

# 10-701 Midterm Exam, Fall 2007

## 1. Personal info:

- Name:
- Andrew account:
- E-mail address:

2. There should be 17 numbered pages in this exam (including this cover sheet).
3. You can use any material you brought: any book, class notes, your print outs of class materials that are on the class website, including my annotated slides and relevant readings, and Andrew Moore's tutorials. You cannot use materials brought by other students. Calculators are not necessary. Laptops, PDAs, phones and Internet access are not allowed.
4. If you need more room to work out your answer to a question, use the back of the page and clearly mark on the front of the page if we are to look at what's on the back.
5. Work efficiently. Some questions are easier, some more difficult. Be sure to give yourself time to answer all of the easy ones, and avoid getting bogged down in the more difficult ones before you have answered the easier ones.
6. Note there are extra-credit sub-questions. The grade curve will be made without considering students' extra credit points. The extra credit will then be used to try to bump your grade up without affecting anyone else's grade.
7. You have 80 minutes.
8. Good luck!

Question	Topic	Max. score	Score
1	Short questions	$20 + 0.1010$ extra	
2	Loss Functions	12	
3	Kernel Regression	12	
4	Model Selection	14	
5	Support Vector Machine	12	
6	Decision Trees and Ensemble Methods	30	

# 1 [20 Points] Short Questions

The following short questions should be answered with at most two sentences, and/or a picture. For yes/no questions, make sure to provide a *short* justification.

1. [2 point] Does a 2-class Gaussian Naive Bayes classifier with parameters  $\mu_{1k}, \sigma_{1k}, \mu_{2k}, \sigma_{2k}$  for attributes  $k = 1, \dots, m$  have exactly the same representational power as logistic regression (i.e., a linear decision boundary), given no assumptions about the variance values  $\sigma_{ik}^2$ ?

★ **SOLUTION:** No. Gaussian Naive Bayes classifier has more expressive power than a linear classifier **if we don't restrict the variance parameters to be class-independent**. In other words, Gaussian Naive Bayes classifier is a linear classifier if we assume  $\sigma_{1k} = \sigma_{2k}, \forall k$ .

2. [2 points] For linearly separable data, can a small slack penalty (“ $C$ ”) hurt the training accuracy when using a linear SVM (no kernel)? If so, explain how. If not, why not?

★ **SOLUTION:** Yes. If the optimal values of  $\alpha$ 's (say in the dual formulation) are greater than  $C$ , we may end up with a sub-optimal decision boundary with respect to the training examples. Alternatively, **a small  $C$  can allow large slacks**, thus the resulting classifier will **have a small value of  $w^2$**  but **can have non-zero training error**.

3. [3 points] Consider running AdaBoost with Multinomial Naive Bayes as the weak learner for two classes and  $k$  binary features. After  $t$  iterations, of AdaBoost, how many parameters do you need to remember? In other words, how many numbers do you need to keep around to predict the label of a new example? Assume that the weak-learner training error is non-zero at iteration  $t$ . Don't forget to mention where the parameters come from.

★ **SOLUTION:** Recall that we predict the label of an example using AdaBoost by  $y = \text{sign}(\sum_t \alpha_t h_t(x))$ . Hence for each iteration of adaboost we need to remember the parameters of  $h_t(x)$  and  $\alpha_t$ . Since our weak classifier is Multinomial Naive Bayes, we need  $2k + 1$  parameters for it including  $2k$  for  $P(X_i|Y = 0), P(X_i|Y = 1)$  for  $k = 1, \dots, k$  and the prior  $P(Y = 1)$ . Hence for each iteration we have  $2k + 2$  parameters including  $\alpha_t$ , and after  $t$  iterations we have  $t(2k + 2)$  parameters.

4. [2 points] In boosting, would you stop the iteration if the following happens? Justify your answer with at most two sentences each question.

- The error rate of the combined classifier on the original training data is 0.

★ **SOLUTION:** No. Boosting is robust to overfitting. Test error may decrease even after the training error is 0.

- The error rate of the current weak classifier on the weighted training data is 0.

★ **SOLUTION:** Yes. In this case,  $\alpha_t = +\infty$ , and the weights of all the examples are 0. On the other hand, the current weak classifier is perfect on the weighted training data, so it is also perfect on the original data set. There is no need to combine this classifier with other classifiers.

5. [4 points] Given  $n$  linearly independent feature vectors in  $n$  dimensions, show that for any assignment to the binary labels you can always construct a linear classifier with weight vector  $w$  which separates the points. Assume that the classifier has the form  $\text{sign}(w \cdot x)$ . Note that a square matrix composed of linearly independent rows is invertible.

★ **SOLUTION:** Lets define the class labels  $y \in \{-1, +1\}$  and the matrix  $X$  such that each of the  $n$  rows is one of the  $n$  dimensional feature vectors. Then we want to find a  $w$  such that  $\text{sign}(Xw) = y$ . We know that if  $Xw = y$  then  $\text{sign}(Xw) = y$ . Because,  $X$  is composed of linearly independent rows we can invert  $X$  to obtain  $w = X^{-1}y$ . Therefore we can construct a linear classifier that separates all  $n$  points. Interestingly, if we add an additional constant term to the features we can separate  $n + 1$  linearly independent point in  $n$  dimensions.

6. [3 points] Construct a one dimensional classification dataset for which the Leave-one-out cross validation error of the One Nearest Neighbors algorithm is always 1. Stated another way, the One Nearest Neighbor algorithm never correctly predicts the held out point.

★ **SOLUTION:** For this question you simply need an alternating configuration of the points  $\{+, -, +, -, \dots\}$  along the real line. In leave-one-out cross validation we compute the predicted class for each point given all the remaining points. Because the neighbors of every point are in the opposite class, the leave-one-out cross validation predictions will always be wrong.

7. [2 points] Would we expect that running AdaBoost using the ID3 decision tree learning algorithm (without pruning) as the weak learning algorithm would have a better true error rate than running ID3 alone (i.e., without boosting (also without pruning))? Explain.

★ **SOLUTION:** No. Unless two differently labeled examples have the same feature vectors, ID3 will find a consistent classifier every time. In particular, after the first iteration of AdaBoost,  $\epsilon_1 = 0$ , so the first decision tree learned gets an infinite weight  $\alpha_1 = \infty$ , and the example weights  $D_{t+1}(i)$  would either all become 0, all become  $\infty$ , or would remain uniform (depending on the implementation). In any case, we either halt, overflow, or make no progress, none of which helps the true error rate.

8. [1 point] Suppose there is a coin with unknown bias  $p$ . Does there exist some value of  $p$  for which we would expect the maximum a-posteriori estimate of  $p$ , using a  $Beta(4, 2)$  prior, to require more coin flips before it is close to the true value of  $p$ , compared to the number of flips required of the maximum likelihood estimate of  $p$ ? Explain. (The  $Beta(4, 2)$  distribution is given in the figure below.)

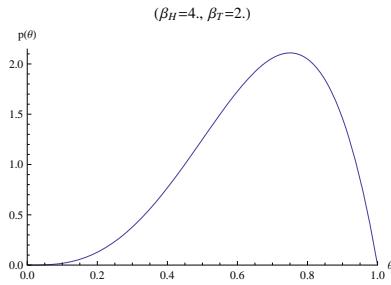


Figure 1:  $Beta(4, 2)$  distribution

★ **SOLUTION:** Yes. Consider  $p = 0$ . Because the prior value there is low, it takes more samples to overcome our prior beliefs and converge to the correct solution, compared to maximum likelihood (which only needs a single observation to find  $p$  exactly).

9. [1 point] Suppose there is a coin with unknown bias  $p$ . Does there exist some value of  $p$  for which we would expect the maximum a-posteriori estimate of  $p$ , using a  $Uniform([0, 1])$  prior, to require more coin flips before it is close to the true value of  $p$ , compared to the number of flips required of the maximum likelihood estimate of  $p$ ? Explain.

★ **SOLUTION:** No. In this case, the prior's PDF is a constant  $f_p(q) = 1$ , so the MAP solution  $\arg \max_{q \in [0,1]} f_p(q) f_x(\text{Data}; q)$  is always identical to the maximum likelihood solution  $\arg \max_{q \in [0,1]} f_x(\text{Data}; q)$ .

10. [0.1010 extra credit] Can a linear classifier separate the positive from the negative examples in the dataset below? Justify.

*Colbert  
for  
president*

*U2  
Loosing my religion*

*The Beatles  
There is a season...  
Turn! Turn! Turn!*

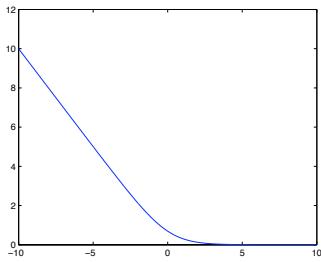
*Nirvana  
Grunge*

★ **SOLUTION:** No. This is an instantiation of XOR.

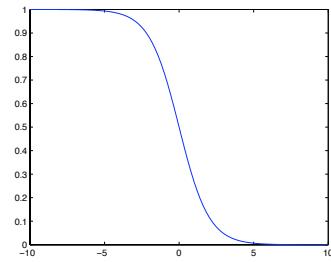
## 2 [12 points] Loss Function

Generally speaking, a classifier can be written as  $H(x) = \text{sign}(F(x))$ , where  $H(x) : \mathbb{R}^d \rightarrow \{-1, 1\}$  and  $F(x) : \mathbb{R}^d \rightarrow \mathbb{R}$ . To obtain the parameters in  $F(x)$ , we need to minimize the loss function averaged over the training set:  $\sum_i L(y^i F(x^i))$ . Here  $L$  is a function of  $yF(x)$ . For example, for linear classifiers,  $F(x) = w_0 + \sum_{j=1}^d w_j x_j$ , and  $yF(x) = y(w_0 + \sum_{j=1}^d w_j x_j)$

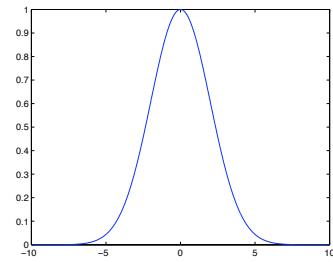
1. [4 points] Which loss functions below are appropriate to use in classification? For the ones that are not appropriate, explain why not. In general, what conditions does  $L$  have to satisfy in order to be an appropriate loss function? The x axis is  $yF(x)$ , and the y axis is  $L(yF(x))$ .



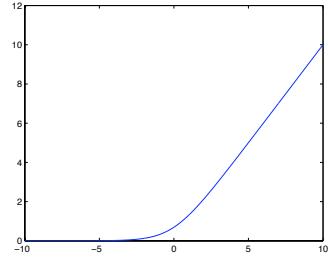
(a)



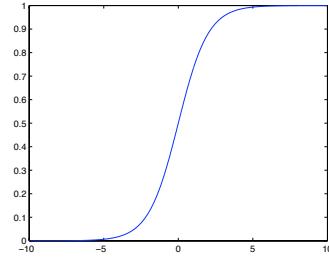
(b)



(c)



(d)



(e)

★ **SOLUTION:** (a) and (b) are appropriate to use in classification. In (c), there is very little penalty for extremely misclassified examples, which correspond to very negative  $yF(x)$ . In (d) and (e), correctly classified examples are penalized, whereas misclassified examples are not. In general,  $L$  should approximate the 0-1 loss, and it should be a non-increasing function of  $yF(x)$ .

2. [4 points] Of the above loss functions appropriate to use in classification, which one is the most robust to outliers? Justify your answer.

★ **SOLUTION:** (b) is more robust to outliers. For outliers,  $yF(x)$  is often very negative. In (a), outliers are heavily penalized. So the resulting classifier is largely affected by the outliers. On the other hand, in (b), the loss of outliers is bounded. So the resulting classifier is less affected by the outliers, and thus more robust.

3. [4 points] Let  $F(x) = w_0 + \sum_{j=1}^d w_j x_j$  and  $L(yF(x)) = \frac{1}{1+\exp(yF(x))}$ . Suppose you use gradient descent to obtain the optimal parameters  $w_0$  and  $w_j$ . Give the update rules for these parameters.

★ **SOLUTION:** To obtain the parameters in  $F(x)$ , we need to minimize  $\sum_i L(y^i F(x^i)) = \sum_i \frac{1}{1+\exp(y^i F(x^i))} = \sum_i \frac{1}{1+\exp(y^i(w_0 + \sum_{j=1}^d w_j x_j^i))}$ .

$$\frac{\partial}{\partial w_0} \sum_i L(y^i F(x^i)) = \sum_i \frac{\partial}{\partial w_0} \left( \frac{1}{1 + \exp(y^i(w_0 + \sum_{j=1}^d w_j x_j^i))} \right) = - \sum_i \frac{y^i \exp(y^i F(x^i))}{(1 + \exp(y^i F(x^i)))^2}$$

$$\forall k = 1, \dots, d,$$

$$\frac{\partial}{\partial w_k} \sum_i L(y^i F(x^i)) = \sum_i \frac{\partial}{\partial w_k} \left( \frac{1}{1 + \exp(y^i(w_0 + \sum_{j=1}^d w_j x_j^i))} \right) = - \sum_i \frac{y^i x_k^i \exp(y^i F(x^i))}{(1 + \exp(y^i F(x^i)))^2}$$

Therefore, the update rules are as follows.

$$w_0^{(t+1)} = w_0^t - \eta \frac{\partial}{\partial w_0} \sum_i L(y^i F(x^i)) = w_0^t + \eta \sum_i \frac{y^i \exp(y^i F(x^i))}{(1 + \exp(y^i F(x^i)))^2}$$

$$\forall k = 1, \dots, d,$$

$$w_k^{(t+1)} = w_k^t - \eta \frac{\partial}{\partial w_k} \sum_i L(y^i F(x^i)) = w_k^t + \eta \sum_i \frac{y^i x_k^i \exp(y^i F(x^i))}{(1 + \exp(y^i F(x^i)))^2}$$

■ **COMMON MISTAKE 1:** Instead of minimizing  $\sum_i L(y^i F(x^i))$ , some people only minimize  $L(yF(x))$ .

### 3 [12 points] Kernel Regression, $k$ -NN

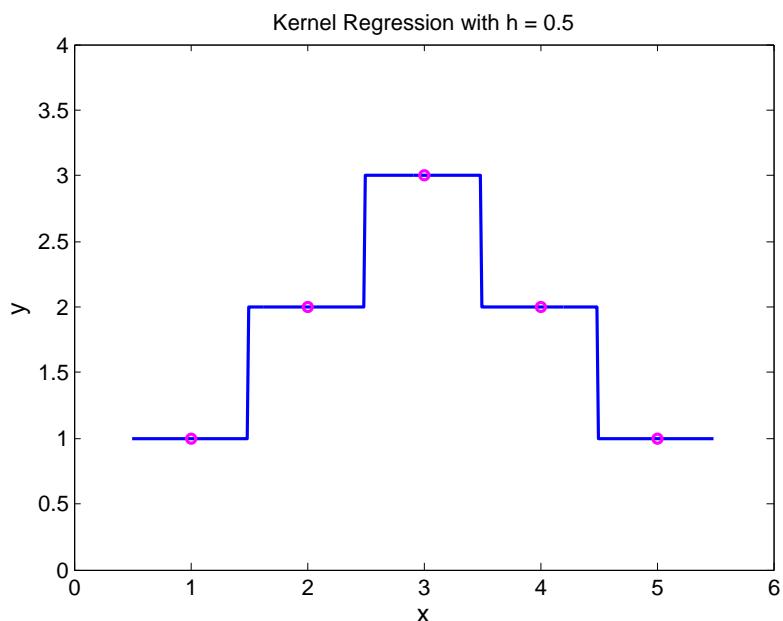
1. [4 points] Sketch the fit  $Y$  given  $X$  for the dataset given below using kernel regression with a box kernel

$$K(x_i, x_j) = I(-h \leq x_i - x_j < h) = \begin{cases} 1 & \text{if } -h \leq x_i - x_j < h \\ 0 & \text{otherwise} \end{cases}$$

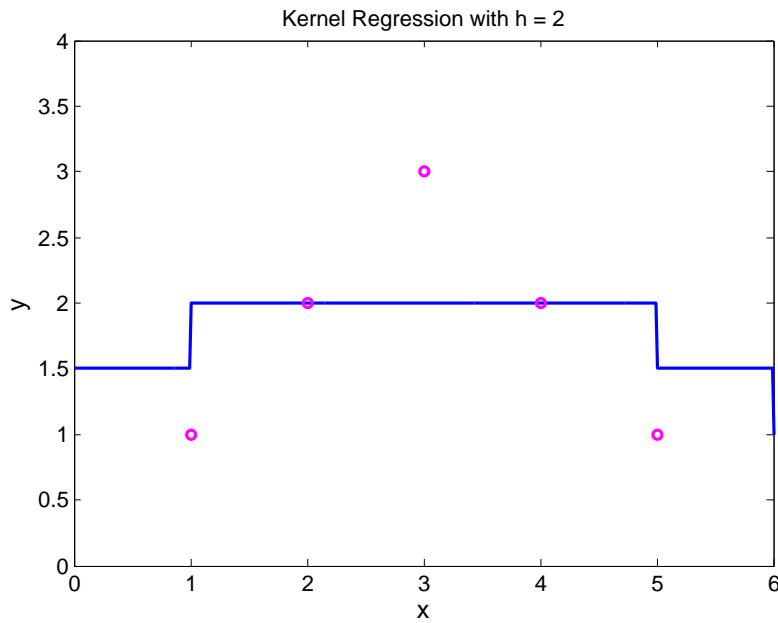
for  $h = 0.5, 2$ .

★ SOLUTION:

- $h = 0.5$

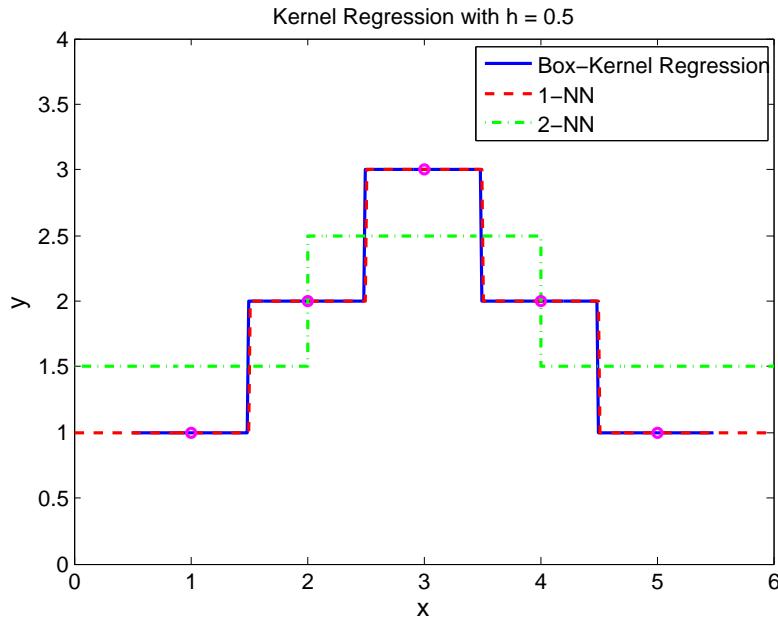


- $h = 2$

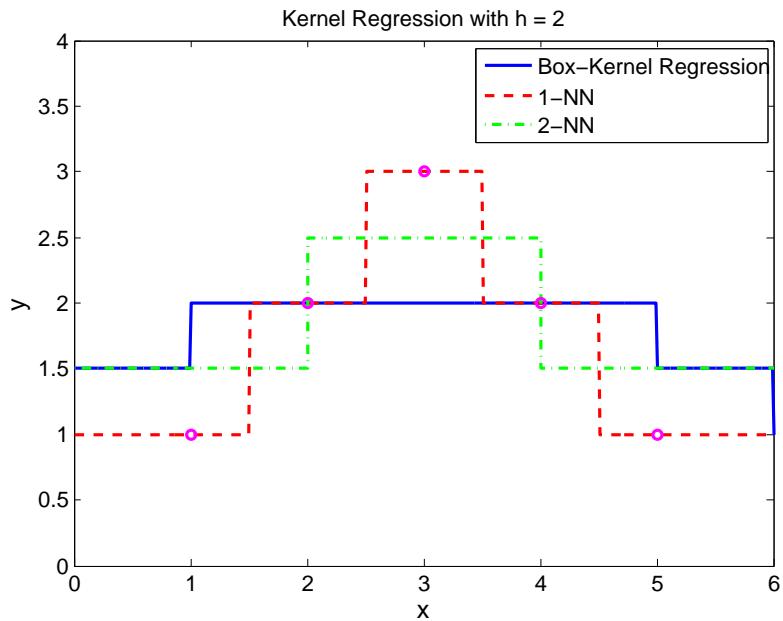


Just for fun, what happens with 1-NN and 2-NN:

- $h = 0.5$



- $h = 2$



- **COMMON MISTAKE 1:** Some people tried to *classify* the given points using  $k$ -NN instead of doing regression.
- **COMMON MISTAKE 2:** Many people sketched what seemed to be kernel regression or local linear regression with a Gaussian kernel.

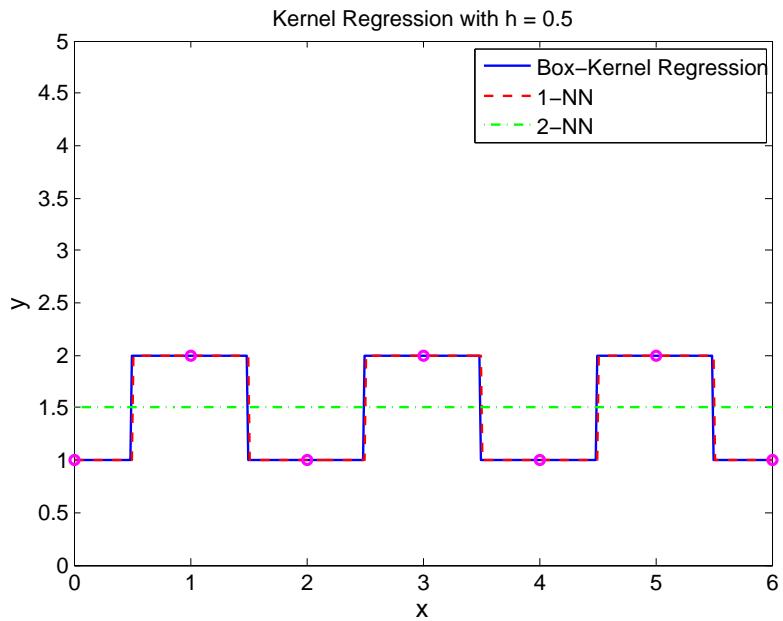
2. [4 points] Sketch or describe a dataset where kernel regression with the box kernel above with  $h = 0.5$  gives the same regression values as 1-NN but not as 2-NN in the domain  $x \in [0, 6]$  below.

★ **SOLUTION:** Part 1 of the problem with  $h = 0.5$  is actually an example where the regression values of kernel regression match 1-NN regression values (in the given domain). The basic idea is to create a dataset where in each interval of width 1 ( $2h$ ) there is only 1 “training” point and not all given  $y$  values are the same (then 1-NN and 2-NN would give the same regression values).

