can be used to fit the "magic parameters". More specifically, you will fit the constant k in the k-Nearest Neighbor algorithm, as well as the slack penalty C that appears in Support Vector Machines.

For all implementation questions, please electronically submit your source code through Blackboard, and supply pseudo-code in your writeup where requested.

1. (2 points) Download the file `hw3_matlab.zip` from

   `http://www.cs.stonybrook.edu/~leman/courses/14CSE512/hws/hw3_matlab.zip`

   and unpack it. The file `cvdataset.mat` contains the Matlab variables `traindata` (training data), `trainlabels` (training labels), `testdata` (test data), `testlabels` (test labels) and `evaldata` (evaluation data, needed later).

   This is a text classification task: given a document, you need to predict its topic. So, each row corresponds to a data point (a document). Each column is a feature, a word. The value of the feature is a relative frequency of the word in a document.

   The `cosineDistance.m` implements the *cosine distance*, a distance function commonly used for text data. It takes two feature vectors, and computes a nonnegative, symmetric distance between x and y. To check your data, compute and report the distance between the first training example from each class.

2. (6 points) Implement the k-Nearest Neighbor (kNN) algorithm in Matlab. Hand in pseudo-code. *Hint: You might want to pre-compute and store the distances between all pairs of points, to speed up the cross-validation later.*

3. (5 points) Implement n-fold cross validation for kNN. Your implementation should partition the training data and labels into n parts of approximately equal size. Hand in pseudo-code.

4. (8 points) Compute the 10-fold (i.e. $n = 10$) cross-validation *accuracy* for the training data, for $k = 1, 2, ..., 100$ and plot your results. Also plot the training and test *accuracy* for the same choices of k. How do you interpret these plots? Does the value of k which maximizes the cross-validation accuracy also maximizes the test set accuracy? Why or why not?

5. (6 points) Now download *libsvm* using the link:

   `http://www.cs.stonybrook.edu/~leman/courses/14CSE512/hws/libsvm-mat-2.84-1.zip`

and unpack it to your working directory. It has a Matlab interface which includes binaries for Windows. It can be used on OS X or Unix but has to be compiled—see the `README` file.

In your `hw3_matlab` folder find the files `testSVM.m` (an example demonstration script), `trainSVM.m` (for training) and `classifySVM.m` (for classification), which will show you how to use *libsvm* for training and classifying using an SVM. Run `testSVM`. This should report a test error of 0.3077. In order to train an SVM with slack penalty C on training set `data` with labels `labels`, call `svmModel = trainSVM(data, labels, C)`. In order to classify examples `test`, call `testLabels = classifySVM(svmModel, test)`. Train an SVM on the training data with $C = 4$, and report the error on the test set.

6. (5 points) Now implement n-fold cross-validation for SVMs. Similar to kNN, split your training data into n roughly equal parts. Hand in the pseudo-code.

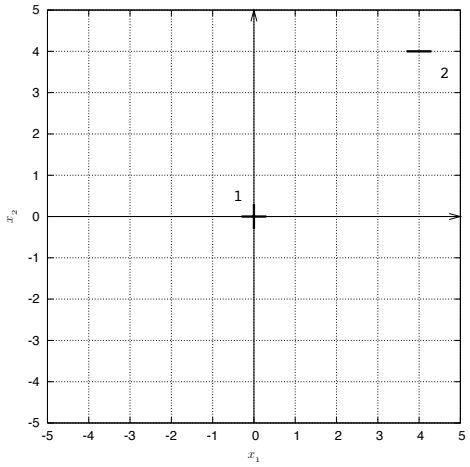
7. (8 points) Compute the 10-fold (i.e. $n = 10$) cross-validation *accuracy* for the training data, for $C = 1, 2, ..., 100$ and plot your results. Also plot the training and test *accuracy* for the same choices of C. How do you interpret these plots? Does the value of C which maximizes the cross-validation accuracy also maximizes the test set error? Why or why not?

8. (10 points) Design your favorite classifier: You have to use either k-NN or SVM, but you are allowed to use arbitrary values for k or for C. For kNN, you can invent different distance functions than the one we gave you or you can try to weigh the influence of training examples by their distance from the test point (*Hint: kernels*). If you want, you can do arbitrary feature selection, e.g. you can ignore columns. You can also perform any linear transformation of the features if you want. Whatever you do, please document it, and apply your algorithm to the `evaldata` data set. Output your class labels for this evaluation set, *one label per line*, in the order of the examples from the evaluation set. Submit your labels as file `evallabels_yourid.txt` where `yourid` is your SBU name, e.g. `sbillah`. Submit the actual code and the predicted labels (in file `evallabels_yourid.txt`) through Blackboard.

2 SVM and Kernels [25 points]

2.1 Working with Linear SVMs

Consider the following data set with one positive example $x_1 = (0, 0)$, $y_1 = +1$ and one negative example $x_2 = (4, 4)$, $y_2 = -1$.



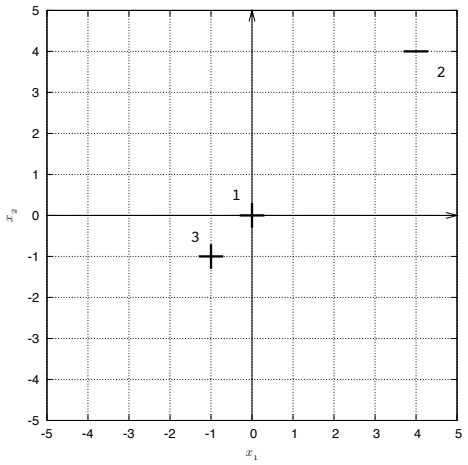
1. (1 points) Is the data linearly separable? Why, why not?

2. (3 points) If you use a linear SVM classifier for the given data, what would be the decision rule? Provide below.

$$h(\bar{x}) = \begin{cases} +1, & \text{if } \_\_\_\_\_\ x_1 + \_\_\_\_\_\ x_2 + \_\_\_\_\_\ \geq 0, \\ -1, & \text{otherwise.} \end{cases}$$

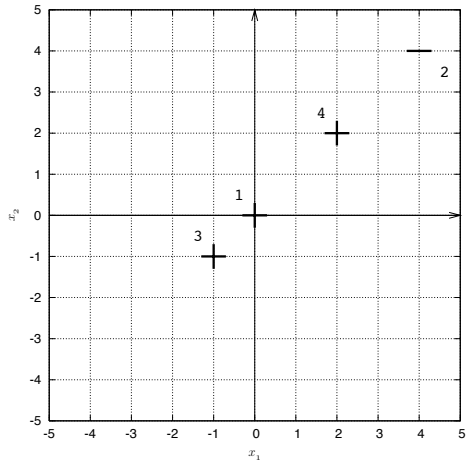
3. (2 points) Provide the weight value of the support vectors α_1 and α_2, as well as the offset-threshold value b of the SVM classifier.

Suppose we have an additional positive example $x_3 = (1, 1)$, $y_3 = +1$ as shown below.



4. (2 points) Which data points are support vectors? (Circle on the figure)
5. (2 points) Is the decision boundary for this data same as or different from for the data in previous part? Explain.
6. (2 points) Relative to the support vectors for the previous data, how do the weights (i.e. α values) of the support vectors for the data in this part change? Explain.

Next suppose we have yet another positive example $x_4 = (2, 2)$, $y_4 = +1$ as shown below.



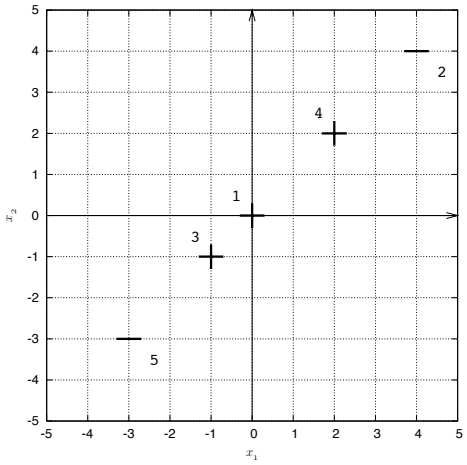
7. (2 points) Which data points are support vectors? (Circle on the figure)

8. (2 points) Is the decision boundary for this data same as or different from for the data in previous part? Explain.

9. (2 points) Relative to the support vectors for the previous data, how do the weights (i.e. α values) of the support vectors for the data in this part change? Explain.