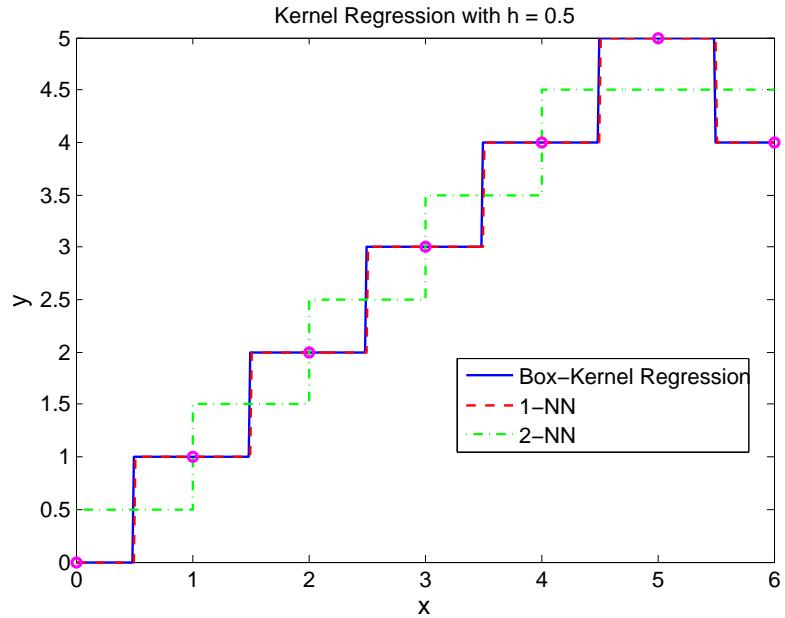
■ **COMMON MISTAKE 1:** Some people tried to come up with examples for  $k$ -NN *classification* instead of regression.

Here are some other example solutions. Note that you only had to give the points and here the lines are added just for you to see what happens.

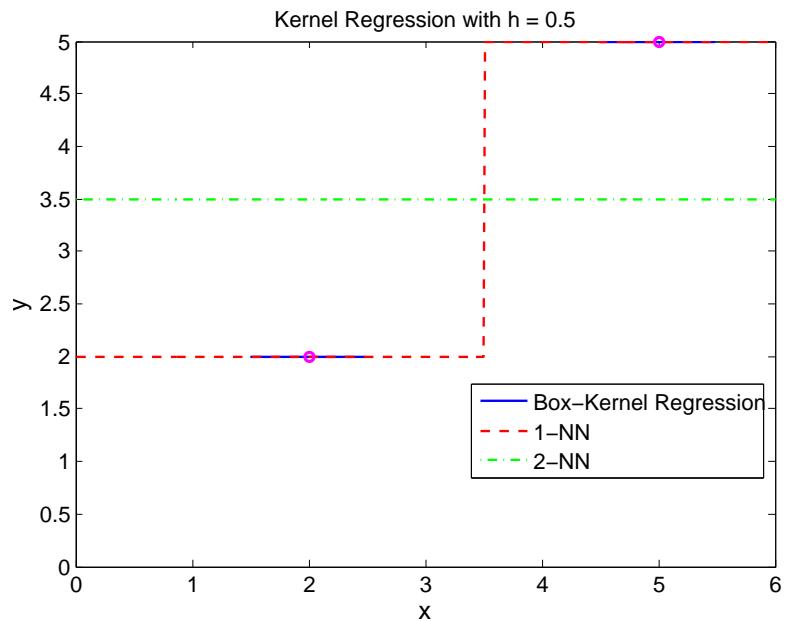
- Example 1



- Example 2



■ COMMON MISTAKE 2: Here kernel regression is not defined for some range (e.g.  $x \in [0, 1.5]$ ).

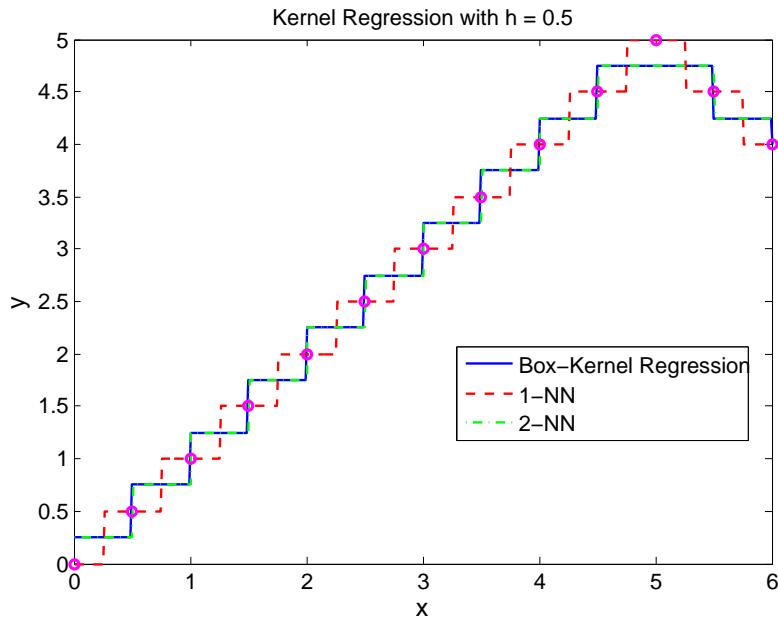


3. [4 points] Sketch or describe a dataset where kernel regression with the box kernel above with  $h = 0.5$  gives the same regression values as 2-NN but not as 1-NN in the domain  $x \in (0, 6)$  below.

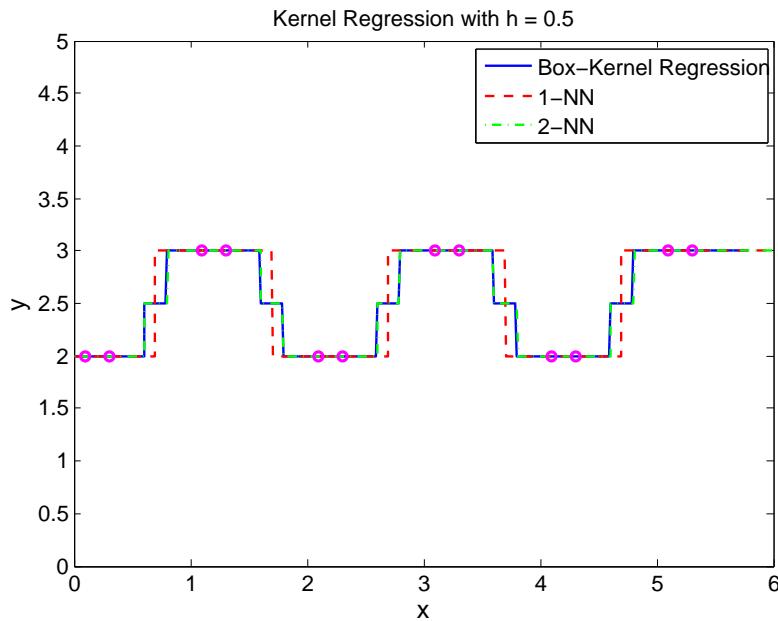
★ **SOLUTION:** As in Part 2, the basic idea is to create a dataset where in each interval of width 1 ( $2h$ ) there are 2 “training” points.

Here are some example solutions.

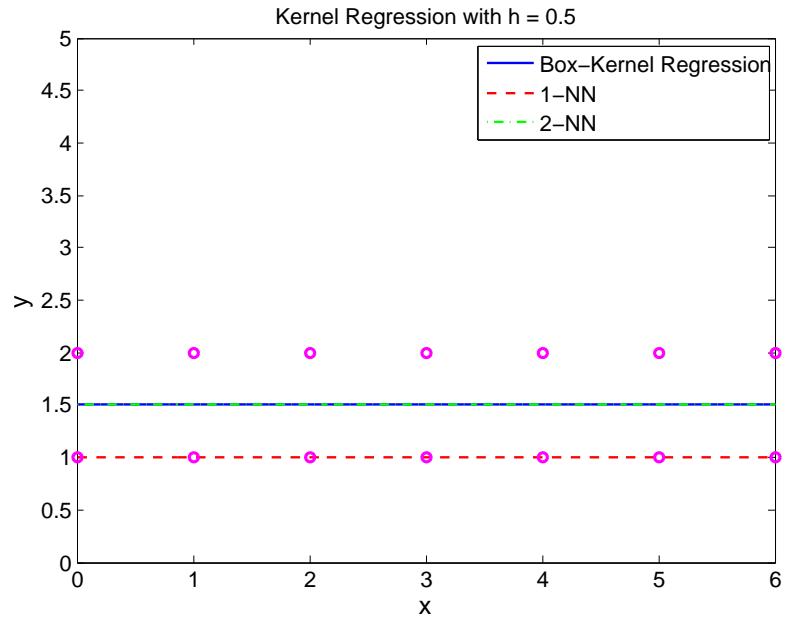
- Example 1



- Example 2



- Example 3: This example works if given a rule for 1-NN to break ties between two neighbors at the same distance from the test point, not average over both their values.



## 4 [14 Points] Model Selection

A central theme in machine learning is model selection. In this problem you will have the opportunity to demonstrate your understanding of various model selection techniques and their consequences. To make things more concrete we will consider the dataset  $\mathcal{D}$  given in [Equation 1](#) consisting of  $n$  independent identically distributed observations. The features of  $\mathcal{D}$  consist of pairs  $(x_1^i, x_2^i) \in \mathbb{R}^2$  and the observations  $y^i \in \mathbb{R}$  are continuous valued.

$$\mathcal{D} = \{(x_1^1, x_2^1), y^1), (x_1^2, x_2^2), y^2), \dots, (x_1^n, x_2^n), y^n)\} \quad (1)$$

Consider the abstract model given [Equation 2](#). The function  $f_{\theta_1, \theta_2}$  is a mapping from the features in  $\mathbb{R}^2$  to an observation in  $\mathbb{R}^1$  which depends on two parameters  $\theta_1$  and  $\theta_2$ . The  $\epsilon^i$  correspond to the noise. Here we will assume that the  $\epsilon^i \sim N(0, \sigma^2)$  are independent Gaussians with zero mean and variance  $\sigma^2$ .

$$y^i = f_{\theta_1, \theta_2}(x_1^i, x_2^i) + \epsilon^i \quad (2)$$

1. [4 Points] Show that the log likelihood of the data given the parameters is equal to [Equation 3](#).

$$l(D; \theta_1, \theta_2) = -\frac{1}{2\sigma^2} \sum_{i=1}^n (y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i))^2 - n \log \left( \sqrt{2\pi\sigma} \right) \quad (3)$$

Recall the probability density function of the  $N(\mu, \sigma^2)$  Gaussian distribution is given by [Equation 4](#).

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{(x - \mu)^2}{2\sigma^2} \right) \quad (4)$$

★ **SOLUTION:** The first thing one should think about is the probability of a single data point under this model. We can of course write this as:

$$y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i) = \epsilon^i$$

Because we know the distribution of  $\epsilon_i$  we know that:

$$y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i) \sim N(0, \sigma^2)$$

We can therefore write the likelihood of the data as:

$$\mathcal{L}(D; \theta_1, \theta_2) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{(y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i))^2}{2\sigma^2} \right)$$

To compute the log likelihood we take the log of the likelihood from above to obtain:

$$\begin{aligned}
l(D; \theta_1 \theta_2) &= \log(\mathcal{L}(D; \theta_1 \theta_2)) \\
&= \log \left( \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{(y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i))^2}{2\sigma^2} \right) \right) \\
&= \sum_{i=1}^n \log \left( \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{(y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i))^2}{2\sigma^2} \right) \right) \\
&= \sum_{i=1}^n \left( -\log(\sqrt{2\pi}\sigma) - \frac{(y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i))^2}{2\sigma^2} \right) \\
&= -n \log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i))^2
\end{aligned}$$

2. [1 Point] If we disregard the parts that do not depend on  $f_{\theta_1, \theta_2}$  and  $Y$  the negative of the log-likelihood given in [Equation 3](#) is equivalent to what commonly used loss function?

★ **SOLUTION:** The negative of the log likelihood is the square loss function. This is the loss function used in least squares regression.

3. [2 Points] Many common techniques used to find the maximum likelihood estimates of  $\theta_1$  and  $\theta_2$  rely on our ability to compute the gradient of the log-likelihood. Compute the gradient of the log likelihood with respect to  $\theta_1$  and  $\theta_2$ . Express your answer in terms of:

$$y^i, \quad f_{\theta_1, \theta_2}(x_1^i, x_2^i), \quad \frac{\partial}{\partial \theta_1} f_{\theta_1, \theta_2}(x_1^i, x_2^i), \quad \frac{\partial}{\partial \theta_2} f_{\theta_1, \theta_2}(x_1^i, x_2^i)$$

★ **SOLUTION:** The important technique we use here is the chain rule. To save space I will take the gradient with respect to  $\theta_j$ .

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} l(D; \theta_1, \theta_2) &= -\frac{1}{2\sigma^2} \sum_{i=1}^n \frac{\partial}{\partial \theta_j} (y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i))^2 \\
&= -\frac{1}{2\sigma^2} \sum_{i=1}^n 2(y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i)) \frac{\partial}{\partial \theta_j} (-f_{\theta_1, \theta_2}(x_1^i, x_2^i)) \\
&= \frac{1}{\sigma^2} \sum_{i=1}^n (y^i - f_{\theta_1, \theta_2}(x_1^i, x_2^i)) \frac{\partial}{\partial \theta_j} f_{\theta_1, \theta_2}(x_1^i, x_2^i)
\end{aligned}$$

■ **COMMON MISTAKE 1:** Many people forgot about the negative inside  $(\cdot)^2$ . This is very important as it would result in an algorithm that finds the minimal likelihood.

4. [2 Points] Given the learning rate  $\eta$ , what update rule would you use in gradient descent to *maximize* the likelihood.

★ **SOLUTION:** The tricky part about this problem is deciding whether to add or subtract the gradient. The easiest way to think about this is consider a simple line. If the slope is positive then adding the slope to the  $x$  value will result in a larger  $y$  value. Because we are trying to maximize the likelihood we will add the gradient:

$$\theta_j^{(t+1)} = \theta_j^{(t)} + \eta \frac{\partial}{\partial \theta_j} l(D; \theta_1, \theta_2) \quad (5)$$

■ **COMMON MISTAKE 1:** Many people had the sign backwards which is what we normally use when we are trying to *minimize* some loss function.

5. [3 Points] Suppose you are given some function  $h$  such that  $h(\theta_1, \theta_2) \in \mathbb{R}$  is large when  $f_{\theta_1, \theta_2}$  is complicated and small when  $f_{\theta_1, \theta_2}$  is simple. Use the function  $h$  along with the negative log-likelihood to write down an expression for the regularized loss with parameter  $\lambda$ .

★ **SOLUTION:** For this problem we simply write the regularized loss as the negative log likelihood plus the regularization term:

$$loss_{reg}(D; \theta_1, \theta_2) = -l(D; \theta_1, \theta_2) + \lambda h(\theta_1, \theta_2) \quad (6)$$

Based on this equation we see that to minimize the loss we want to maximize the likelihood and minimize the model complexity.

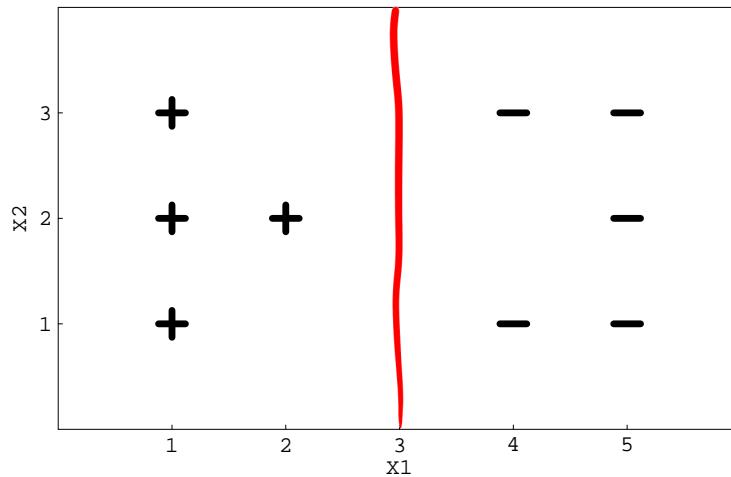
■ **COMMON MISTAKE 1:** Again, the signs are important here. Conceptually we want to minimize model complexity and maximize model fit (or likelihood). Since we always try to minimize the regularized loss we know that we want a negative sign on the likelihood and positive sign on the regularization term  $h(\theta_1, \theta_2)$ .

6. [2 Points] For small and large values of  $\lambda$  describe the bias variance trade off with respect to the regularized loss provided in the previous part.

★ **SOLUTION:** As we increase the value of  $\lambda$  we place a greater penalty on model complexity resulting in simpler models, more model bias, and less model variance. Alternatively, if we decrease the value of  $\lambda$  then we permit more complex models resulting in less model bias and greater model variance.

## 5 [12 points] Support Vector Machine

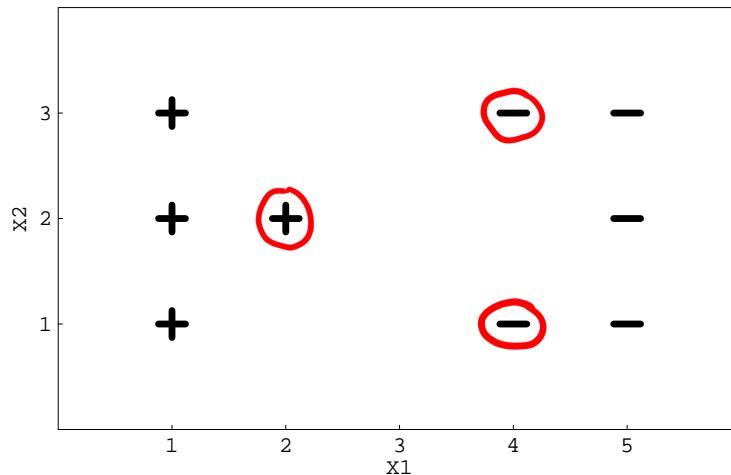
1. [2 points] Suppose we are using a linear SVM (i.e., no kernel), with some large  $C$  value, and are given the following data set.



Draw the decision boundary of linear SVM. Give a brief explanation.

★ **SOLUTION:** Because of the large  $C$  value, the decision boundary will classify all of the examples correctly. Furthermore, among separators that classify the examples correctly, it will have the largest margin (distance to closest point).

2. [3 points] In the following image, circle the points such that removing that example from the training set and retraining SVM, we would get a different decision boundary than training on the full sample.



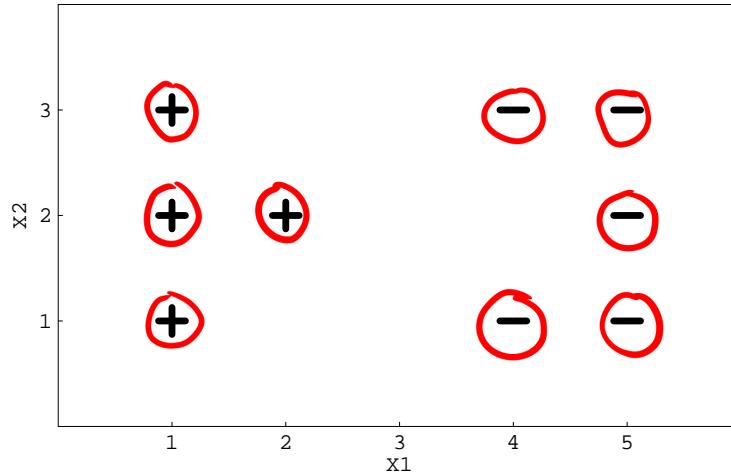
You do not need to provide a formal proof, but give a one or two sentence explanation.

★ **SOLUTION:** These examples are the support vectors; all of the other examples are such that their corresponding constraints are not tight in the optimization problem, so removing them will not create a solution with smaller objective function value (norm of  $w$ ). These three examples are positioned such that removing any one of them introduces slack in the constraints, allowing for a solution with a smaller objective function value and with a different third support vector; in this case, because each of these new (replacement) support vectors is not close to the old separator, the decision boundary shifts to make its distance to that example equal to the others.

3. [3 points] Suppose instead of SVM, we use regularized logistic regression to learn the classifier. That is,

$$(w, b) = \arg \min_{w \in \mathbb{R}^2, b \in \mathbb{R}} \frac{\|w\|^2}{2} - \sum_i \mathbb{1}[y^{(i)} = 0] \ln \frac{1}{1 + e^{(w \cdot x^{(i)} + b)}} + \mathbb{1}[y^{(i)} = 1] \ln \frac{e^{(w \cdot x^{(i)} + b)}}{1 + e^{(w \cdot x^{(i)} + b)}}.$$

In the following image, circle the points such that removing that example from the training set and running regularized logistic regression, we would get a different decision boundary than training with regularized logistic regression on the full sample.



You do not need to provide a formal proof, but give a one or two sentence explanation.

★ **SOLUTION:** Because of the regularization, the weights will not diverge to infinity, and thus the probabilities at the solution are not at 0 and 1. Because of this, *every* example contributes to the loss function, and thus has an influence on the solution.

4. [4 points] Suppose we have a kernel  $K(\cdot, \cdot)$ , such that there is an implicit high-dimensional feature map  $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^D$  that satisfies  $\forall x, z \in \mathbb{R}^d, K(x, z) = \phi(x) \cdot \phi(z)$ , where  $\phi(x) \cdot \phi(z) = \sum_{i=1}^D \phi(x)_i \phi(z)_i$  is the dot product in the  $D$ -dimensional space.

Show how to calculate the Euclidean distance in the  $D$ -dimensional space

$$\|\phi(x) - \phi(z)\| = \sqrt{\sum_{i=1}^D (\phi(x)_i - \phi(z)_i)^2}$$

without explicitly calculating the values in the  $D$ -dimensional vectors. For this question, you should provide a formal proof.

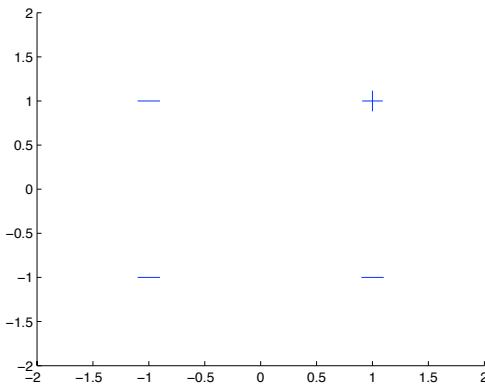
★ SOLUTION:

$$\begin{aligned} \|\phi(x) - \phi(z)\| &= \sqrt{\sum_{i=1}^D (\phi(x)_i - \phi(z)_i)^2} \\ &= \sqrt{\sum_{i=1}^D \phi(x)_i^2 + \phi(z)_i^2 - 2\phi(x)_i \phi(z)_i} \\ &= \sqrt{\left(\sum_{i=1}^D \phi(x)_i^2\right) + \left(\sum_{i=1}^D \phi(z)_i^2\right) - \left(\sum_{i=1}^D 2\phi(x)_i \phi(z)_i\right)} \\ &= \sqrt{\phi(x) \cdot \phi(x) + \phi(z) \cdot \phi(z) - 2\phi(x) \cdot \phi(z)} \\ &= \sqrt{K(x, x) + K(z, z) - 2K(x, z)}. \end{aligned}$$

## 6 [30 points] Decision Tree and Ensemble Methods

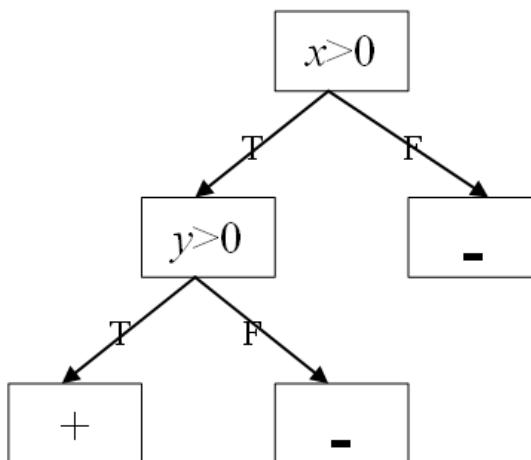
An ensemble classifier  $H_T(x)$  is a collection of  $T$  weak classifiers  $h_t(x)$ , each with some weight  $\alpha_t$ ,  $t = 1, \dots, T$ . Given a data point  $x \in \mathbb{R}^d$ ,  $H_T(x)$  predicts its label based on the weighted majority vote of the ensemble. In the binary case where the class label is either 1 or -1,  $H_T(x) = \text{sgn}(\sum_{t=1}^T \alpha_t h_t(x))$ , where  $h_t(x) : \mathbb{R}^d \rightarrow \{-1, 1\}$ , and  $\text{sgn}(z) = 1$  if  $z > 0$  and  $\text{sgn}(z) = -1$  if  $z \leq 0$ . Boosting is an example of ensemble classifiers where the weights are calculated based on the training error of the weak classifier on the weighted training set.

1. [10 points] For the following data set,



- Describe a binary decision tree with the minimum depth and consistent with the data;

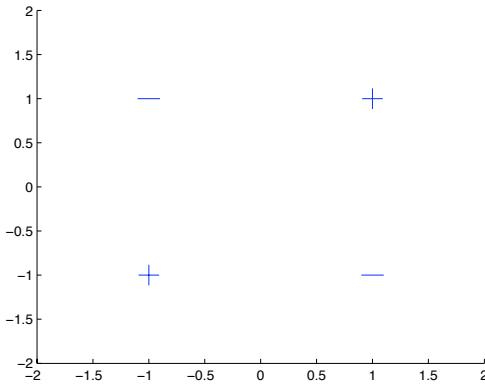
★ **SOLUTION:** The decision tree is as follows.



- Describe an ensemble classifier  $H_2(x)$  with 2 weak classifiers that is consistent with the data. The weak classifiers should be simple decision stumps. Specify the weak classifiers and their weights.

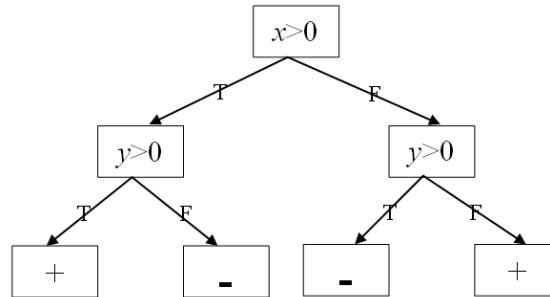
★ SOLUTION:  $h_1(x) = \text{sgn}(x), \alpha_1 = 1, h_2(x) = \text{sgn}(y), \alpha_2 = 1$ .

2. [10 points] For the following XOR data set,

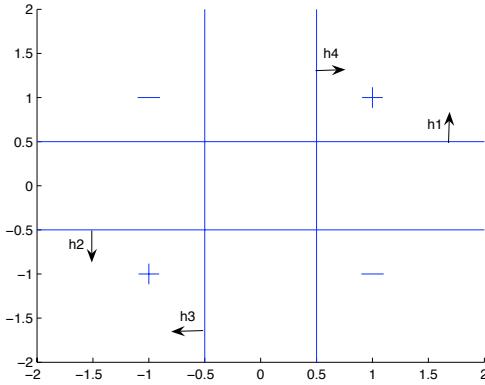


- Describe a binary decision tree with the minimum depth and consistent with the data;

★ SOLUTION: The decision tree is as follows.



- Let the ensemble classifier consist of the four binary classifiers shown below (the arrow means that the corresponding classifier classifies every data point in that direction as +), prove that there are no weights  $\alpha_1, \dots, \alpha_4$ , that make the ensemble classifier consistent with the data.



★ **SOLUTION:** Based on the binary classifiers, we have the following inequalities.

$$\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 > 0$$

$$-\alpha_1 + \alpha_2 - \alpha_3 + \alpha_4 \leq 0$$

$$-\alpha_1 + \alpha_2 + \alpha_3 - \alpha_4 > 0$$

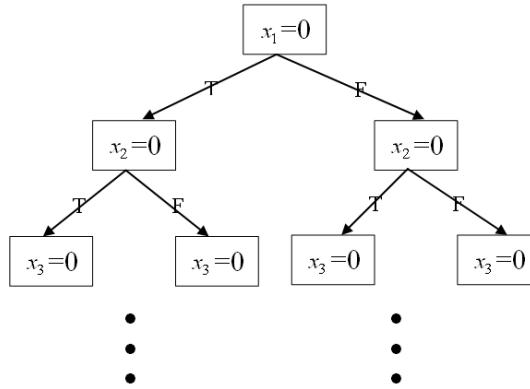
$$\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 \leq 0$$

Apparently, the first and third equations can not be satisfied at the same time. Therefore, there are no weights  $\alpha_1, \dots, \alpha_4$ , that make the ensemble classifier consistent with the data.

3. [10 points] Suppose that for each data point, the feature vector  $x \in \{0, 1\}^m$ , i.e.,  $x$  consists of  $m$  binary valued features, the class label  $y \in \{-1, 1\}$ , and the true classifier is a majority vote over the features, i.e.  $y = \text{sgn}(\sum_{i=1}^m (2x_i - 1))$ , where  $x_i$  is the  $i^{\text{th}}$  component of the feature vector.

- Describe a binary decision tree with the minimum depth and consistent with the data. How many leaves does it have?

★ SOLUTION: The decision tree looks as follows.



Any answers between  $2^{\frac{m}{2}}$  and  $2^m$  will get full points.

- Describe an ensemble classifier with the minimum number of weak classifiers. Specify the weak classifiers and their weights.

★ SOLUTION: The ensemble classifier has  $m$  weak classifiers.  $h_i(x) = 2x_i - 1, \alpha_i = 1, i = 1, \dots, m$

**Instructions.** (16 points)

You have 2 hours to answer this exam.

Answer all questions on this paper, just circle the correct answer. If you **make a mistake**, mark a cross through your wrong choice and circle your next alternative. This paper is the only one that you have to deliver at the end of the exam.

For the multiple choice questions each correct answer scores the value of question divided by the number of correct answers of the question (2 or 3).

**Wrong answers penalize** 1/3 of the value of the question for questions with only one answer and 1/2 of the value of the question for each wrong answer in the multiple choice questions.

- (2pts) 1. Assume that you have a concept description language that only allows pure conjunctive formulas and we have a domain with the following attributes:

Attr1:	A, B
Attr2:	A, B, C
Attr3	1, 2, 3
Attr4:	a, b
Attr5:	1, 2, 3, 4

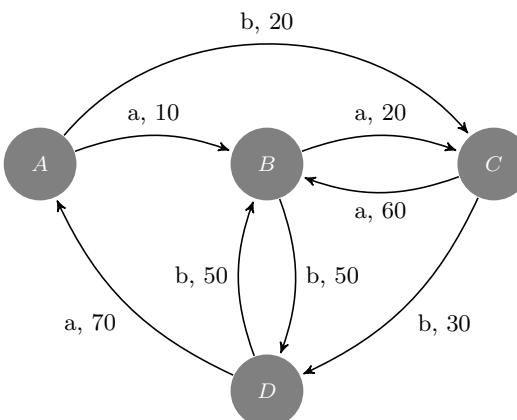
Given the following examples:

- + (A,B,2,a,3)
- (B,B,1,a,3)
- + (A,C,2,a,1)
- (B,A,1,a,3)
- + (A,B,2,a,4)
- (A,C,3,b,3)
- (A,C,2,a,2)

What concept learns the Version Space algorithm?

- (a) (A, \*, 2, a, \*)
- (b) There is not enough examples to converge
- (c) The examples are inconsistent
- (d) (A, \*, 2, \*, \*)

- (2pts) 2. We want to build a system able to control a process with four states {A, B, C, D} where we can perform the actions **a** and **b**. The following figure shows the state transition function ( $\delta$ ) and the reinforcement ( $r$ ) obtained by each action:



Using the Q-Learning algorithm with  $\gamma = 0.9$  and  $\alpha = 1$ , the action value function  $Q$  that is obtained with the sequence that begins in the state  $A$  and performs the actions  $\{a,a,b,a,b,a\}$  is:

	$Q(s,a)$	A	B	C	D
(a)	a	10	20	30	70
	b	20	0	60	0

	$Q(s,a)$	A	B	C	D
(b)	a	10	20	78	79
	b	47	0	30	0

	$Q(s,a)$	A	B	C	D
(c)	a	10	20	30	70
	b	20	50	60	50

	$Q(s,a)$	A	B	C	D
(d)	a	78	10	47	30
	b	79	0	20	0

- (1<sup>pt</sup>) 3. Thinking about unsupervised learning and the k-Means and expectation maximization (EM) algorithms, which one of the following statements is true:
- (a) K-Means algorithm fits clusters only in hyperspheres and EM algorithm fits data only in hyperelipsoids
  - (b) K-Means using euclidean distance is a particular case of the EM-algorithm when we are fitting K-gaussian distributions with the same variance for each attribute.
  - (c) EM algorithm has the same computational cost no matter the number of parameters that have to be estimated for the probability distribution that we are fitting to the attributes.
  - (d) K-Means assigns a probability to the membership of each example to each cluster
- (1<sup>pt</sup>) 4. Thinking about unsupervised learning, which ones of the following statements are true (multiple choice):
- (a) Differently from partitional graph based algorithms and density estimation algorithms the K-means and the EM algorithm need as a parameter the number of clusters to find
  - (b) With hierarchical clustering algorithms based on graph theory we obtain a partition of a dataset in K different classes
  - (c) The K-Means algorithm obtains a global optimal solution for the partition of a dataset by minimizing the square distance between examples and their nearest centroid
  - (d) The EM algorithm assumes that the model of the data comes from a mixture of K n-dimensional probability distributions
- (1<sup>pt</sup>) 5. Thinking about the naive bayes algorithm and bayesian networks, which ones of the following statements are true (multiple choice):
- (a) Naive Bayes is a probabilistic method that assumes that the statistical distribution of the examples is independent
  - (b) Naive Bayes is a particular case of a Bayesian Network where all the attributes are assumed to be independent and that their individual distributions are not independent of the class attribute
  - (c) A Bayesian Network represents the joint probability distribution of a set of variables
  - (d) Bayesian Networks are unsupervised methods and can not be used for classification
- (1<sup>pt</sup>) 6. Thinking about algorithms for rule induction, which ones of the following statements are true (multiple choice):
- (a) The advantage of rule induction over learning a decision tree is that usually the rules obtained are shorter than the ones that can be extracted from a decision tree
  - (b) The sequential covering algorithm for rule induction is based on learning sequentially the rules with the lower accuracy until all the dataset is covered
  - (c) ILP methods learn propositional rules and decision trees and rule induction algorithms learn first order rules
  - (d) FOIL is an ILP method that learns rules by sequentially specializing a candidate rule

- (1<sup>pt</sup>) 7. Thinking about learning a bayesian network topology, which ones of the following statements are true (multiple choice):
- (a) To learn a bayesian network for a dataset means to search in the space of Directed Acyclic Graphs that can be built with the attributes
  - (b) Bayesian networks only can be applied to datasets where all the attribute are discrete
  - (c) The heuristic functions used in learning a BN topology usually include a term that represents the a priori information about the network, another that evaluates the fitting of the data to the probability distribution function defined by the current topology and a penalization for the network complexity
  - (d) The K2 algorithm uses a hill climbing algorithm to learn the topology of a bayesian network assuming that any order of precedence among the variables in the network is initially possible
- (1<sup>pt</sup>) 8. Given a K-nn classifier which ones of the following statements are true (multiple choice):
- (a) The more examples are used for classifying an example, the higher accuracy we obtain
  - (b) The more attributes we use to describe the examples the more difficult is to obtain high accuracy
  - (c) The most costly part of this method is to learn the model
  - (d) We can use K-nn for classification and regression
- (1<sup>pt</sup>) 9. Thinking about Reinforcement Learning which ones of the following statements are true (multiple choice):
- (a) The maximization of the future cumulative reward allows to Reinforcement Learning to perform global decisions with local information
  - (b) Q-learning is a temporal difference RL method that does not need a model of the task to learn the action value function
  - (c) Reinforcement Learning only can be applied to problems with a finite number of states
  - (d) In Markov Decision Problems (MDP) the future actions from a state depend on the previous states
- (1<sup>pt</sup>) 10. Thinking about reinforcement learning which one of the following statements is true:
- (a) Estimation using Dynamic Programming is less computational costly than using Temporal Difference Learning
  - (b) Estimating using Montecarlo methods has the advantage that it is not needed to have absorbent states in the problem
  - (c) Temporal Difference learning allows on-line learning and Montecarlo methods need complete training sequences for estimation
  - (d) Dynamic Programming and Montecarlo methods only work if we know the transitions probabilities for the actions and the reward function
- (1<sup>pt</sup>) 11. Thinking about dimensionality reduction and attribute selection, which ones of the following statements are true (multiple choice):
- (a) PCA and ICA transform a dataset to a space with the same dimensionality optimizing a measure that preserves the distances among all the pairs of examples
  - (b) PCA and multidimensional scaling are unsupervised methods for dimensionality reduction
  - (c) Wrappers are feature selection methods that, given a classifier as a performance criteria, search in the space of subset of features for the minimal one that obtains the higher accuracy
  - (d) Filters are unsupervised feature selection methods because they do not use a classifier as selection criteria

- (1pt) **12.** Thinking about Explanation Based Learning (EBL), which ones of the following statements are true (multiple choice):
- (a) The goal regression algorithm is used for the generalization of resolution traces
  - (b) In EBL only an example is needed to perform the learning
  - (c) EBL does not allow to discover new knowledge, it only makes explicit knowledge that we already have
  - (d) EBL learning is not used in problem solving
- (1pt) **13.** Thinking about classifier ensembles (meta-learning), which ones of the following statements are true (multiple choice):
- (a) Classifier ensembles usually have better performance for weak classifiers because they combine the different points of view of each individual classifier
  - (b) Boosting is an ensemble method that uses sampling with replacement of a dataset and learns independently several classifiers
  - (c) Random subspaces methods (like random forest) are meta-learning methods that learn a set of classifiers using datasets that have subsets of attributes of the original data
  - (d) For methods like Boosting or Bagging to perform well, it is necessary that the different subsamples from the dataset used in each classifier are as similar as possible
- (1pt) **14.** Given a dataset we first learn a decision tree and we apply after that a postpruning to the tree. We look at the pruned tree and we identify in the leaves what examples are in the majority class. We consider the examples not in the majority class of the leaves as misclassifications. We create a new dataset only with the examples in the majority classes of the leaves. We train a new decision tree only with this examples. Which one of this statements is true:
- (a) The new tree will be the same than the pruned original tree
  - (b) The new tree will be different and, given a new test set, the error of this new tree will be the same than the one of the original pruned tree with this new test set
  - (c) The new tree will be different, but, given a new test set, the error of this new tree with this new test set will be lower than one of the original pruned tree
  - (d) None of the above

# EXAM IN STATISTICAL MACHINE LEARNING

## STATISTISK MASKININLÄRNING

DATE AND TIME: March 10, 2017, 8.00–13.00

RESPONSIBLE TEACHER: Fredrik Lindsten

NUMBER OF PROBLEMS: 5

AIDING MATERIAL: Calculator, mathematical handbooks

PRELIMINARY GRADES: grade 3 23 points  
grade 4 33 points  
grade 5 43 points

Some general instructions and information:

- Your solutions can be given in Swedish or in English.
- Only write on one page of the paper.
- Write your exam code and a page number on all pages.
- Do not use a red pen.
- Use separate sheets of paper for the different problems (i.e. the numbered problems, 1–5).
- When asked to pair e.g. plots with corresponding formulas, the order of the plots/formulas is always randomly generated using the function `sample(n)` in R, where `n` is the number of options. Thus, it is not possible to infer the correct answer from the way in which the problem is presented.

*With the exception of Problem 1, all your answers must be clearly motivated! A correct answer without a proper motivation will score zero points!*

Good luck!



## Some useful formulas

Pages 1–3 contain some expressions that may or may not be useful for solving the exam problems. *This is not a complete list of formulas used in the course!* Consequently, some of the problems may require knowledge about certain expressions not listed here. Furthermore, the formulas listed below are not all self-explanatory, meaning that you need to be familiar with the expressions to be able to interpret them. Thus, the list should be viewed as a support for solving the problems, rather than as a comprehensive collection of formulas.

**Marginalization and conditioning of probability densities:** For a partitioned random vector  $Z = (Z_1^\top \ Z_2^\top)^\top$  with joint probability density function  $p(z) = p(z_1, z_2)$ , the marginal probability density function of  $Z_1$  is

$$p(z_1) = \int_{Z_2} p(z_1, z_2) dz_2$$

and the conditional probability density function for  $Z_1$  given  $Z_2 = z_2$  is

$$p(z_1 | z_2) = \frac{p(z_1, z_2)}{p(z_2)} = \frac{p(z_2 | z_1)p(z_1)}{p(z_2)}.$$

**The Gaussian distribution:** The probability density function of the  $p$ -dimensional Gaussian distribution is

$$\mathcal{N}(x | \mu, \Sigma) = \frac{1}{(2\pi)^{p/2} \sqrt{\det \Sigma}} \exp\left(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu)\right), \quad x \in \mathbb{R}^p.$$

For a Gaussian random vector  $X \sim \mathcal{N}(\mu, \Sigma)$  partitioned according to,

$$X = \begin{pmatrix} X_a \\ X_b \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \Sigma_a & \Sigma_{ab} \\ \Sigma_{ab}^\top & \Sigma_b \end{pmatrix}$$

it holds that the marginal probability density of  $X_a$  is  $p(x_a) = \mathcal{N}(x_a | \mu_a, \Sigma_a)$  and the conditional density of  $X_a$  given  $X_b = x_b$  is  $p(x_a | x_b) = \mathcal{N}(x_a | \mu_{a|b}, \Sigma_{a|b})$ , where

$$\begin{aligned} \mu_{a|b} &= \mu_a + \Sigma_{ab}\Sigma_b^{-1}(x_b - \mu_b), \\ \Sigma_{a|b} &= \Sigma_a - \Sigma_{ab}\Sigma_b^{-1}\Sigma_{ab}^\top. \end{aligned}$$

If we have that  $X_a$ , as well as  $X_b$  conditioned on  $X_a = x_a$ , are Gaussian distributed according to

$$p(x_a) = \mathcal{N}(x_a | \mu_a, \Sigma_a),$$

$$p(x_b | x_a) = \mathcal{N}(x_b | Mx_a + b, \Sigma_{b|a}),$$

where  $M$  is a matrix (of appropriate dimension) and  $b$  is a constant vector, then the joint distribution of  $X_a$  and  $X_b$  is given by

$$p(x_a, x_b) = \mathcal{N}\left(\begin{pmatrix} x_a \\ x_b \end{pmatrix} \middle| \begin{pmatrix} \mu_a \\ M\mu_a + b \end{pmatrix}, \begin{pmatrix} \Sigma_a & \Sigma_a M^\top \\ M\Sigma_a & \Sigma_{b|a} + M\Sigma_a M^\top \end{pmatrix}\right).$$

**Sum of identically distributed variables:** For identically distributed random variables  $\{Z_i\}_{i=1}^n$  with mean  $\mu$ , variance  $\sigma^2$  and average correlation between distinct variables  $\rho$ , it holds that  $\mathbb{E}\left[\frac{1}{n} \sum_{i=1}^n Z_i\right] = \mu$  and  $\text{Var}\left(\frac{1}{n} \sum_{i=1}^n Z_i\right) = \frac{1-\rho}{n} \sigma^2 + \rho \sigma^2$ .

### Linear regression and regularization:

- The least-squares estimate of  $\beta$  in the linear regression model  $Y = \beta^\top X + \varepsilon$  is given by  $\hat{\beta}_{\text{LS}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ , where

$$\mathbf{X} = \begin{pmatrix} x_1^\top \\ \vdots \\ x_N^\top \end{pmatrix}, \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}.$$

- Ridge regression uses the regularization term  $\lambda \|\beta\|_2^2 = \lambda \sum_{j=0}^p \beta_j^2$ . The ridge regression estimate is  $\hat{\beta}_{\text{RR}} = (\mathbf{X}^\top \mathbf{X} + \lambda I)^{-1} \mathbf{X}^\top \mathbf{y}$ .
- LASSO uses the regularization term  $\lambda \|\beta\|_1 = \lambda \sum_{j=0}^p |\beta_j|$ . (The LASSO estimate does not admit a simple closed form expression.)
- For a probabilistic linear regression model with  $\varepsilon \sim \mathcal{N}(0, \sigma^2)$  and prior distribution  $p(\beta) = \mathcal{N}(\beta | \mu_0, \Sigma_0)$  the posterior distribution is  $p(\beta | \mathbf{y}) = \mathcal{N}(\beta | \mu_N, \Sigma_N)$  with

$$\mu_N = \Sigma_N \left( \Sigma_0^{-1} \mu_0 + \sigma^{-2} \mathbf{X}^\top \mathbf{y} \right), \quad \Sigma_N = \left( \Sigma_0^{-1} + \sigma^{-2} \mathbf{X}^\top \mathbf{X} \right)^{-1}.$$

**Maximum likelihood:** The maximum likelihood estimate is given by  $\hat{\beta}_{\text{ML}} = \arg \max_{\beta} \ell(\beta)$  where  $\ell(\beta) = \log p(\mathbf{y} | \beta) = \sum_{i=1}^N \log p(y_i | \beta)$  is the log-likelihood

function (the last equality holds when the  $N$  training data points are independent).