2.2 Working with Kernel SVMs

Suppose we have an additional negative example $x_5 = (3, 3)$, $y_5 = 1$ as below.



1. (2 points) Circle the letter(s) to the left side of the kernels described in the list below that can separate the data.

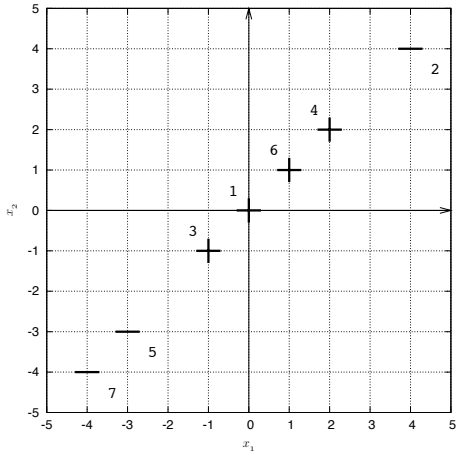
   (a) linear: $K(u, v) = u \cdot v$

   (b) polynomial of degree $n \geq 2$: $K(u, v) = (1 + u \cdot v)^n$

   (c) Radial Basis Function / Gaussian with sufficiently small scale parameter $\sigma : K(u, v) = e^{-\frac{||u-v||}{2\sigma^2}}$

   (d) none

Now suppose we have two additional examples: one positive example $x_6 = (+1, +1)$, $y_6 = +1$, and one negative example $x_7 = (4, 4)$, $y_7 = 1$ as given below.



Consider using the following kernel: $K(u,v) = 2\ ||u||\ ||v||$.

2. (3 points) Find the SVM classifier:

$$h(\bar{x}) = \begin{cases} +1, & \text{if } -2\ x_1^2 + \_\_\_\_\_\ x_1x_2 + \_\_\_\_\_\ x_2^2 + \_\_\_\_\_\ x_1 + \_\_\_\_\_\ x_2 + \_\_\_\_\_ \geq 0, \\ -1, & \text{otherwise.} \end{cases}$$

3. (2 points) Which data points are support vectors? (Circle on the figure)

Hint: It is possible to solve this problem without solving for the αs. Think of the definition of a kernel.

3 Learning Theory [25 points]

3.1 VC Dimension

In this section you will calculate the lower-bound for the VC-dimension of some hypothesis classes.

1. **[8 points]** Consider the hypothesis class of linear classifiers with offset in d dimensions:

$$\mathcal{H} = \{\text{sign}(\theta \cdot x + \theta_0) : \theta \in R^d, \theta_0 \in R\}$$

   Show that there exists a set of $d+1$ points $\{x_1, x_2, \ldots, x_{d+1}\}$ that can be shattered by $\mathcal{H}$. Specifically, first specify the points, and then given any labeling $y_1, y_2, \ldots, y_{d+1}$, describe explicitly how to construct a classifier in $\mathcal{H}$ that agrees with the labeling.

2. **[7 points]** Consider the hypothesis class of convex d-gons in the plane. A point is labeled positive if it is inside the d-gon. Demonstrate that there exists a set of $2d+1$ points on which any labeling can be shattered. *Hint:* You may think of data points on a circle.

3.2 Sample Complexity

In this part, you will use sample complexity bounds to determine how many training examples are needed to find a good classifier.

Let $\mathcal{H}$ be the hypothesis class of convex d-gons in the plane. In part 1, you showed that the VC dimension of d-gons in R^2 is at least $2d+1$. It can be shown that the upper bound is also $2d+1$.

Suppose we sample a number of m training examples i.i.d. according to some unknown distribution $\mathcal{D}$ over $R^2 \times \{+,-\}$.

3. **[10 points]** What is the least number of training examples $m > 1$ you need to have such that with probability at least 0.95 the convex **4-gon** separator in the plane with the smallest training error $\hat{h}_{ERM} = \arg\min_{h \in \mathcal{H}} \text{error}_{train}(h)$ has the following? Please show all your work.

$$\text{error}_{true}(\hat{h}_{ERM}) - \text{error}_{train}(\hat{h}_{ERM}) \leq 0.05$$

   Note that you may <u>not</u> assume $\text{error}_{train}(\hat{h}_{ERM})$. You may use any formulas from the lecture slides, textbook, or readings from the website, but please tell us where you found the formula(s) you use.