**Logistic regression:** The logistic regression model uses a linear regression for the log-odds. In the binary classification context we thus have

$$\log \left( \frac{\Pr(Y = 1 | X)}{\Pr(Y = 0 | X)} \right) = \beta^\top X.$$

**Discriminant Analysis:** The linear discriminant analysis (LDA) classifier assigns a test input  $X = x$  to class  $k$  for which,

$$\hat{\delta}_k(x) = x^\top \hat{\Sigma}^{-1} \hat{\mu}_k - \frac{1}{2} \hat{\mu}_k^\top \hat{\Sigma}^{-1} \hat{\mu}_k + \log \hat{\pi}_k$$

is largest, where  $\hat{\pi}_k = N_k/N$  and  $\hat{\mu}_k = \frac{1}{N_k} \sum_{i:y_i=k} x_i$  for  $k = 1, \dots, K$ , and  $\hat{\Sigma} = \frac{1}{N-K} \sum_{k=1}^K \sum_{i:y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^\top$ .

For quadratic discriminant analysis (QDA) we instead use the discriminant functions

$$\hat{\delta}_k(x) = -\frac{1}{2} \log |\hat{\Sigma}_k| - \frac{1}{2} (x - \hat{\mu}_k)^\top \hat{\Sigma}_k^{-1} (x - \hat{\mu}_k) + \log \hat{\pi}_k,$$

where  $\hat{\Sigma}_k = \frac{1}{N_k-1} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^\top$  for  $k = 1, \dots, K$ .

### Loss functions for classification:

- Misclassification loss:  $I(y \neq G(x))$ .
- Exponential loss:  $\exp(-yC(x))$  where  $G(x) = \text{sign}(C(x))$ .

**Gaussian processes:** If the function  $f$  is distributed according to a Gaussian process,  $f \sim \mathcal{GP}(m, k)$ , it holds that for any arbitrary selection of inputs  $\{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$  the output values  $f(x^{(1)}), f(x^{(2)}), \dots, f(x^{(n)})$  are jointly Gaussian,

$$\begin{pmatrix} f(x^{(1)}) \\ \vdots \\ f(x^{(n)}) \end{pmatrix} \sim \mathcal{N} \left( \begin{pmatrix} m(x^{(1)}) \\ \vdots \\ m(x^{(n)}) \end{pmatrix}, \begin{pmatrix} k(x^{(1)}, x^{(1)}) & \dots & k(x^{(1)}, x^{(n)}) \\ \ddots & \ddots & \vdots \\ k(x^{(n)}, x^{(1)}) & \dots & k(x^{(n)}, x^{(n)}) \end{pmatrix} \right),$$

For a Gaussian process regression model  $Y = f(X) + \varepsilon$  with  $f \sim \mathcal{GP}(m, k)$  and  $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ , the prediction model is given by

$$f(x_\star) | \mathbf{y} \sim \mathcal{N} \left( m(x_\star) + \mathbf{s}^\top (\mathbf{y} - m(\mathbf{X})), k(x_\star, x_\star) - \mathbf{s}^\top k(\mathbf{X}, x_\star) \right),$$

where  $\mathbf{s}^\top = k(x_\star, \mathbf{X})(k(\mathbf{X}, \mathbf{X}) + \sigma^2 I_N)^{-1}$ .

1. This problem is composed of 10 true-or-false statements. You only have to classify these as either **true** or **false**. For this problem (*only!*) no motivation is required. Each correct answer scores 1 point and each incorrect answer scores -1 point (capped at 0 for the whole problem).

- i. Regression problems have only quantitative inputs.
- ii. The following (so called “probit”) classifier is linear:

$$\hat{G}(x) = I(\Phi(\beta^T x) > 0.2)$$

where  $\Phi(x) = \int_{-\infty}^x \mathcal{N}(z|0,1)dz$  is the cumulative distribution function of the standard Gaussian distribution,  $I(\cdot)$  is the indicator function, and the class labels are 0 and 1.

- iii. Ensemble methods can be used to reduce both bias and variance (compared to the base model used).
- iv. The input partitioning shown in Figure 1 could **not** have been generated by recursive binary splitting.

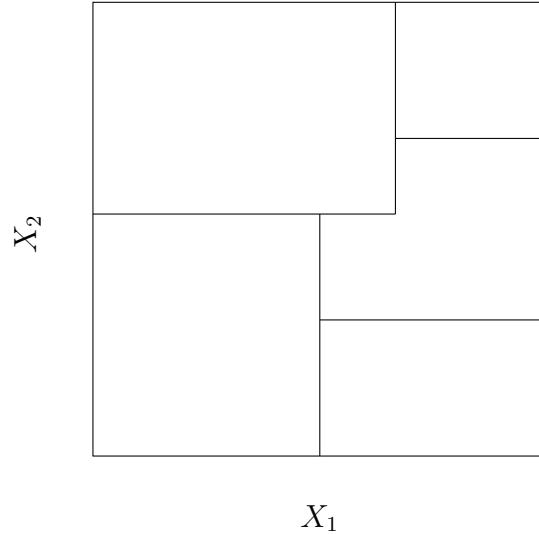


Figure 1: Input partitioning for Problem 1.iv.

- v. The model  $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$  (where  $\beta_0$  and  $\beta_1$  are model parameters) is a linear regression model.
- vi. Over-fitting for a  $k$ -NN classifier occurs when there is too much data, so that the  $k$ -neighborhoods become localized in the input space.

- vii. Maximum likelihood estimation is another word for least-squares estimation.
- viii. The mean-squared-error (in  $\hat{f}(X; \mathcal{T})$  w.r.t.  $f(X)$ ) can be decomposed into the sum of the squared model bias and the model variance, i.e.

$$\begin{aligned}\mathbb{E}[(\hat{f}(X; \mathcal{T}) - f(X))^2] = \\ (\mathbb{E}[\hat{f}(X; \mathcal{T}) - f(X)])^2 + \mathbb{E}[(\hat{f}(X; \mathcal{T}) - \mathbb{E}[\hat{f}(X; \mathcal{T})])^2].\end{aligned}$$

- ix. AdaBoost can use logistic regression as a base classifier.
- x. The  $k$ -NN classifier is sensitive to the scaling of the input variables.

(10p)

2. A large Swedish retailer of alcoholic beverages has asked you to build a model for predicting the price of a wine based on various types of information, such as the wine's chemical composition, region, etc. They have collected a database with their data, containing the following columns:
- **id** – a unique identification number for each row of the database, specified as an integer value
  - **grape** – one out of 6 different grapes (e.g. **Syrah**, **Zinfandel**, **Merlot**, etc.)
  - **alcohol** – percentage of alcohol in the wine, specified as a real number between 0 and 1 (e.g. 0.135 for 13.5%)
  - **year** – production year of the wine, specified as an integer in the range 2010–2016
  - **region** – one out of 78 different wine regions (e.g. **Bordeaux**, **Burgundy**, **Rhône**, etc.)
  - **proline** – concentration of the amino acid proline in mg/l specified as an integer (typically in the range 200–2000)
  - **timestamp** – a time stamp specifying the time when the row was entered into the database, on the format 'YYYY-MM-DD HH:MM:SS'
  - **price** – the current price of the wine in SEK, specified as an integer (typically in the range 80–400)

*Don't forget to clearly motivate all your answers!*

- (a) The customer want's to try a simple model first, like a linear regression or a logistic regression. Which one of these two methods do you suggest? (1p)
- (b) For each column of the customer's database as listed above, specify whether you would consider that variable as an input of the model, an output of the model, or neither. (4p)
- (c) For each of the inputs and outputs of your model (from the previous question), specify whether that variable is best viewed as quantitative or qualitative. (3p)
- (d) The customer has previously tried to use a CART<sup>1</sup> model for this problem, but ran into problems when trying to learn this model. In fact, they were not able to train the CART model at all! The

---

<sup>1</sup>CART=Classification And Regression Trees

issue was traced to the variable `region`. Considering specifically this variable, the splitting criteria they considered was to compute

$$\hat{I} = \arg \min_{I \subset R} \sum_{i: x_i \in I} L(y_i, \hat{c}_1) + \sum_{i: x_i \notin I} L(y_i, \hat{c}_2)$$

where  $L$  is a given loss function and  $\hat{c}_1$  and  $\hat{c}_2$  are constant predictions for the two regions obtained in the split. Furthermore,  $R$  is the set of all possible values for `region`, i.e.

$$R = \{\text{Bordeaux, Burgundy, Rhône, ...}\}$$

and the optimization is with respect to all proper subsets  $I \subset R$ . The CART implementation they used was based on a brute-force solution of the optimization problem above, i.e. all possible values of  $I$  are enumerated, the loss is computed, and  $\hat{I}$  is taken as the value of  $I$  which attains the smallest loss. Why did they not succeed in this approach? (2p)

3. (a) Consider the scatter plots in Figure 2 which depict three different training data sets for three binary classification problems. In which dataset(s) could the classes be well separated by...
- ... an LDA classifier?
  - ... a QDA classifier?

(In both cases we assume that  $X_1$  and  $X_2$  are the only inputs to the classifiers.) (2p)

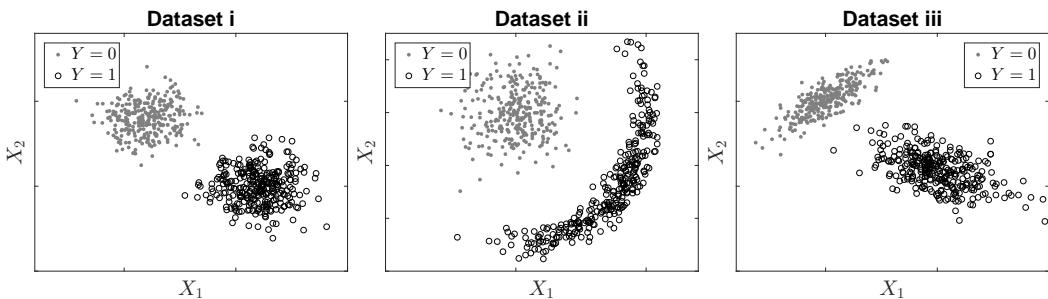


Figure 2: Scatter plots of training data for Problems 3a and 3b.

- (b) Consider again the scatter plots in Figure 2. The LDA and QDA classifiers are based on different *assumptions* about the properties of the data. Which dataset(s) in Figure 2 appear to correspond well to the assumptions made by LDA and QDA, respectively? (4p)

- (c) Consider a Gaussian process regression model  $Y = f(X) + \varepsilon$  with  $\varepsilon \sim \mathcal{N}(0, \sigma^2)$  and  $f \sim \mathcal{GP}(0, k(x, x'))$ . Five different GP models are fit to a given training data set  $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^3$ ,

**M1.** squared-exponential kernel, length scale  $\ell = 0.1$ ; noise variance  $\sigma^2 = 0.3^2$

**M2.** squared-exponential kernel, length scale  $\ell = 1$ ; noise variance  $\sigma^2 = 3^2$

**M3.** periodic kernel, length scale  $\ell = 1$  and period  $p = 3$ ; noise variance  $\sigma^2 = 0.3^2$

**M4.** Matérn-3 kernel, length scale  $\ell = 5$ ; noise variance  $\sigma^2 = 0.3^2$

**M5.** Matérn-3 kernel, length scale  $\ell = 1$ ; noise variance  $\sigma^2 = 0$

The kernels for the models M1-M5 are plotted in Figure 3. The four panels of Figure 4 show sample paths from the posterior distributions of  $f(x) | \mathcal{T}$  for *four out of these five* models. Specify which model that belongs to which panel in the figure. (4p)

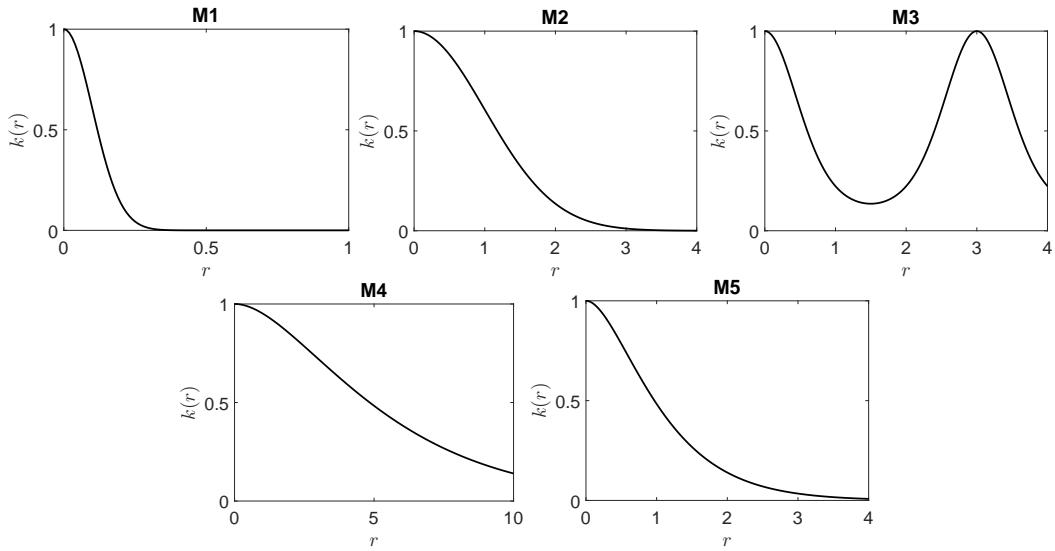


Figure 3: Kernels  $k(r) = k(|x - x'|)$  for the models in Problem 3c.

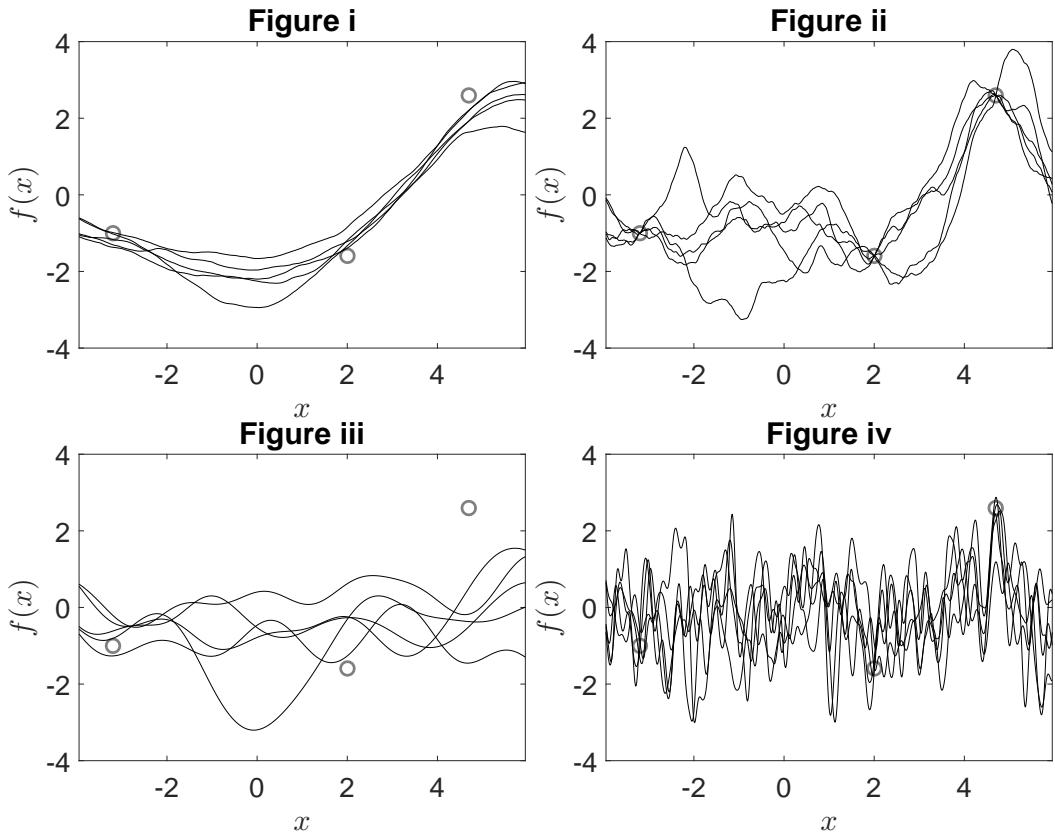


Figure 4: Sample paths from the posterior Gaussian process for four of the models in Problem 3c. The circles mark the three training data points.

4. (a) Consider the simple linear regression model with one input variable  $X$ ,

$$Y = f(X) + \varepsilon = \beta_0 + \beta_1 X + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2).$$

Assume that we have a training data set consisting of two data points  $\mathcal{T} = \{(x_1, y_1), (x_2, y_2)\} = \{(-1, 2), (1, 1)\}$ . What is the least-squares estimate  $\hat{\beta}_{\text{LS}}$  of  $\beta$ ? (2p)

- (b) What is the ridge regression estimate  $\hat{\beta}_{\text{RR}}$  of  $\beta$ ? Express your solution in terms of the regularization parameter  $\lambda$ . (2p)

- (c) What is the LASSO estimate  $\hat{\beta}_{\text{LASSO}}$  of  $\beta$ ? Express your solution in terms regularization parameter  $\lambda$ . (6p)

*Hint: In two dimensions, the LASSO estimate of  $\beta_j$  is either 0, or has the same sign as the least-squares estimate of  $\beta_j$ . (This does not hold in higher dimensions.) Furthermore, the LASSO estimate varies continuously with  $\lambda$  in a piecewise linear way.*

5. (a) Explain briefly the difference between parametric models and non-parametric models. (2p)
- (b) Consider a logistic regression model for a binary classification problem,  $Y \in \{0, 1\}$ , with log-odds for class 1 given by  $\hat{\beta}^T X$  where

$$\hat{\beta}^T = (\beta_0 \ \ \beta_1 \ \ \beta_2 \ \ \beta_3) = (3 \ -1 \ 3 \ 2),$$

where the parameter  $\beta_0$  corresponds to an offset (“intercept”) term. According to this model, what is the probability that the test input  $x_* = (2 \ 1 \ -1)$  belongs to class 0? (2p)

- (c) Consider a logistic regression model for a binary classification problem  $Y \in \{0, 1\}$ . We have observed a training data set  $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^N$  where the  $N$  training data points are mutually independent. Write down an expression for the likelihood, i.e. the probability of the observed training data

$$\Pr(Y_1 = y_1, \dots, Y_N = y_N \mid X_1 = x_1, \dots, X_N = x_N, \beta)$$

expressed in terms of the class-1 probabilities

$$p(x, \beta) := \Pr(Y = 1 \mid X = x, \beta).$$

(4p)

- (d) For the logistic regression model above, write down an expression for the log-likelihood function

$$\ell(\beta) = \log \Pr(Y_1 = y_1, \dots, Y_N = y_N \mid X_1 = x_1, \dots, X_N = x_N, \beta)$$

on the form,

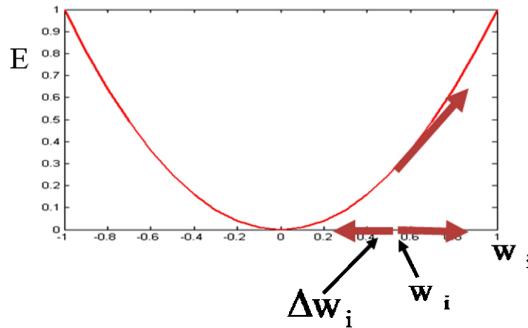
$$\ell(\beta) = \sum_{i=1}^N h(x_i, y_i, \beta).$$

The function  $h$  should be expressed only in terms of elementary functions (like additions, multiplications, logarithms, exponentials, ...) and its dependence on the variables  $x_i$ ,  $y_i$  and  $\beta$  should be explicit. (2p)

## EXAMPLE Machine Learning (C395) Exam Questions

- (1) **Question:** Explain the principle of the gradient descent algorithm. Accompany your explanation with a diagram. Explain the use of all the terms and constants that you introduce and comment on the range of values that they can take.

**Solution:** Training can be posed as an optimization problem, in which the goal is to optimize a function (usually to minimize a cost function  $E$ ) with respect to a number of free variables, usually weights  $w_i$ . The gradient decent algorithm begins from an initialization of the weights (e.g. a random initialization) and in an iterative procedure updates the weights  $w_i$  by a quantity  $\Delta w_i$ , where  $\Delta w_i = -\alpha (\partial E / \partial w_i)$  and  $(\partial E / \partial w_i)$  is the gradient of the cost function with respect to the weights, while  $\alpha$  is a constant which takes small values in order to keep the updates low and avoid oscillations.



- (2) **Question:** Derive the gradient descent training rule assuming that the target function representation is:

$$o_d = w_0 + w_1 x_1 + \dots + w_n x_n.$$

Define explicitly the cost/error function  $E$ , assuming that a set of training examples  $D$  is provided, where each training example  $d \in D$  is associated with the target output  $t_d$ .

**Solution:** The error function:  $E = \sum_{d \in D} (t_d - o_d)^2$

The gradient decent algorithm:  $\Delta w_i = -\alpha (\partial E / \partial w_i)$

First represent  $(\partial E / \partial w_i)$  in terms of the unit inputs  $x_{id}$ , outputs  $o_d$ , and target values  $t_d$ :

$$\begin{aligned} (\partial E / \partial w_i) &= (\partial \sum_{d \in D} (t_d - o_d)^2 / \partial w_i) = \sum_{d \in D} 2(t_d - o_d) (\partial(t_d - o_d) / \partial w_i) = \\ &= \sum_{d \in D} 2(t_d - o_d) (-\partial o_d / \partial w_i) = -\sum_{d \in D} 2(t_d - o_d) (\partial(w_0 + \dots + w_i x_{id} + \dots + w_n x_{nd}) / \partial w_i) = \\ &= -\sum_{d \in D} 2(t_d - o_d) (x_{id}) \\ &\Rightarrow \Delta w_i = \alpha \sum_{d \in D} 2(t_d - o_d) x_{id} \end{aligned}$$

- (3) **Question:** Prove that the LMS training rule performs a gradient descent to minimize the cost/error function  $E$  defined in (2).

**Solution:** Given the target function representation

$$o_d = w_0 + w_1 x_1 + \dots + w_n x_n,$$

LMS training rule is a learning algorithm for choosing the set of weights  $w_i$  to best fit the set of training examples  $\{ \langle d, t_d \rangle \}$ , i.e., to minimize the squared error  $E = \sum_{d \in D} (t_d - o_d)^2$ .

LMS training rule works as follows:

( $\forall \langle d, t_d \rangle$ ) use the current weights  $w_i$  to calculate  $o_d$

$$(\forall w_i) w_i \leftarrow w_i + \eta (t_d - o_d) x_{id} \quad (*)$$

$$\text{From (2)} \rightarrow (\partial E / \partial w_i) = -\sum_{d \in D} 2(t_d - o_d)x_{id} \rightarrow -(1/2)x_{id}(\partial E / \partial w_i) = (t_d - o_d)$$

$$\text{Substitute this in (*)} \rightarrow (\forall w_i) w_i \leftarrow w_i + (\eta/2)(-\partial E / \partial w_i)$$

This shows that LMS alters weights in the very same proportion as does the gradient descent algorithm (i.e.,  $-\partial E / \partial w_i$ ), proving that LMS performs gradient descent.

- (4) **Question:** Consider the following set of training examples:

Instance	Classification	a1	a2
1	+	T	T
2	+	T	T
3	-	T	F
4	+	F	F
5	-	F	T
6	-	F	T

What is the information gain of  $a_2$  relative to these training examples? Provide the equation for calculating the information gain as well as the intermediate results.

**Solution:**

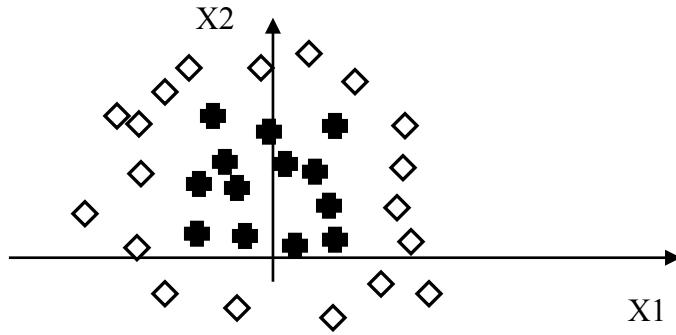
$$\text{Entropy } E(S) = E([3+, 3-]) = -(3/6) \log_2 (3/6) - (3/6) \log_2 (3/6) = 1.$$

$$\text{Gain } (S, a_2) = E(S) - (4/6)E(T) - (2/6)E(F) = 1 - 4/6 - 2/6 \approx 0.$$

$$E(T) = E([2+, 2-]) = 1.$$

$$E(F) = E([1+, 1-]) = 1.$$

- (5) **Question:** Suppose that we want to build a neural network that classifies two dimensional data (i.e.,  $X = [x_1, x_2]$ ) into two classes: diamonds and crosses. We have a set of training data that is plotted as follows:

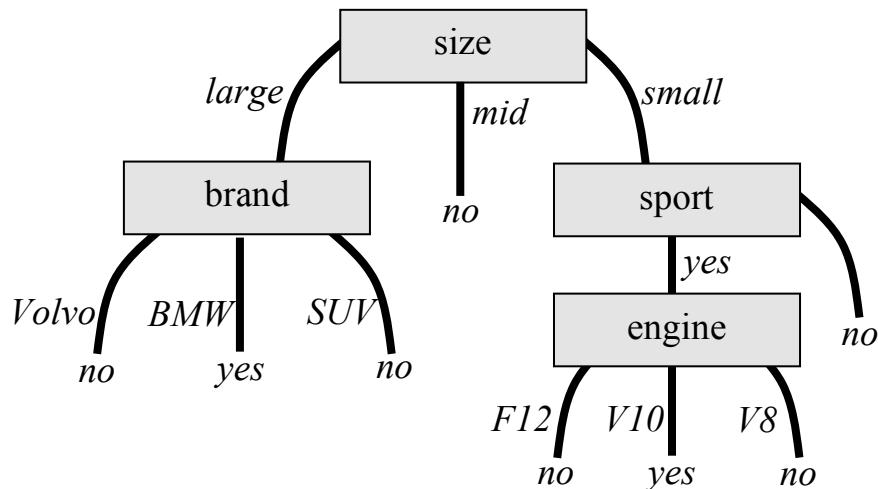


Draw a network that can solve this classification problem. Justify your choice of the number of nodes and the architecture. Draw the decision boundary that your network can find on the diagram.

**Solution:**

A solution is a multilayer FFNN with 2 inputs, one hidden layer with 4 neurons and 1 output layer with 1 neuron. The network should be fully connected, that is there should be connections between all nodes in one layer with all the nodes in the previous (and next) layer. We have to use two inputs because the input data is two dimensional. We use an output layer with one neuron because we have 2 classes. One hidden layer is enough because there is a single compact region that contains the data from the crosses-class and does not contain data from the diamonds-class. This region can have 4 lines as borders, therefore it suffices if there are 4 neurons at the hidden layer. The 4 neurons in the hidden layer describe 4 separating lines and the neuron at the output layer describes the square that is contained between these 4 lines.

- (6) **Question:** Suppose that we want to solve the problem of finding out what a good car is by using genetic algorithms. Suppose further that the solution to the problem can be represented by a decision tree as follows:



What is the appropriate chromosome design for the given problem? Which Genetic Algorithm parameters need to be defined? What would be the suitable values of those parameters for the given problem? Provide a short explanation for each.

What is the result of applying a single round of the prototypical Genetic Algorithm? Explain your answer in a clear and compact manner by providing the pseudo code of the algorithm.

**Solution:**

size = {large, mid, small} → 100, 010, 001, 011, ..., 111, 000  
brand = {Volvo, BMW, SUV} → 100, 010, 001, 011, ..., 111, 000  
sport = {yes, no} → 10, 01, 11, 00  
engine = {F12, V12, V8} → 100, 010, 001, 011, ..., 111, 000  
GoodCar = {yes, no} → 10, 01, 11, 00

→ chromosome design:

size	brand	sport	engine	GoodCar
100	100	11	111	01

Fitness function for the given problem can be defined as a Sigmoid function  $f(x) = 1 / (1 + e^{-x})$ , where  $x$  is the percentage of all training examples correctly classified by a specific solution (chromosome).

Selection method – e.g., rank selection method can be used;

Crossover technique – 2-point crossover can be used for the given problem with a crossover mask 1111110000011; the reason is that either size + brand or sport + engine define the solution

Crossover rate – usually  $k = 60\%$

Mutation rate – usually 1%

Termination condition – e.g., all training examples are correctly classified

GA pseudo code:

Step 1: Choose initial population.

Step 2: Evaluate the fitness of individuals in the population.

Step 3: Select  $k$  individuals to reproduce; breed new generation through crossover and mutation; evaluate the individual fitness of offspring; replace  $k$  worse ranked part of population with offspring.

Step 4: Repeat step 3 until the termination condition is reached.

s1/2: 1010101001010, 101111110101, 010001111101, 0011111001010,  
101101110101

s3: 1010101110110 (fit 1), 0011111001001 (fit 0), 1010101111110 (fit 2),  
1011011001001 (fit 1), 0011111110101 (fit 2), 1011011110101 (fit 3)

result: 101111110101, 101101110101, 1011011110101, 0011111110101,  
1010101111110

# FINAL: CS 6375 (Machine Learning) Fall 2014

The exam is closed book. You are allowed a one-page cheat sheet. Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, use an additional sheet (available from the instructor) and staple it to your exam.

- NAME \_\_\_\_\_

- UTD-ID if known \_\_\_\_\_

Question	Points	Score
Decision Trees	10	
Naive Bayes and the EM algorithm	10	
Linear Methods	10	
SVMs and Kernels	10	
Ensemble Methods	10	
Neural Networks	10	
Nearest Neighbors and K-means	10	
Learning Theory	10	
Bayesian networks	20	
Total:	100	



**Question 1: Decision Trees (10 points)**

Consider the training dataset given below. In the dataset,  $X_1$ ,  $X_2$ , and  $X_3$  are the attributes and  $Y$  is the class variable.

Example#	$X_1$	$X_2$	$X_3$	$Y$
E1	0	0	0	+ve
E2	0	0	1	-ve
E3	0	1	0	-ve
E4	0	1	1	+ve
E5	1	0	0	-ve

(a) (3 points) Which attribute has the highest information gain? Justify your answer?

(b) (3 points) Draw the (full) decision tree for this dataset using the information gain criteria.

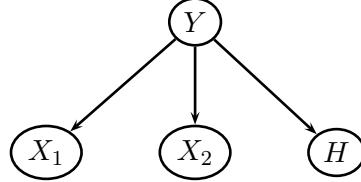
Suppose that we have the following validation dataset:

Example#	$X_1$	$X_2$	$X_3$	$Y$
E6	1	1	0	+ve
E7	0	1	1	-ve
E8	1	1	1	-ve

- (c) (4 points) Can you construct a decision tree that will have 100% accuracy on the validation dataset as well as the training dataset. Answer Yes or No. If your answer is yes, then draw the decision tree that will have 100% accuracy. If your answer is no, then explain why it cannot be done. No credit will be given if the explanation is incorrect.

**Question 2: Naive Bayes and the EM algorithm (10 points)**

In this question, we will develop EM algorithm for Naive Bayes with hidden variables. Consider the Naive Bayes model given below.  $X_1$  and  $X_2$  are always observed and  $H$  is a hidden variable.  $Y$  is the class variable.



Assume that all variables are binary and all parameters are initialized as follows:

Parameter	$P(Y)$	$P(X_1 Y)$	$P(X_1 \bar{Y})$	$P(X_2 Y)$	$P(X_2 \bar{Y})$	$P(H Y)$	$P(H \bar{Y})$
Initial Value	$\theta$	$\alpha_1$	$\alpha_0$	$\beta_1$	$\beta_0$	$\gamma_1$	$\gamma_0$

We are given a dataset having 1000 examples and is such that all tuples in the truth-table over  $X_1$ ,  $X_2$  and  $Y$  occur exactly 125 times in it. Thus, for example  $\#(X_1, X_2, Y) = 125$ ;  $\#(\bar{X}_1, \bar{X}_2, \bar{Y}) = 125$ ; etc.

- (a) (7 points) After running the EM algorithm until convergence, what will be the values of the parameters  $\theta$ ,  $\alpha_1$ ,  $\alpha_0$ ,  $\beta_1$ ,  $\beta_0$ ,  $\gamma_1$  and  $\gamma_0$ . Explain your answer. No credit without correct explanation. To figure out the parameters at convergence, try running EM for 1-2 iterations and the pattern will be clear to you.

- (b) (3 points) Your Boss claims that those hidden variables in the Naive Bayes model, assuming that they are children of the class variable, are useless. Is your Boss correct? Explain your answer.

### Question 3: Linear Methods (10 points)

- (a) (2 points) Does a 2-class Gaussian Naive Bayes classifier with parameters  $\mu_{1,k}$ ,  $\sigma_{1,k}$ ,  $\mu_{2,k}$  and  $\sigma_{2,k}$  for attributes  $k = 1, \dots, m$  have exactly the same representational power as logistic regression, given no assumptions about the variance values  $\sigma_{1,k}^2$  and  $\sigma_{2,k}^2$ ? True or False. Explain your answer (no credit if the explanation is incorrect).

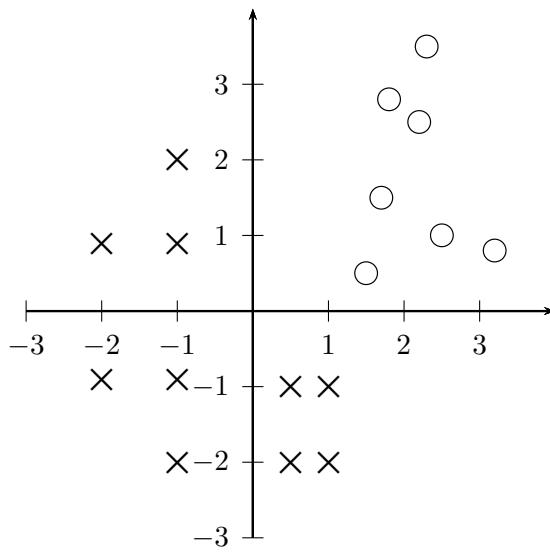
- (b) (8 points) Consider the 2-D dataset given below. Suppose we are going to use logistic regression to train a classifier on this dataset and let us further assume that we are going to apply regularization to all the weights. Formally, we will train a classifier that maximizes the following expression:

$$\max \sum_{i=1}^n \log(P(y^{(i)}|x_1^{(i)}, x_2^{(i)})) - (\lambda_0 w_0^2 + \lambda_1 w_1^2 + \lambda_2 w_2^2)$$

where  $i$  indexes the  $i$ -th datapoint given by the three tuple  $(x_1^{(i)}, x_2^{(i)}, y^{(i)})$ .  $x_1$  and  $x_2$  are the features and  $y$  is the class variable. Recall that  $P(y|x_1, x_2)$  is given by:

$$P(y|x_1, x_2) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

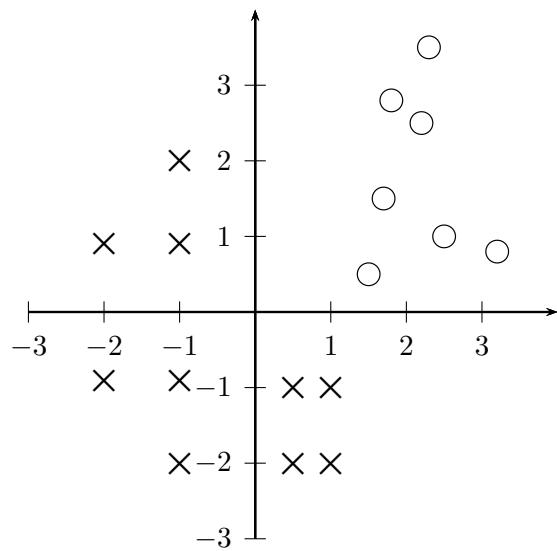
Dataset:



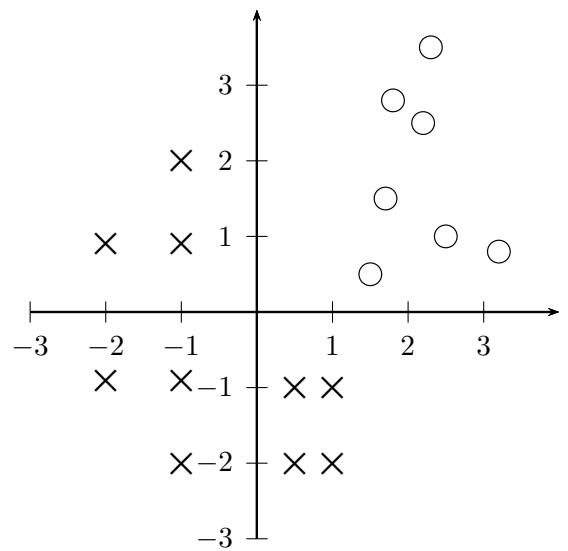
(Question continues to the next page.)

Draw the decision boundary for each of the following cases:

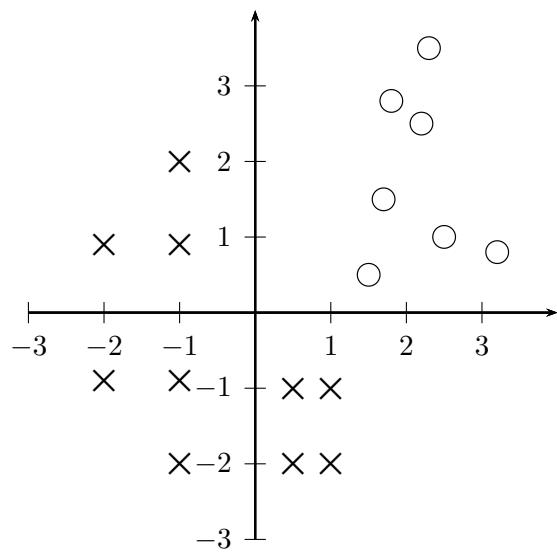
Case 1:  $\lambda_0 = 0$ ,  $\lambda_1 = 0$  and  $\lambda_2 = 0$



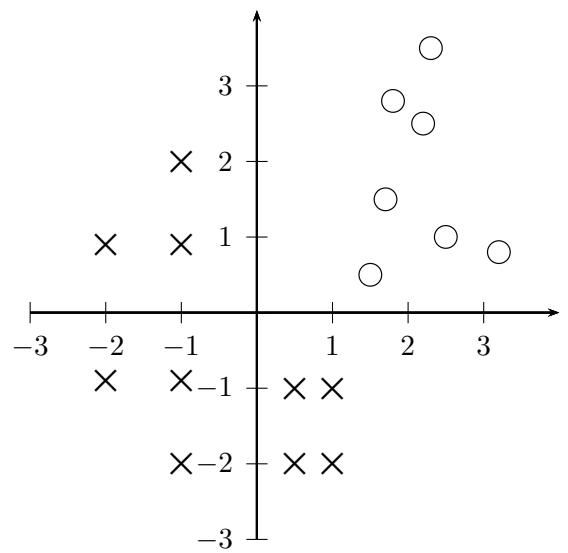
Case 2:  $\lambda_0 = \infty$ ,  $\lambda_1 = 0$  and  $\lambda_2 = 0$



Case 3:  $\lambda_0 = 0$ ,  $\lambda_1 = \infty$  and  $\lambda_2 = 0$

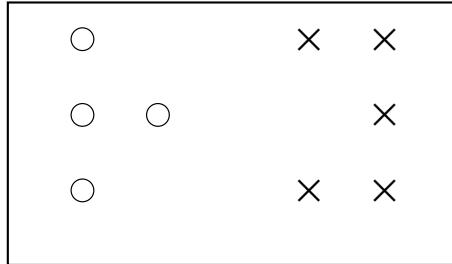


Case 4:  $\lambda_0 = 0$ ,  $\lambda_1 = 0$  and  $\lambda_2 = \infty$



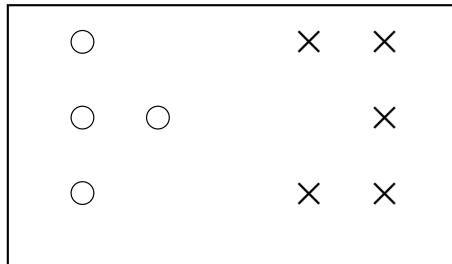
**Question 4: SVMs and Kernels (10 points)**

- (a) (3 points) In the following image, circle the points such that removing that example from the training set and retraining linear SVM, we would get a different decision boundary than training on the full sample. Also, give a one or two sentence explanation for why you circled the points.



**Explanation:**

- (b) (3 points) Suppose you used a perceptron instead of linear SVMs. Circle the points such that removing that example from the training set and running a perceptron, we **may** get a different decision boundary than training with perceptron on the full sample. Note the word “may” and therefore you should take the worst case view. Also, give a one or two sentence explanation for why you circled the points.



**Explanation:**

(c) (4 points) You are trying to use SVMs to build a classifier for a dataset that your boss gave you. In the dataset, there are only a few positive training examples and you are certain that they are always classified correctly by your SVM classifier. The dataset also has a large number of negative training examples and your boss tells you that it is OK if we misclassify some of them. Modify the basic dual form of the SVM optimization problem given below:

$$\text{maximize} \quad \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{subject to } \alpha_i \geq 0, \text{ and } \sum_{i=1}^n \alpha_i y_i = 0$$

to better solve this type of problem. Namely, you would like to ensure that you won't misclassify any of the positive examples but could misclassify some of the negative examples. Introduce additional parameter(s) (or constants for the purpose of solving the quadratic programming problem) in order to achieve this.

In your solution, please use  $I+$  to index positively labeled examples ( $y_i = +1$ ) and  $I-$  for negative examples ( $y_i = -1$ ). In other words,  $i \in I+$  means that  $y_i = +1$  and  $i \in I-$  means that  $y_i = -1$ .

Write down your solution using the dual formulation.

**Question 5: Ensemble Methods (10 points)**

For your convenience, AdaBoost is given on the last page.

- (a) (1 point) Circle the method or methods that are slower in terms of training time than the decision tree classifier. More than one answer is possible. No explanation needed.

BAGGING      BOOSTING

- (b) (1 point) Circle the method or methods that can be parallelized easily (after all we live in the age of multi-core machines). More than one answer is possible. No explanation needed.

BAGGING      BOOSTING

- (c) (1 point) Circle the method or methods that attempt to reduce the variance. More than one answer is possible. No explanation needed.

BAGGING      BOOSTING

- (d) (1 point) Circle the method or methods that hurt the performance of stable classifiers.

To recap, a stable classifier is a classifier whose decision boundary does not change much as we add new data points. More than one answer is possible. No explanation needed.

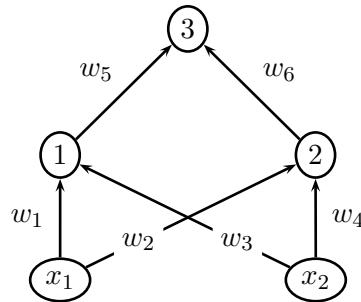
BAGGING      BOOSTING

- (e) (3 points) Answer True or False. Explain your answer. A weak learner with less than 50% accuracy does not present any problem to the Adaboost algorithm.

- (f) (3 points) True or False. AdaBoost is not susceptible to outliers. If your answer is true, explain why. If your answer is false, then describe a simple heuristic to fix Adaboost so that it is not susceptible to outliers.

**Question 6: Neural Networks (10 points)**

Consider the Neural network given below.



**Assume that all internal nodes and output nodes compute the  $\tanh$  function.** In this question, we will derive an explicit expression that shows how back propagation (applied to minimize the least squares error function) changes the values of  $w_1, w_2, w_3, w_4, w_5$  and  $w_6$  when the algorithm is given the example  $(x_1, x_2, y)$  with  $y$  being class variable (there are no bias terms). Assume that the learning rate is  $\eta$ . Let  $o_1$  and  $o_2$  be the output of the hidden units 1 and 2 respectively. Let  $o_3$  be the output of the output unit 3.

Hint: Derivative of  $\tanh(x) = 1 - \tanh^2(x)$ .

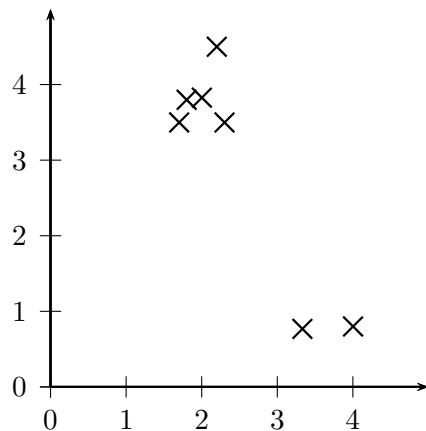
(a) (3 points) Forward propagation. Write equations for  $o_1$ ,  $o_2$  and  $o_3$ .

(b) (3 points) Backward propagation. Write equations for  $\delta_1$ ,  $\delta_2$  and  $\delta_3$  where  $\delta_1$ ,  $\delta_2$  and  $\delta_3$  are the values propagated backwards by the units denoted by 1, 2 and 3 respectively in the neural network.

- (c) (4 points) Give an explicit expression for the new (updated) weights  $w_1, w_2, w_3, w_4, w_5$  and  $w_6$  after backward propagation.

**Question 7: Nearest Neighbors and K-means (10 points)**

- (a) (3 points) As we increase  $k$ , the  $k$  nearest neighbor algorithm begins to overfit the training dataset. True or False. Explain your answer.
- (b) (3 points) What is the time complexity of the  $K$ -means algorithm. Use the  $O$ -notation. Explain your notation clearly.



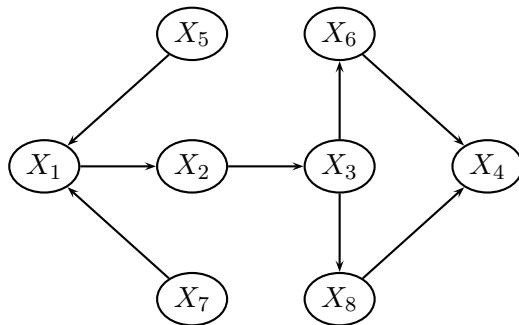
- (c) (4 points) Draw the agglomerative clustering tree for the dataset given above. You must use single link clustering.

**Question 8: Learning Theory (10 points)**

- (a) (5 points) Prove that the VC dimension of the concept class of all axis parallel rectangles is greater than or equal to 4.
- (b) (5 points) Consider the class of concepts that can be expressed as an axis parallel rectangle. Assume further that the rectangles can be defined in terms of 4 integers:  $a, b, c$ , and  $d$  that lie between 1 and 100 (both included). Namely,  $a, b, c, d \in \{1, \dots, 100\}$ . We assume that a consistent algorithm is available.
- i. What is the size of the hypothesis space.
  - ii. How many randomly obtained examples are needed for PAC learning with accuracy parameter  $\epsilon = 0.1$  and confidence parameter  $\delta = 0.05$ ?
  - iii. What is the accuracy level (what is  $\epsilon$ ) if it is known that 1000 randomly chosen examples were used, and the desired confidence level is  $\delta = 0.05$ ?

**Question 9: Bayesian networks (20 points)**

Consider the Bayesian network given below:



- (a) (7 points) Let  $X_4$  be evidence. Consider the elimination order  $(X_1, X_2, X_3, X_5, X_6, X_7, X_8)$ . Write down the factors that you will encounter during this elimination for computing  $P(X_4 = x_4)$  (namely, trace the variable elimination algorithm). What is the time and space complexity along this ordering.

(b) (7 points) Let  $X_4$  be evidence for the Bayesian network given on the previous page. Consider the elimination order  $(X_5, X_7, X_1, X_2, X_3, X_6, X_8)$ . Write down the factors that you will encounter during this elimination for computing  $P(X_4 = x_4)$ . What is the time and space complexity along this ordering.

(c) (6 points) Suppose we want to learn a Bayesian network over two binary variables  $X_1$  and  $X_2$ . You have  $n$  training examples  $\{(x_1^{(1)}, x_2^{(1)}), \dots, (x_1^{(n)}, x_2^{(n)})\}$ . Let  $BN1$  denote the Bayesian network having no edges and  $BN2$  denote the Bayesian network having an edge from  $X_1$  to  $X_2$ . Suppose you are learning both networks using the maximum likelihood estimation approach.

i. Describe a situation in which you will prefer  $BN1$  over  $BN2$  for modeling this data?

ii. Describe a situation in which you will prefer  $BN2$  over  $BN1$  for modeling this data?

## Entropy

```
Entropy(0,0) = 0.0
Entropy(0,1) = 0.0
Entropy(0,2) = 0.0
Entropy(0,3) = 0.0
Entropy(0,4) = 0.0
Entropy(1,1) = 1.0
Entropy(1,2) = 0.918295834054
Entropy(1,3) = 0.811278124459
Entropy(1,4) = 0.721928094887
Entropy(2,2) = 1.0
Entropy(2,3) = 0.970950594455
Entropy(2,4) = 0.918295834054
Entropy(3,3) = 1.0
Entropy(3,4) = 0.985228136034
Entropy(4,4) = 1.0
```

## Useful formulas for Learning Theory

$$m \geq \frac{1}{\epsilon} (\ln(1/\delta) + \ln(|H|))$$

$$m \geq \frac{1}{2\epsilon^2} (\ln(1/\delta) + \ln(|H|))$$

$$m \geq \frac{1}{\epsilon} (4 \log_2(2/\delta) + 8VC(H) \log_2(13/\epsilon))$$

## AdaBoost

1. Initialize the data weighting coefficients  $\{w_n\}$  by setting  $w_n^{(1)} = 1/N$  for  $n = 1, \dots, N$ .
2. For  $m = 1, \dots, M$ :
  - (a) Fit a classifier  $y_m(\mathbf{x})$  to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) \quad (14.15)$$

where  $I(y_m(\mathbf{x}_n) \neq t_n)$  is the indicator function and equals 1 when  $y_m(\mathbf{x}_n) \neq t_n$  and 0 otherwise.

- (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^N w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^N w_n^{(m)}} \quad (14.16)$$

and then use these to evaluate

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}. \quad (14.17)$$

- (c) Update the data weighting coefficients

$$w_n^{(m+1)} = w_n^{(m)} \exp \{ \alpha_m I(y_m(\mathbf{x}_n) \neq t_n) \} \quad (14.18)$$

3. Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \text{sign} \left( \sum_{m=1}^M \alpha_m y_m(\mathbf{x}) \right). \quad (14.19)$$

Name: \_\_\_\_\_ Andrew ID: \_\_\_\_\_

## Instructions

- Anything on paper is OK in arbitrary shape size, and quantity.
- Electronic devices are not acceptable. This includes iPods, iPads, Android tablets, Blackberries, Nokias, Windows phones, Microsoft Surface, laptops, Chromebooks, MP3 players, digital cameras, Google Glass, Android Wear, or anything else that requires electricity.<sup>1</sup>
- **If we catch you cheating or using any such device, the exam is over for you. Your results will not count and you will have failed this exam automatically. There might be more serious consequences, too. No exceptions. Switch off your phones before the exam.**
- There are more questions in the exam than what you are likely to be able to solve in 90 minutes time. Choose wisely and answer those first that you believe you can solve easily. This is an opportunity.
- None of the questions should require more than 1 page to answer. Its OK to be concise if you address the key points of the derivation (it should be human-readable, though).

## Point Distribution

Problem	Points
1	/10
2	/10
3	/15
4	/10
5	/15
6	/10
7	/15
Total	/85

---

<sup>1</sup>Obvious exceptions for pacemakers and hearing aids.

## 1 Loss, Regularization and Optimization [10 points]

### 1.1 Quick Questions [4 points]

Explain in **one or two sentences** why the statements are true (or false).

1. L2 loss is more robust to outliers than L1 loss.

**Solution:**

False

The gradient of L2 loss can grow without bound whereas the L1 loss gradient is bounded, hence the influence of an outlier is limited.

2. Logistic loss is better than L2 loss in classification tasks.

**Solution:**

True

With logistic loss, correctly classified points that are far away from the decision boundary have much less impact on the decision boundary

3. In terms of feature selection, L2 regularization is preferred since it comes up with sparse solutions.

**Solution:**

False

L1 regularization (LASSO) comes up with sparse solutions due to nonvanishing gradient at 0.

### 1.2 Gradient Descent [6 points]

Denote by  $x \in \mathbb{R}^d$  data, by  $w \in \mathbb{R}^d$  the weight vector, by  $y \in \mathbb{R}$  labels, by  $\lambda > 0$  a regularization constant, and by  $m$  the total number of data points. Let  $R(w)$  be the regularized risk function:

$$R(w) := \frac{1}{m} \sum_{i=1}^m l(y_i - \langle w, x_i \rangle) + \frac{\lambda}{2} \|w\|^2 \text{ where } l(\xi) = \begin{cases} \frac{1}{2}\xi^2 & \text{if } |\xi| < 1 \\ |\xi| - \frac{1}{2} & \text{otherwise} \end{cases}$$

1. Calculate the batch gradient of  $R(w)$  with respect to  $w$ .

**Solution:**

Define  $g_i$  as the gradient of  $i$ th data point.

$$g_i = \begin{cases} (\langle w, x_i \rangle - y)x_i & \text{if } |\langle w, x_i \rangle - y| < 1 \\ \text{sgn}(\langle w, x_i \rangle - y)x_i & \text{otherwise} \end{cases}$$

$$\frac{\partial R}{\partial w} = \frac{1}{m} \sum_{i=1}^m g_i + \lambda w$$

2. Write out a stochastic gradient descent learning algorithm for minimizing  $R(w)$

**Solution:**

Randomly select a training instance  $x_i$ , update  $w$  as follows:

$$w \leftarrow (1 - \lambda \eta_t)w - \eta_t g_i$$

3. Name two common choices for choosing the learning rate and briefly explain their properties.

**Solution:**

$O(\frac{1}{\sqrt{t}})$ ,  $O(\frac{1}{t})$  for strong convexity, Polynomial decay, AdaGrad, etc.

## 2 Neural Networks

### 2.1 Quick Questions [3 points]

Provide a **brief** answer to the following questions (1-2 sentences).

1. State one advantage of linear rectified activation compared to logistic sigmoid activation.

**Solution:**

Stable gradient, sparse activation, easy computation, etc.

2. Does it make sense to initialize all weights in a deep network to 0.

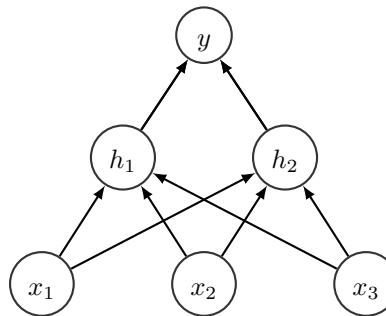
**Solution:**

No.

Symmetry breaking: If all weights are equal, their gradients will be the same as well, and the gradients will possibly vanish. Hence all neurons will learn the same feature.

### 2.2 Forward and Backward Propagation [7 points]

The following graph shows the structure of a simple neural network with a single hidden layer. The input layer consists of three dimensions  $x = (x_1, x_2, x_3)$ . The hidden layer includes two units  $h = (h_1, h_2)$ . The output layer includes one unit  $y$ . We ignore bias terms for simplicity.



We use linear rectified units  $\sigma(z) = \max(0, z)$  as activation function for the hidden and the output layer. Moreover, denote by  $l(y, t) = \frac{1}{2}(y - t)^2$  the loss function. Here  $t$  is the target value for the output unit  $y$ .

Denote by  $W$  and  $V$  weight matrices connecting input and hidden layer, and hidden layer and output respectively. They are initialized as follows:

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \text{ and } V = \begin{bmatrix} 0 & 1 \end{bmatrix} \text{ and } x = [1, 2, 1] \text{ and } t = 1.$$

Also assume that we have at least one sample  $(x, t)$  given by the values above.

1. Write out *symbolically* (no need to plug in the specific values of  $W$  and  $V$  just yet) the mapping  $x \rightarrow y$  using  $\sigma, W, V$ .

**Solution:**

$$y = \sigma(V\sigma(Wx))$$

2. Assume that the current input is  $x = (1, 2, 1)$ . The target value is  $t = 1$ . Compute the *numerical* output value  $y$ , clearly show all intermediate steps. You can reuse the results of the previous question. Using

matrix form is recommended but not mandatory.

**Solution:**

$$h = \sigma(Wx) = (2, 0)^\top$$

$$y = \sigma(Vh) = 0$$

3. Compute the gradient of the loss function with respect to the weights. In particular, compute the following terms *symbolically*:

- The gradient relative to  $V$ , i.e.  $\frac{\partial l}{\partial V}$
- The gradient relative to  $W$ , i.e.  $\frac{\partial l}{\partial W}$
- Compute the values *numerically* for the choices of  $W, V, x, y$  given above.

**Solution:**

The gradients are computed with the chain rule as follows:

$$\frac{\partial l}{\partial V} = \left( \frac{\partial y}{\partial V} \right)^\top \frac{\partial l}{\partial y}$$

$$\frac{\partial l}{\partial W} = \left( \frac{\partial h}{\partial W} \right)^\top \left( \frac{\partial y}{\partial h} \right)^\top \frac{\partial l}{\partial y}$$

Plugging in numerical values yields

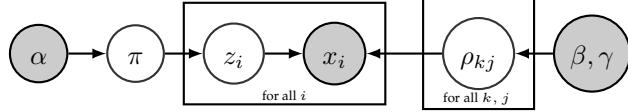
$$\frac{\partial l}{\partial V} = \left( \frac{\partial Vh}{\partial V} \right)^\top \left( \frac{\partial y}{\partial Vh} \right)^\top \frac{\partial l}{\partial y} = h^\top g(y - t) = [-2g, 0]$$

where  $0 \leq g \leq 1$  is the subgradient of ReLU

$$\frac{\partial l}{\partial W} = \left( \frac{\partial Wx}{\partial W} \right)^\top \left( \frac{\partial h}{\partial Wx} \right)^\top \left( \frac{\partial y}{\partial h} \right)^\top \frac{\partial l}{\partial y} = MV^\top (y - t)x^\top = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ where } M = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

### 3 Expectation Maximization for Clustering Bit Vectors [15 points]

Assume that we observe binary vectors  $x_i \in \{0, 1\}^d$ , e.g.  $x_i = \boxed{0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1}$ . Our goal is to find a generative model that represents these vectors. A mixture of binary distributions might be a good idea. In particular, we pick the following graphical model:



The associated generative model is given as follows:

1. For each topic  $k \in \{1, \dots, K\}$  and for each coordinate  $j \in \{1, \dots, d\}$ 
  - (a) Draw  $\rho_{kj} \sim \text{Beta}(\beta, \gamma)$
2. Draw mixture weights  $\pi \sim \text{Dirichlet}(\alpha)$
3. For each observation  $i \in \{1, \dots, N\}$ 
  - (a) Draw a component index  $z_i \sim \text{Categorical}(\pi)$
  - (b) Draw each coordinate  $x_{ij} \sim \text{Binomial}(\rho_{z_{ij}})$

#### 3.1 Generative Model

Write the joint probability  $p(\pi, z, x, \rho | \alpha, \beta, \gamma)$  for a set of  $n$  vectors  $\{x_1, \dots, x_n\}$  drawn from this distribution. Hint, the Beta and Dirichlet distributions are given by

$$\text{Dirichlet}(\pi | \alpha) = \frac{\Gamma(k\alpha)}{\Gamma^k(\alpha)} \prod_{k=1}^K \pi_k^{\alpha-1} \text{ and } \text{Beta}(\rho | \beta, \gamma) = \frac{\Gamma(\beta + \gamma)}{\Gamma(\beta)\Gamma(\gamma)} \rho^{\beta-1} (1 - \rho)^{\gamma-1}$$

Solution:

$$\begin{aligned} p(\pi, z, x, \rho | \alpha, \beta, \gamma) &= p(\pi | \alpha) \prod_{i=1}^N p(z_i | \pi) \prod_{j=1}^d p(x_{ij} | \rho_{z_{ij}}) \prod_{k=1}^K p(\rho_{kj} | \beta, \gamma) \\ &= \frac{\Gamma(\beta + \gamma)}{\Gamma(\beta)\Gamma(\gamma)} \frac{\Gamma(k\alpha)}{\Gamma^k(\alpha)} \prod_{k=1}^K \pi_k^{\alpha-1} \prod_{i=1}^N \pi_{z_i} \prod_{j=1}^d \rho_{z_{ij}}^{x_{ij}} (1 - \rho_{z_{ij}})^{1-x_{ij}} \prod_{k=1}^K \rho_{kj}^{\beta-1} (1 - \rho_{kj})^{\gamma-1} \\ &= \frac{\Gamma(\beta + \gamma)}{\Gamma(\beta)\Gamma(\gamma)} \frac{\Gamma(k\alpha)}{\Gamma^k(\alpha)} \prod_{k=1}^K \pi_k^{n_k + \alpha - 1} \prod_{j=1}^d \rho_{kj}^{\beta-1 + m_{kj}} (1 - \rho_{kj})^{\gamma-1 + n_k - m_{kj}} \end{aligned}$$

Last step follows if we define  $n_k = \sum_{i=1}^N \delta_{(z_i=k)}$ ,  $m_{kj} = \sum_{i=1}^N \delta_{(z_i=k)} x_{ij}$ .

#### 3.2 Inference

1. Explain why it is nontrivial to infer the posterior  $\pi, z, \rho | \alpha, \beta, \gamma, x$  directly.
2. Suggest at least one alternative approach to Expectation Maximization for solving this problem.

Solution:

3.2.1: There is no closed form solution  $p(x)$  which is the numerical normalizer.

3.2.2. Gibbs Sampling or any other MCMC methods.

### 3.3 Expectation Maximization Algorithm

Use the variational inequality

$$\log p(a) \geq \mathbf{E}_{b \sim q(b)} [\log p(a, b)] + H[q]$$

to derive a lower bound on  $p(x, \pi, \rho | \alpha, \beta, \gamma)$ . Hint — you need to make an approximation for the distribution over  $z$  when obtaining the variational distribution.

**Solution:**

$$\begin{aligned} a &= \rho, \pi, b = z \\ \Rightarrow \log p(x, \pi, \rho | \alpha, \beta, \gamma) &= \log \sum_z p(x, \pi, z, \rho | \alpha, \beta, \gamma) \\ &\geq \mathbf{E}_{z \sim q(z)} [\log p(\pi, z, \rho)] + H[q] \\ &= \mathbf{E}_{z \sim q(z)} \left[ \sum_k^K (n_k + \alpha - 1) \log \pi_k + \sum_{j=1}^d (\beta - 1 + m_{kj}) \log \rho_{kj} + (\gamma - 1 + n_k - m_{kj}) \log(1 - \rho_{kj}) \right] \\ &\quad + H[q] + \text{constant} \end{aligned}$$

### 3.4 Expectation Step

Compute  $p(z_i = k | x_i, \pi, \rho)$ .

**Solution:**

$$\begin{aligned} p(z_i = k | x_i, \pi, \rho) &= \frac{p(z_i = k | \pi) \prod_{j=1}^d p(x_{ij} | \rho_{kj})}{\sum_{k=1}^K p(z_i = k | \pi) \prod_{j=1}^d p(x_{ij} | \rho_{kj})} \\ &\propto \pi_k \prod_{j=1}^d \rho_{kj}^{x_{ij}} (1 - \rho_{kj})^{1 - x_{ij}} \end{aligned}$$

### 3.5 Maximization step

Using  $q_i(k) = p(z_i = k | x_i, \pi, \rho)$  compute the estimates for  $\pi$  and  $\rho$ . Hint — if you didn't manage to derive the explicit expression for  $q_i(z_i)$ , you may plug in the term symbolically and give the update in terms of the variational distribution  $q$  only.

**Solution:**

- $\pi$ :

$$\hat{\pi}_k = \frac{\sum_i q_i(k) + \alpha - 1}{\sum_{i,k} q_i(k) + K(\alpha - 1)}.$$

- $\rho$ : Recall from 3.3 that

$$\begin{aligned} \log p(x, \pi, \rho | \alpha, \beta, \gamma) \geq & \mathbf{E}_{z \sim q(z)} \left[ \sum_k^K (n_k + \alpha - 1) \log \pi_k + \sum_{j=1}^d (\beta - 1 + m_{kj}) \log \rho_{kj} + (\gamma - 1 + n_k - m_{kj}) \log(1 - \rho_{kj}) \right] \\ & + H[q] + \text{constant} \end{aligned}$$

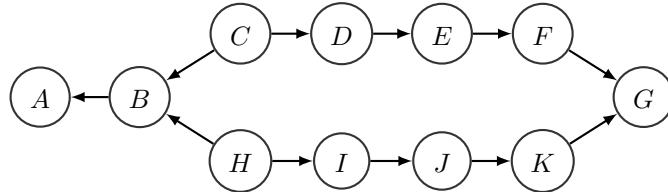
Maximizing this lower bound leads to

$$\rho_{kj} = \frac{\beta + \sum_i q_i(k) x_{ij} - 1}{\beta + \gamma + \sum_i q_i(k) - 2}$$

## 4 Graphical Models [10 points]

### 4.1 Independence [5 points]

Consider the following graphical model:



Check whether the following independence holds or not? Provide a brief reason for your answer. Alternatively, give a path for the Bayes ball algorithm as illustration.

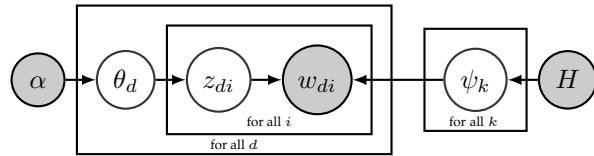
- $J \perp E|G$
- $J \perp E|K, G$
- $J \perp E|B$
- $J \perp E|A$
- $J \perp E|A, H$

**Solution:**

- $J \perp E|G$ : No, explaining away at I
- $J \perp E|K, G$ : Yes, Interaction block
- $J \perp E|B$ : No, explaining away
- $J \perp E|A$ : No, flow of influence
- $J \perp E|H, A$ : Yes, observation block

## 4.2 Relevant Random Variables [5 points]

Consider the following graphical model:



1. For the given graphical model as it is, denote the minimal set of random variables which affect  $z_{di} = k$  for a given  $d, i$  pair?
2. Suppose we integrate out the random variables  $\theta_d$  and  $\psi_k$ . Which nodes affect  $z_{di} = k$  for a given  $d, i$  pair now?

**Solution:**

- Depends on  $w_{di}, \theta_{dk}$  and  $\psi_k$ .
- Depends on  $\{w_{d'i'} : z_{d'i'} = k\}$ .

Markov blanket!

## 5 Regression with Gaussian Prior [15 points]

Assume that we have some parameter  $w \sim \mathcal{N}(w_0, \sigma^2 \mathbf{1})$  drawn from a  $d$ -dimensional Normal distribution. Moreover, assume that we perform regression with additive Normal noise  $\xi$ . That is, assume that we have

$$y = \langle w, \phi(x) \rangle + \xi \text{ where } \xi \sim \mathcal{N}(0, \tau^2) \text{ and } w \sim \mathcal{N}(w_0, \sigma^2 \mathbf{1}) \text{ and } \phi(x) \in \mathbb{R}^d.$$

### 5.1 Gaussian Process

Prove that  $y|x$  is drawn from a Gaussian Process. That is, show that  $y|x$  is normal with a suitable mean function  $\mu(x)$  and covariance kernel  $k(x, x')$ .

**Solution:**

$y$  is a linear combination of  $w$  and  $\xi$ , each of which are Normal.

$$\begin{aligned} \mu(x) &= \mathbb{E}[y] = \mathbb{E}[\langle w, \phi(x) \rangle + \xi] = \langle w_0, \phi(x) \rangle + 0 = \langle w_0, \phi(x) \rangle \\ k(x, x') &= \mathbb{E}[(y - \langle w_0, \phi(x) \rangle)(y' - \langle w_0, \phi(x') \rangle)^T] \\ &= \mathbb{E}[(\langle w, \phi(x) \rangle + \xi - \langle w_0, \phi(x) \rangle)(\langle w, \phi(x') \rangle + \xi - \langle w_0, \phi(x') \rangle)^T] \\ &= \mathbb{E}[(\langle w - w_0, \phi(x) \rangle + \xi)(\langle w - w_0, \phi(x') \rangle + \xi)^T] \\ &= \mathbb{E}[\langle w - w_0, \phi(x) \rangle \langle w - w_0, \phi(x') \rangle] + \mathbb{E}[\xi] \mathbb{E}[\langle w - w_0, \phi(x) \rangle] + \mathbb{E}[\xi] \mathbb{E}[\langle w - w_0, \phi(x') \rangle] + \mathbb{E}[\xi^2] \\ &= \phi(x)^T \mathbb{E}[(w - w_0)(w - w_0)^T] \phi(x') + \tau^2 \\ &= \sigma^2 \phi(x)^T \phi(x') + \tau^2 \\ &= \sigma^2 \langle \phi(x), \phi(x') \rangle + \tau^2 \end{aligned}$$

### 5.2 Kernels

What is the difference between a kernel obtained from the feature map  $\phi(x)$  described above and a kernel such as  $k(x, x') = \exp(-\frac{1}{2} \|x - x'\|^2)$ . Can you characterize the type of functions that can be approximated? Hint: what does the rank of the kernel matrix tell you?

**Solution:**

The data point is mapped to an infinite dimensional feature space in case of  $k(x, x') = \exp(-\frac{1}{2} \|x - x'\|^2)$ . Polynomials of order  $d$ .

### 5.3 Posterior Distribution

Assume that we observe  $m$  pairs  $(x_i, y_i)$ . Show that the posterior distribution is normal.

$$p(w | (x_1, y_1), \dots, (x_m, y_m), \tau^2, \sigma^2)$$

**Solution:**

$$\begin{aligned} p(w | (x_1, y_1), \dots, (x_m, y_m), \tau^2, \sigma^2) &\propto p(w, (x_1, y_1), \dots, (x_m, y_m), \tau^2, \sigma^2) \\ &= p(w | \sigma^2) \prod_i p(y_i | x_i, w, \tau^2) \end{aligned} \tag{1}$$

Each of them is Gaussian, so overall will be Gaussian.

## 5.4 Mean and Covariance

Compute the mean and covariance of the posterior distribution  $p(w|(x_1, y_1), \dots, (x_m, y_m), \tau^2, \sigma^2)$ .

**Solution:**

$$\begin{aligned}\mathbb{E}[w|(x_1, y_1), \dots, (x_m, y_m), \tau^2, \sigma^2] &= w_0 + \sigma^2 \Phi^T (\sigma^2 \Phi \Phi^T + \tau^2 \mathbf{1})^{-1} (y - \Phi w_0) \\ \mathbf{C}_{w|(x_1, y_1), \dots, (x_m, y_m), \tau^2, \sigma^2} &= \sigma^2 \mathbf{1} - \sigma^4 \Phi (\sigma^2 \Phi \Phi^T + \tau^2 \mathbf{1})^{-1} \Phi\end{aligned}\quad (2)$$

where  $\Phi = [\phi(x_1); \phi(x_2); \dots; \phi(x_m)]$  and  $y = [y_1; y_2; \dots; y_m]^T$ .

## 5.5 Prediction

Assume that we want to estimate  $y'|x'$  using the previous observations, i.e. we regress on

$$p(y'|x', (x_1, y_1), \dots, (x_m, y_m), \tau^2, \sigma^2)$$

Derive the mode of  $p(y'|x', \text{rest})$ .

**Solution:**

Mode of the posterior predictive  $p(y'|x', \text{rest})$  is simply

$$\hat{y}' = \phi(x')^T (w_0 + \sigma^2 \Phi^T (\sigma^2 \Phi \Phi^T + \tau^2 \mathbf{1})^{-1} (y - \Phi w_0))$$

## 6 Matrix Factorization [10 points]

Recommendations can be generated by a wide range of algorithms. While user-based or item-based similarity methods are simple and intuitive, matrix factorization techniques are usually more effective because they allow us to discover the latent features underlying the interactions between users and items.

That is, for observed ratings  $r_{um}$  for a given  $(u, m)$  pair of a user  $u$  and a movie  $m$ , one typically tries to estimate the score by

$$f_{um} = \langle v_u, w_m \rangle + b_u + b_m.$$

Here  $v_u$  and  $w_m$  are vectors in  $\mathbb{R}^d$  and  $b_u, b_m$  are scalars, indicating the bias.

### 6.1 Bias Updates

Assume that our objective is given by

$$\frac{1}{2} \sum_{u \sim m} (f_{um} - r_{um})^2 + \frac{\lambda}{2} \left[ \sum_{u \in U} b_u^2 + \|v_u\|^2 + \sum_{m \in M} b_m^2 + \|w_m\|^2 \right] \text{ where } \lambda > 0.$$

Here  $U$  denotes the set of all users,  $M$  the set of all movies, and  $u \sim m$  represents the sum over all  $(u, m)$  pairs for which a rating exists. Write the optimal values of  $b_u$ , provided that all other values are fixed. That is, compute the optimal value of  $b_u | v, w, b_m, r$ .

**Solution:**

Set the derivative wrt. a particular user  $u'$  to 0:

$$\sum_{u' \sim m} (f_{u'm} - r_{u'm}) + \lambda b_{u'} = 0$$

$$b_{u'} = \frac{\sum_{u' \sim m} (r_{u'm} - f_{u'm})}{\lambda}$$

where  $u' m$  are the movies rated by  $u'$ .

### 6.2 Optimization

Is the problem jointly convex in  $v$  and  $w$ ? You need to prove this rather than just giving a binary answer. It suffices to prove this a very simplistic case, say for only 1 user and 1 movie.

**Solution:**

Since we assume only one user and one movie, by ignoring the bias terms, the Hessian of  $O(f(v, w)) = \frac{1}{2} \sum_{u \sim m} (f_{um} - r_{um})^2$  can be calculated as:

$$\begin{bmatrix} 2w^2 & 4vw - 2r_{um} \\ 4vw - 2r_{um} & 2v^2 \end{bmatrix}$$

It is not positive semi-definite. Therefore the problem is only element-wise convex but not jointly convex in  $v$  and  $w$ .

### 6.3 Cold Start

1. How could you address the problem of recommending movies to a new user?

**Solution:**

When a new user come in, we have little information about them, and thus the matrix factorization method can not learn much associations between the new user and the existing users. We should use the demographics information of the user to bridge its associations with existing users. Many ideas can be applied here. The most intuitive way is perhaps to do regression based on the demographics features and compute a similarity between the new user and existing users, then approximate  $v_u$  with a linear combination.  
Active learning?

2. How could you accomplish this for a new movie?

**Solution:**

Using meta data of the movie as additional information to encode the similarity, perhaps approximating the corresponding  $w_m$  as a linear combination of existing movies based on their similarities in terms of meta information. This can be encoded in the objective function.

Active learning?

## 7 Weather Prediction [15 points]

The weather in Pittsburgh is notoriously fickle. For simplicity we only consider sun and rain and we assume that the weather changes once per day. The weather satisfies the following transition probabilities:

- When it rains, the probability of sun the following day is 0.6.
- When the sun shines, the probability of rain the following day is 0.3.

### 7.1 Transition Matrix [3 points]

Write down the state transition matrix  $T$  for Pittsburgh's weather.

**Solution:**

	$r$	$s$
$r$	0.4	0.6
$s$	0.3	0.7

### 7.2 Stationary Distribution [5 points]

1. What are the eigenvalues of  $T$ ? Bonus points if you can find stationary distribution.

**Solution:**

The first eigenvalue is 1 with eigenvector  $\langle 1, 1 \rangle$ .

The second eigenvalue is 0.1 with eigenvector  $\langle 1, -0.5 \rangle$ .

Solve a linear system  $\bar{\pi}T = \bar{\pi}$  with the constraint that  $\bar{\pi}_1 + \bar{\pi}_2 = 1$ .

The stationary point is that  $P(r) = \frac{1}{3}$ ,  $P(s) = \frac{2}{3}$

2. What does this mean for numerically computing the stationary distribution to at least 1% accuracy.

**Solution:**

With a large finite or countably infinite states, the standard way of computing the stationary distribution is not possible. One good example is to compute the PageRank scores of the web.

We can use power iterations or MCMC which simulates random walks to find approximations to the stationary distribution.

In terms of MCMC, with fixed parameters the Monte Carlo Standard Error (MCSE) is bounded by the sample size. We can compute a Confidence Interval regarding how far our estimation is from the truth.

### 7.3 Unobserved Days [7 points]

Assume that we observe the weather over a ten day period. In particular, we observe the following:

- The sun shines on the first day.
- It rains on day 5.
- It rains on day 7.
- The sun shines on day 10.

1. What is the probability of sun on day 6? Derive the equation.

**Solution:**

Given day 5 and day 7 are rainy ,

$$P(r_6|r_5, r_7) = 0.4 * 0.4 = 0.16$$

$$P(s_6|r_5, r_7) = 0.6 * 0.3 = 0.18$$

$$P(r_6) = \frac{0.18}{0.16 + 0.18} = \frac{9}{17}$$

2. What is the most likely *weather sequence* on days 8 and 9. Derive the equation rather than just stating the result.

**Solution:**

Given day 7 is rainy and day 10 is sunny, enumerate all possible sequence and evaluate their likelihoods.

$$P(r_8, r_9|r_7, s_{10}) = 0.4 * 0.4 * 0.6 = 0.096$$

$$P(r_8, s_9|r_7, s_{10}) = 0.4 * 0.6 * 0.7 = 0.168$$

$$P(s_8, r_9|r_7, s_{10}) = 0.6 * 0.3 * 0.6 = 0.108$$

$$P(s_8, s_9|r_7, s_{10}) = 0.6 * 0.7 * 0.7 = 0.294$$

Thereore the most likely sequence is  $s_8 \rightarrow s_9$

*(This page was left blank intentionally)*

*(This page was left blank intentionally)*

# 10-601 Machine Learning, Midterm Exam

Instructors: Tom Mitchell, Ziv Bar-Joseph

Wednesday 12<sup>th</sup> December, 2012

There are 9 questions, for a total of 100 points.

This exam has 20 pages, make sure you have all pages before you begin.  
This exam is open book, open notes, but *no computers or other electronic devices*.

This exam is challenging, but don't worry because we will grade on a curve. Work efficiently.

Good luck!

Name: \_\_\_\_\_

Andrew ID: \_\_\_\_\_

Question	Points	Score
Short Answers	11	
GMM - Gamma Mixture Model	10	
Decision trees and Hierarchical clustering	8	
D-separation	9	
HMM	12	
Markov Decision Process	12	
SVM	12	
Boosting	14	
Model Selection	12	
Total:	100	

## Question 1. Short Answers

- (a) [3 points] For data  $D$  and hypothesis  $H$ , say whether or not the following equations must always be true.

- $\sum_h P(H = h|D = d) = 1$  ... is this always true?

**Solution:**

yes

- $\sum_h P(D = d|H = h) = 1$  ... is this always true?

**Solution:**

no

- $\sum_h P(D = d|H = h)P(H = h) = 1$  ... is this always true?

**Solution:**

no

- (b) [2 points] For the following equations, describe the relationship between them. Write one of four answers:

- (1) “=” (2) “ $\leq$ ” (3) “ $\geq$ ” (4) “(depends)”

Choose the most specific relation that always holds; “(depends)” is the least specific. Assume all probabilities are non-zero.

$P(H = h D = d)$	$P(H = h)$
$P(H = h D = d)$	$P(D = d H = h)P(H = h)$

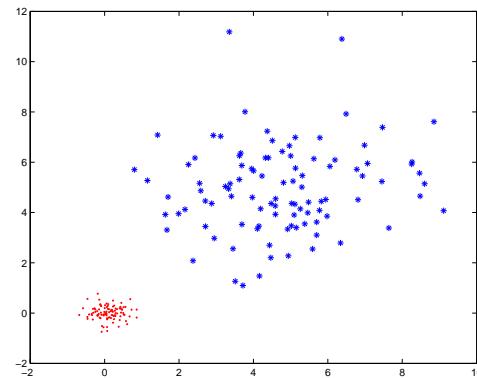
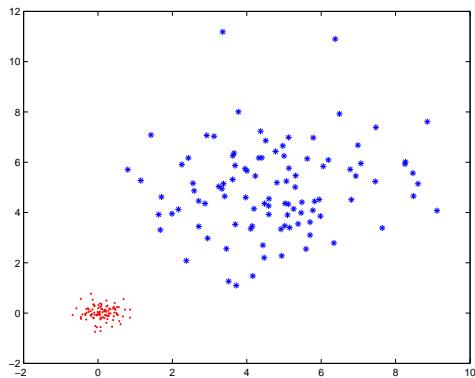
**Solution:**

$P(H|D)$  (DEPENDS)  $P(H)$

$P(H|D) \geq P(D|H)P(H)$  .. this is the numerator in Bayes Rule, have to divide by the normalizer  $P(D)$ , which is less than 1. Tricky...  $P(H|D) = P(D|H)P(H)/P(D) > P(D|H)P(H)$ .

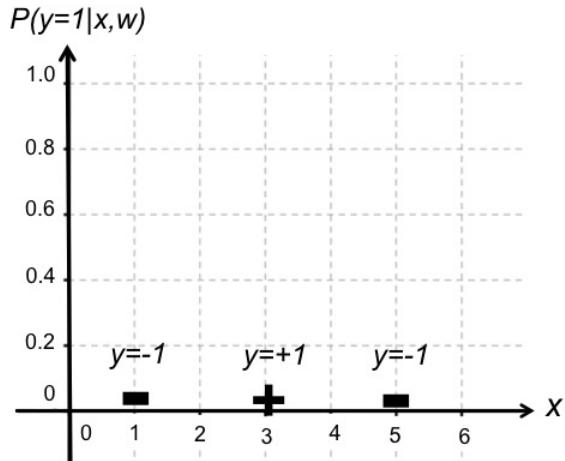
- (c) [2 points] Suppose you are training Gaussian Naive Bayes (GNB) on the training set shown below. The dataset satisfies Gaussian Naive Bayes assumptions. Assume that the variance is independent of instances but dependent on classes, i.e.  $\sigma_{ik} = \sigma_k$  where  $i$  indexes instances  $X^{(i)}$  and  $k \in 1, 2$  indexes classes. Draw the decision boundaries when you train GNB

- using the **same** variance for both classes,  $\sigma_1 = \sigma_2$
- using separate variance for each class  $\sigma_1 \neq \sigma_2$

**Solution:**

The decision boundary for part a will be linear, and part b will be quadratic.

- (d) [2 points] Assume that we have two possible conditional distributions ( $P(y = 1|x, w)$ ) obtained by training a logistic regression on the dataset shown in the figure below:

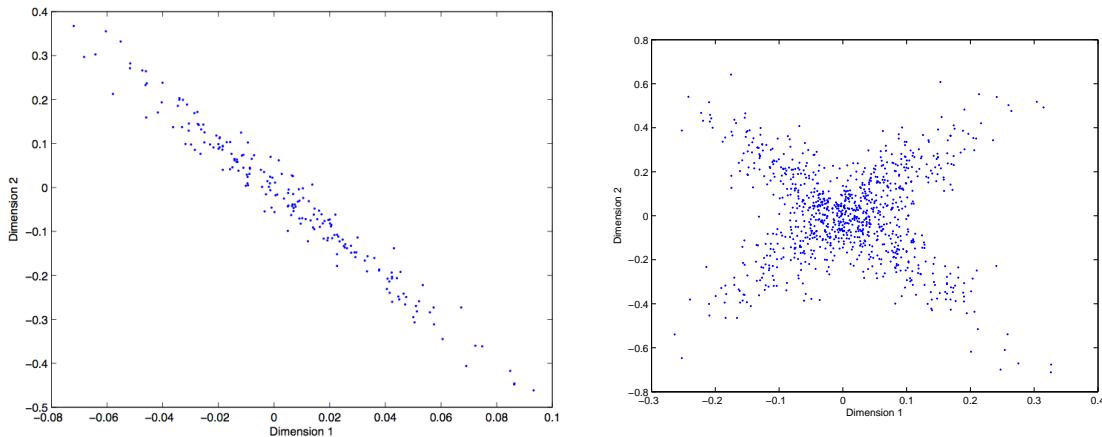


In the first case, the value of  $P(y = 1|x, w)$  is equal to  $1/3$  for all the data points. In the second case,  $P(y = 1|x, w)$  is equal to zero for  $x = 1$  and is equal to 1 for all other data points. One of these conditional distributions is obtained by finding the maximum likelihood of the parameter  $w$ . Which one is the MLE solution? Justify your answer in at most three sentences.

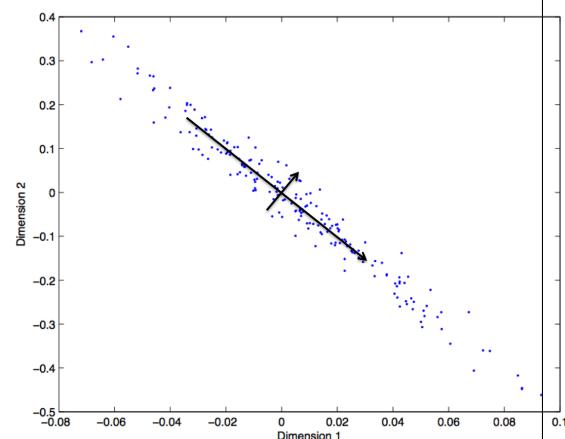
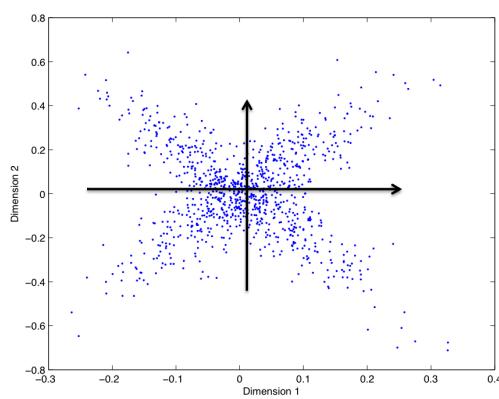
**Solution:**

The MLE solution is the first case where the value of  $P(y = 1|x, w)$  is equal to  $1/3$  for all the data points.

- (e) [2 points] Principal component analysis is a dimensionality reduction method that projects a dataset into its most variable components. You are given the following 2D datasets, draw the first and second principle components on each plot.



**Solution:**



## Question 2. GMM - Gamma Mixture Model

A Assume each data point  $X_i \in \mathbb{R}^+$  ( $i = 1 \dots n$ ) is drawn from the following process:

$$Z_i \sim \text{Multinomial}(\pi_1, \pi_2, \dots, \pi_K)$$

$$X_i \sim \text{Gamma}(2, \beta_{Z_i})$$

The probability density function of  $\text{Gamma}(2, \beta)$  is  $P(X = x) = \beta^2 x e^{-\beta x}$ .

- (a) [3 points] Assume  $K = 3$  and  $\beta_1 = 1, \beta_2 = 2, \beta_3 = 4$ . What's  $P(Z = 1|X = 1)$ ?

**Solution:**

$$P(Z = 1|X = 1) \propto P(X = 1|Z = 1)P(Z = 1) = \pi_1 e^{-1}$$

$$P(Z = 2|X = 1) \propto P(X = 1|Z = 2)P(Z = 2) = \pi_2 4 e^{-2}$$

$$P(Z = 3|X = 1) \propto P(X = 1|Z = 3)P(Z = 3) = \pi_3 16 e^{-4}$$

$$P(Z = 1|X = 1) = \frac{\pi_1 e^{-1}}{(\pi_1 e^{-1} + \pi_2 4 e^{-2} + \pi_3 16 e^{-4})}$$

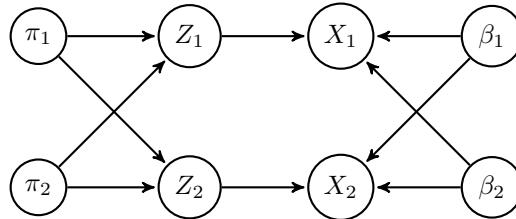
- (b) [3 points] Describe the E-step. Write an equation for each value being computed.

**Solution:**

For each  $X = x$ ,

$$P(Z = k|X = x) = \frac{P(X = x|Z = k)P(Z = k)}{\sum_{k'} P(X = x|Z = k')P(Z = k')} = \frac{\beta_k^2 x e^{-\beta_k x} \pi_k}{\sum_{k'} \beta_{k'}^2 x e^{-\beta_{k'} x} \pi_{k'}}$$

- (c) [2 points] Here's the Bayes net representation of the Gamma mixture model for  $k = n = 2$ . Note that we are treating  $\pi$ 's and  $\beta$ 's as variables – we have priors for them.



Would you say  $\pi$ 's are independent given the observations  $X$ ? Why?

**Solution:**

No.  $\pi_1 \rightarrow Z_1 \rightarrow \pi_2$  is an active trail since  $X$  is given.

- (d) For the following parts, choose true or false with an explanation in **one sentence**

- i. [1 point] Gamma mixture model can capture overlapping clusters, like Gaussian mixture model.

**Solution:**

(All or none. 1 pt iff you get the answer and the explanation correct) true. in the e-step it does soft assignment

ii. [1 point] As you increase  $K$ , you will **always** get better likelihood of the data.

**Solution:**

(All or none. 1 pt iff you get the answer and the explanation correct) false. Won't improve after  $K > N$

### Question 3. Decision trees and Hierarchical clustering

Assume we are trying to learn a decision tree. Our input data consists of  $N$  samples, each with  $k$  attributes ( $N \gg k$ ). We define the depth of a tree as the maximum number of nodes between the root and any of the leaf nodes (including the leaf, not the root).

- (a) [2 points] If all attributes are binary, what is the maximal number of leaf (decision) nodes that we can have in a decision tree for this data? What is the maximal possible depth of a decision tree for this data?

**Solution:**

$2^{(k-1)}$ . Each feature can only be used once in each path from root to leaf. The maximum depth is  $O(k)$ .

- (b) [2 points] If all attributes are continuous, what is the maximum number of leaf nodes that we can have in a decision tree for this data? What is the maximal possible depth for a decision tree for this data?

**Solution:**

Continuous values can be used multiple times, so the maximum number of leaf nodes can be the same as the number of samples,  $N$  and the maximal depth can also be  $N$ .

- (c) [2 points] When using **single link** what is the maximal possible depth of a hierarchical clustering tree for the data in 1? What is the maximal possible depth of such a hierarchical clustering tree for the data in 2?

**Solution:**

When using single link with binary data, we can obtain cases where we are always growing the cluster by 1 node at a time leading to a tree of depth  $N$ . This is also clearly the case for continuous values.

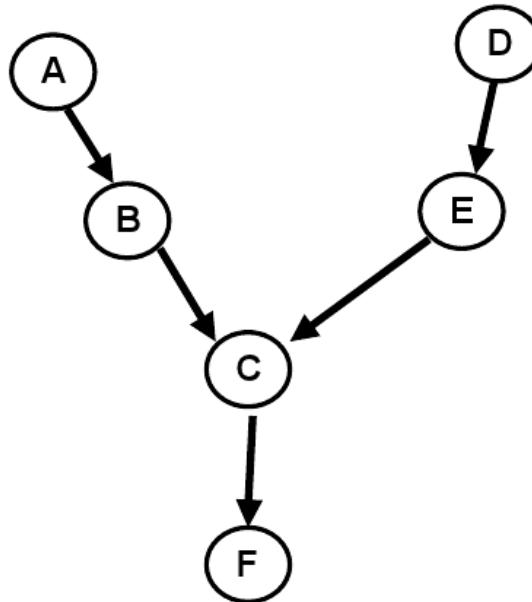
- (d) [2 points] Would your answers to (3) change if we were using **complete link** instead of **single link**? If so, would it change for both types of data? Briefly explain.

**Solution:**

While the answer for continuous values remain the same (its easy to design a dataset where each new sample is farther from any of the previous samples) for binary data, if  $k$  is small compared to  $N$  we will not be able to continue to add one node at a time to the initial cluster and so the depth will change to be lower than  $N$ .

## Question 4. D-separation

Consider the following Bayesian network of 6 variables.



- (a) [3 points] Set  $X = \{B\}$  and  $Y = \{E\}$ . Specify two distinct (not-overlapping) sets  $Z$  such that:  $X \perp\!\!\!\perp Y | Z$  (in other words,  $X$  is independent of  $Y$  given  $Z$ ).

**Solution:**

$Z = \{A\}$  and  $Z = \{D\}$

- (b) [2 points] Can you find another distinct set for  $Z$  (i.e. a set that does not intersect with any of the sets listed in 1)?

**Solution:**

The empty set  $Z = \{\}$

- (c) [2 points] How many distinct  $Z$  sets can you find if we replace  $B$  with  $A$  while  $Y$  stays the same (in other words, now  $X = \{A\}$  and  $Y = \{E\}$ )? What are they?

**Solution:**

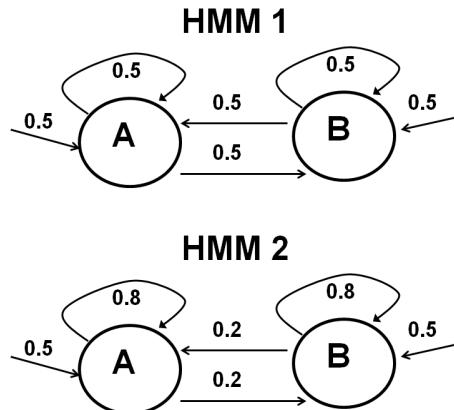
$Z = \{\}, Z = \{B\}$  and  $Z = \{D\}$

- (d) [2 points] If  $W \perp\!\!\!\perp X | Z$  and  $X \perp\!\!\!\perp Y | Z$  for some distinct variables  $W, X, Y, Z$ , can you say  $W \perp\!\!\!\perp Y | Z$ ? If so, show why. If not, find a counterexample from the graph above.

**Solution:**

No.  $A \perp\!\!\!\perp F | B$  and  $D \perp\!\!\!\perp A | B$  but  $D$  and  $F$  are not independent given  $B$ .

## Question 5. HMM



The figure above presents two HMMs. States are represented by circles and transitions by edges. In both, emissions are deterministic and listed inside the states.

Transition probabilities and starting probabilities are listed next to the relevant edges. For example, in HMM 1 we have a probability of 0.5 to start with the state that emits A and a probability of 0.5 to transition to the state that emits B if we are now in the state that emits A.

In the questions below,  $O_{100}=A$  means that the 100th symbol emitted by the HMM is A.

(a) [3 points] What is  $P(O_{100} = A, O_{101} = A, O_{102} = A)$  for HMM1?

**Solution:**

Note that  $P(O_{100}=A, O_{101}=A, O_{102}=A) = P(O_{100}=A, O_{101}=A, O_{102}=A, S_{100}=A, S_{101}=A, S_{102}=A)$  since if we are not always in state A we will not be able to emit A. Given the Markov property this can be written as:

$$P(O_{100}=A, O_{101}=A, O_{102}=A, S_{100}=A, S_{101}=A, S_{102}=A) = P(O_{100}=A|S_{100}=A) P(S_{100}=A) P(O_{101}=A|S_{101}=A) P(S_{101}=A|S_{100}=A) P(O_{102}=A|S_{102}=A) P(S_{102}=A|S_{101}=A)$$

The emission probabilities in the above equation are all 1. The transitions are all 0.5. So the only question is: What is  $P(S_{100}=A)$ ? Since the model is fully symmetric, the answer to this is 0.5 and so the total equation evaluates to:  $0.5^3$

(b) [3 points] What is  $P(O_{100} = A, O_{101} = A, O_{102} = A)$  for HMM2?

**Solution:**

$$0.5 * 0.8^2$$

(c) [3 points] Let  $P_1$  be:  $P_1 = P(O_{100} = A, O_{101} = B, O_{102} = A, O_{103} = B)$  for HMM1 and let  $P_2$  be:  $P_2 = P(O_{100} = A, O_{101} = B, O_{102} = A, O_{103} = B)$  for HMM2. Choose the correct answer from the choices below and briefly explain.

1.  $P_1 > P_2$
2.  $P_2 > P_1$
3.  $P_1 = P_2$
4. Impossible to tell the relationship between the two probabilities

**Solution:**

(a). P1 evaluates to  $0.5^4$  while P2 is  $0.5 * 0.2^4$  so clearly  $P_1 > P_2$ .

- (d) [3 points] Assume you are told that a casino has been using one of the two HMMs to generate streams of letters. You are also told that among the first 1000 letters emitted, 500 are As and 500 are Bs. Which of the following answers is the most likely (briefly explain):

1. The casino has been using HMM 1
2. The casino has been using HMM 2
3. Impossible to tell

**Solution:**

(c). While we saw in the previous question that it is much more likely to switch between A and B in HMM2, this is only true if we switch at every step. However, when aggregating over 1000 steps, since the two HMMs are both symmetric, both are likely to generate the \*same\* number of As and Bs.

## Question 6. Markov Decision Process

Consider a robot that is moving in an environment. The goal of the robot is to move from an initial point to a destination point as fast as possible. However, the robot has the limitation that if it moves fast, its engine can overheat and stop the robot from moving. The robot can move with two different speeds: *slow* and *fast*. If it moves fast, it gets a reward of 10; if it moves slowly, it gets a reward of 4. We can model this problem as an MDP by having three states: *cool*, *warm*, and *off*. The transitions are shown in below. Assume that the discount factor is 0.9 and also assume that when we reach the state *off*, we remain there without getting any reward.

$s$	$a$	$s'$	$P(s' a, s)$
cool	slow	cool	1
cool	fast	cool	1/4
cool	fast	warm	3/4
warm	slow	cool	1/2
warm	slow	warm	1/2
warm	fast	warm	7/8
warm	fast	off	1/8

- (a) [2 points] Consider the **conservative** policy when the robot always moves slowly. What is the value of  $J^*(cool)$  under the conservative policy? Remember that  $J^*(s)$  is the expected discounted sum of rewards when starting at state  $s$

**Solution:**

$$J^*(cool) = 4 + 0.9J^*(cool)$$

$$J^*(cool) = 40$$

- (b) [3 points] What is the optimal policy for each state?

**Solution:**

If in state *cool* then move *fast*. If in state *warm* then move *slow*.

- (c) [2 points] Is it possible to change the discount factor to get a different optimal policy? If yes, give such a change so that it results to a **minimum** changes in the optimal policy and if no justify your answer in at most two sentences.

**Solution:**

Yes, by decreasing the discount factor. For example by choosing the discount factor equal to zero the robot always chooses an action that gives the highest immediate reward.

- (d) [2 points] Is it possible to change the immediate reward function so that  $J^*$  changes but the optimal policy remains unchanged? If yes, give such a change and if no justify your answer in at most two sentences.

**Solution:**

Yes, for example by multiplying all the rewards by two.

- (e) [3 points] One of the important problems in MDPs is to decide what should be the value of the discount factor. For now assume that we don't know the value of discount factor but an expert person tells us that action sequence  $\{fast, slow, slow\}$  is preferred to the action sequence  $\{slow, fast, fast\}$  if we start from either of states *cool* or *warm*. What does it tell us about the discount factor? What ranges of discount factor is consistent with this preference?

**Solution:**

The discounted sum of future rewards using discount factor  $\lambda$  is calculated by:  $r + r(\lambda) + r(\lambda^2) + \dots$

So by solving the below equation, we would be able to find a range for discount factor  $\lambda$ :

$$10 + 4\lambda + 4\lambda^2 > 4 + 10\lambda + 10\lambda^2$$

## Question 7. SVM

### (a) Kernels

- i. [4 points] In class we learnt that SVM can be used to classify linearly inseparable data by transforming it to a higher dimensional space with a kernel  $K(x, z) = \phi(x)^T \phi(z)$ , where  $\phi(x)$  is a feature mapping. Let  $K_1$  and  $K_2$  be  $R^n \times R^n$  kernels,  $K_3$  be a  $R^d \times R^d$  kernel and  $c \in R^+$  be a positive constant.  $\phi_1 : R^n \rightarrow R^d$ ,  $\phi_2 : R^n \rightarrow R^d$ , and  $\phi_3 : R^d \rightarrow R^d$  are feature mappings of  $K_1$ ,  $K_2$  and  $K_3$  respectively. Explain how to use  $\phi_1$  and  $\phi_2$  to obtain the following kernels.

a.  $K(x, z) = cK_1(x, z)$

b.  $K(x, z) = K_1(x, z)K_2(x, z)$

**Solution:**

- a.  $\phi(x) = \sqrt{c}\phi_1(x)$   
b.  $\phi(x) = \phi_1(x)\phi_2(x)$

- ii. [2 points] One of the most commonly used kernels in SVM is the Gaussian RBF kernel:  $k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$ . Suppose we have three points,  $z_1$ ,  $z_2$ , and  $x$ .  $z_1$  is geometrically very close to  $x$ , and  $z_2$  is geometrically far away from  $x$ . What is the value of  $k(z_1, x)$  and  $k(z_2, x)$ ? Choose one of the following:

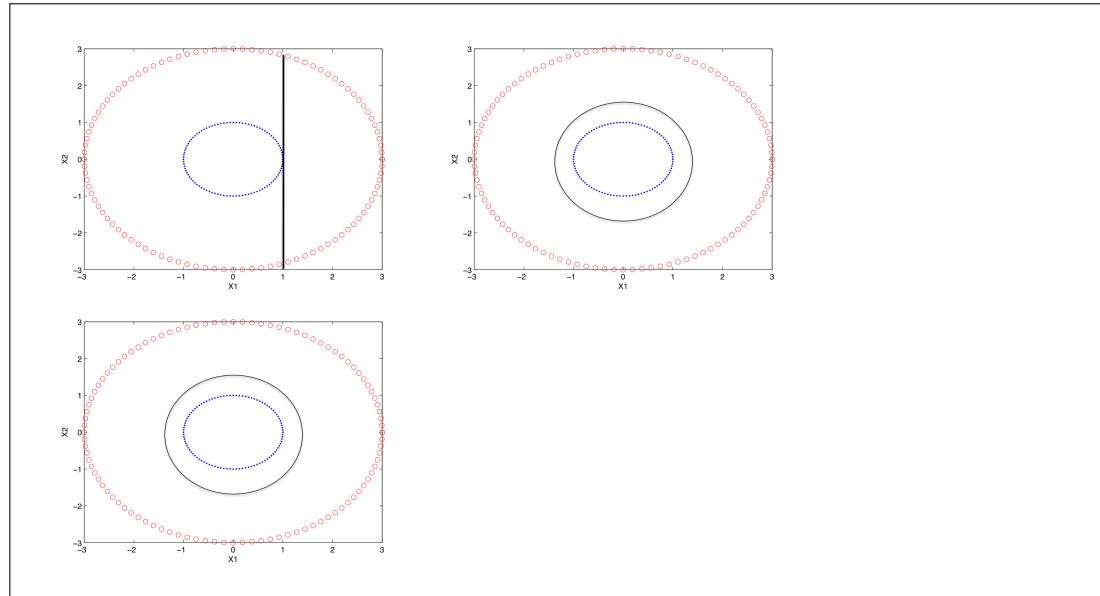
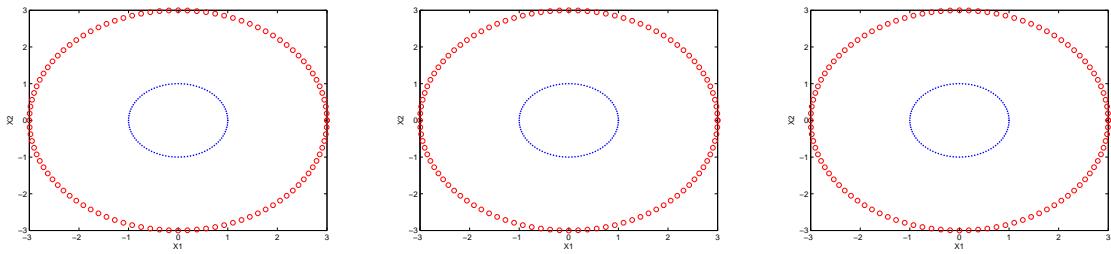
- a.  $k(z_1, x)$  will be close to 1 and  $k(z_2, x)$  will be close to 0.  
b.  $k(z_1, x)$  will be close to 0 and  $k(z_2, x)$  will be close to 1.  
c.  $k(z_1, x)$  will be close to  $c_1$ ,  $c_1 \gg 1$  and  $k(z_2, x)$  will be close to  $c_2$ ,  $c_2 \ll 0$ , where  $c_1, c_2 \in R$   
d.  $k(z_1, x)$  will be close to  $c_1$ ,  $c_1 \ll 0$  and  $k(z_2, x)$  will be close to  $c_2$ ,  $c_2 \gg 1$ , where  $c_1, c_2 \in R$

**Solution:**

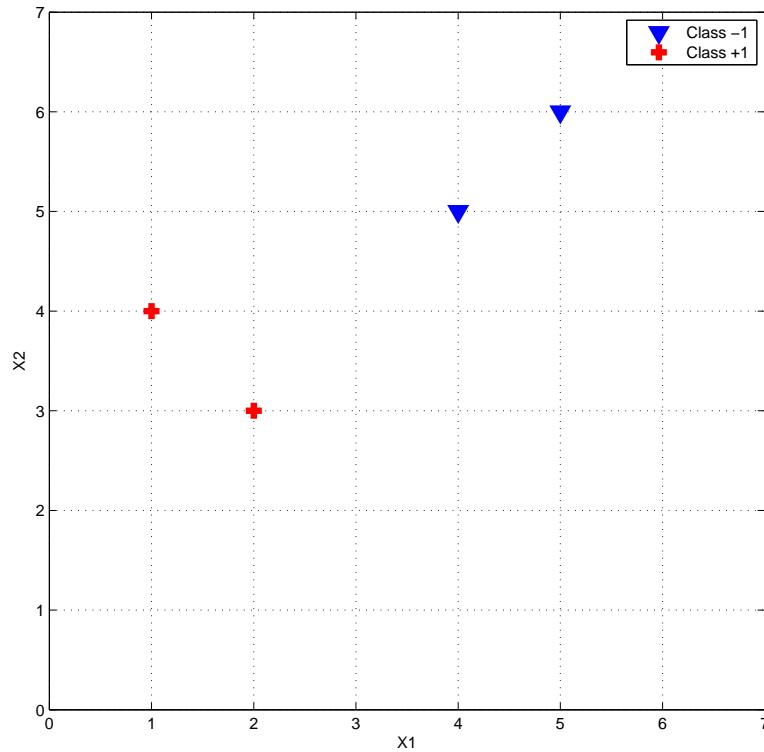
Correct answer is a, RBF kernel generates a "bump" around the center  $x$ . For points  $z_1$  close to the center of the bump,  $K(z_1, x)$  will be close to 1, for points away from the center of the bump  $K(z_2, x)$  will be close to 0.

- iii. [3 points] You are given the following 3 plots, which illustrates a dataset with two classes. Draw the decision boundary when you train an SVM classifier with linear, polynomial (order 2) and RBF kernels respectively. Classes have equal number of instances.

**Solution:**



(b) [3 points] Hard Margin SVM



Support vector machines learn a decision boundary leading to the largest margin from both classes. You are training SVM on a tiny dataset with 4 points shown in Figure 2. This dataset consists of two examples with class label -1 (denoted with plus), and two examples with class label +1 (denoted with triangles).

- Find the weight vector  $w$  and bias  $b$ . What's the equation corresponding to the decision boundary?

**Solution:**

SVM tries to maximize the margin between two classes. Therefore, the optimal decision boundary is diagonal and it crosses the point (3,4). It is perpendicular to the line between support vectors (4,5) and (2,3), hence its slope is  $m = -1$ . Thus the line equation is  $(x_2 - 4) = -1(x_1 - 3) = x_1 + x_2 = 7$ . From this equation, we can deduce that the weight vector has to be of the form  $(w_1, w_2)$ , where  $w_1 = w_2$ . It also has to satisfy the following equations:

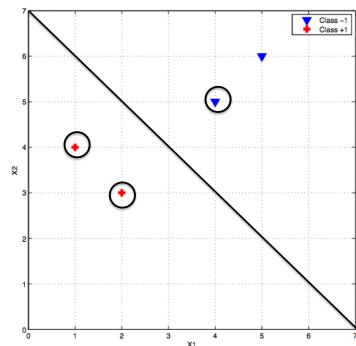
$$2w_1 + 3w_2 + b = 1 \text{ and}$$

$$4w_1 + 5w_2 + b = -1$$

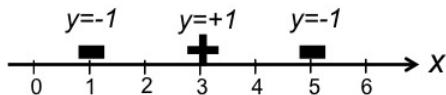
Hence  $w_1 = w_2 = -1/2$  and  $b = 7/2$

- ii. Circle the support vectors and draw the decision boundary.

**Solution:**



## Question 8. Boosting



**Solution:**

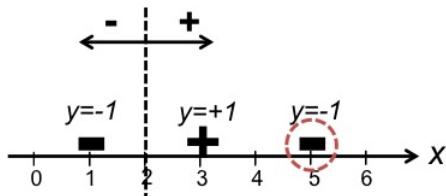


Figure 1: Sample training data for boosting algorithm.

In this problem, we study how boosting algorithm performs on a very simple classification problem shown in Figure 1. We use decision stump for each weak hypothesis  $h_i$ . Decision stump classifier chooses a constant value  $c$  and classifies all points where  $x > c$  as one class and other points where  $x \leq c$  as the other class.

- (a) [2 points] What is the initial weight that is assigned to each data point?

**Solution:**

$$\frac{1}{3}$$

- (b) [2 points] Show the decision boundary for the first decision stump (indicate the positive and negative side of the decision boundary).

**Solution:**

One possible solution is shown in the figure.

- (c) [3 points] Circle the point whose weight increases in the boosting process.

**Solution:**

One possible solution is shown in the figure.

- (d) [3 points] Write down the weight that is assigned to each data point after the first iteration of boosting algorithm.

**Solution:**

$$\epsilon_t = \frac{1}{3}$$
$$\alpha_t = \frac{1}{2} \ln(2) = 0.3465$$

For data points that are classified correctly  $D_2(i) = \frac{1/3 \exp(-0.3465)}{Z_2} \approx 0.25$  and for the data point that is classified incorrectly  $D_2(i) = \frac{1/3 \exp(0.3465)}{Z_2} \approx 0.5$  where  $Z_2$  is the normalization factor.

- (e) [3 points] Can boosting algorithm perfectly classify all the training examples? If no, briefly explain why. If yes, what is the minimum number of iteration?

**Solution:**

No, since the data is not linearly separable.

- (f) [1 point] **True/False** The training error of boosting classifier (combination of all the weak classifier) monotonically decreases as the number of iterations in the boosting algorithm increases. Justify your answer in at most two sentences.

**Solution:**

False, boosting minimizes loss function:  $\sum_{i=1}^m \exp(-y_i f(x_i))$  which doesn't necessary mean that the training error monotonically decrease. Please look at slides 14-18 [http://www.cs.cmu.edu/~tom/10601\\_fall2012/slides/boosting.pdf](http://www.cs.cmu.edu/~tom/10601_fall2012/slides/boosting.pdf).

## Question 9. Model Selection

- (a) [2 points] Consider learning a classifier in a situation with 1000 features total. 50 of them are truly informative about class. Another 50 features are direct copies of the first 50 features. The final 900 features are not informative.

Assume there is enough data to reliably assess how useful features are, and the feature selection methods are using good thresholds.

- How many features will be selected by mutual information filtering?

**Solution:**

about 100

- How many features will be selected by a wrapper method?

**Solution:**

about 50

- (b) Consider  $k$ -fold cross-validation. Let's consider the tradeoffs of larger or smaller  $k$  (the number of folds). For each, please select one of the multiple choice options.

- i. [2 points] With a higher number of folds, the estimated error will be, on average,

- (a) Higher.
- (b) Lower.
- (c) Same.
- (d) Can't tell.

**Solution:**

Lower (because more training data)

- (c) [8 points] Nearly all the algorithms we have learned about in this course have a tuning parameter for regularization that adjusts the bias/variance tradeoff, and can be used to protect against overfitting. More regularization tends to cause less overfitting.

For each of the following algorithms, we point out one such tuning parameter. If you increase the parameter, does it lead to MORE or LESS regularization? (In other words, MORE bias (and less variance), or LESS bias (and more variance)?) For every blank, please write MORE or LESS.

Naive Bayes: MAP estimation of binary features' $p(X Y)$ , using a $Beta(\alpha, \alpha)$ prior.	Higher $\alpha$ means...		regularization
Logistic regression, linear regression, or a neural network with a $\lambda \sum_j w_j^2$ penalty in the objective	Higher $\lambda$ means...		regularization
Bayesian learning for real-valued parameter $\theta$ , given a prior $p(\theta)$ , which might a wide or narrow shape. (For example, a high vs. low variance gaussian prior.)	Higher width of the prior distribution means...		regularization
Neural Network: number of hidden units, $n$	Higher $n$ means...		regularization
Feature selection with mutual information scoring: Include a feature in the model only if its $MI(feat, class)$ is higher than a threshold $t$ .	Higher $t$ means...		regularization
Decision tree: $n$ , an upper limit on number of nodes in the tree.	Higher $n$ means...		regularization
Boosting: number of iterations, $n$	Higher $n$ means...		regularization
Dimension reduction as preprocessing: Instead of using all features, reduce the training data down to $k$ dimensions with PCA, and use the PCA projections as the only features.	Higher $k$ means...		regularization

**Solution:**

- NB  $\alpha$ : more
- $\lambda$  L2 penalty: more
- Bayesian prior width: less
- Num. hidden units: less
- MI threshold: more
- Num. dtree nodes: less
- Num. boosting iter: less
- Num. PC's: less

# Exam MA 2823: Foundations of Machine Learning (Solution)

Instructor: Chloé-Agathe Azencott

December 18, 2015

- Exam duration: 3 hours.
- The exam is closed book and notes. No computer, phone, calculator.
- You can answer in English or in French.
- Please write concise (but argumented!) answers.
- No exact numerical computations are required.

## Useful formulas:

$$\exp(x) = \lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n.$$

The logistic function is defined by  $u \mapsto \frac{1}{1+\exp(-u)}$ .

$$e^{-1} = 0.368.$$

This exam has 15 questions, for a total of 42 points. Don't panic!

Question 1..... *1 point*  
What is regularization, and what purpose does it serve?

**Solution:** Regularization consists in penalizing the training error by a term that depends on the complexity of the model (equivalent to adjusting the bias/variance trade-off). The goal is to avoid overfitting. It is often used for sparsity / feature selection.

Question 2..... 1 point

A decision tree is said to be fully grown if each leaf only contains training points with the same label. Is a decision tree more likely to overfit if it is fully grown or not?

**Solution:** A fully-grown decision tree has a more complex decision boundary and is hence more likely to overfit.

Question 3..... 1 point

What kind of decision boundary can be learned with a multi-layer perceptron with one hidden layer?

**Solution:** Any continuous function (on a compact subset of  $\mathbb{R}^p$ ) can be approximated with a multi-layer perceptron with one hidden layer (“universal approximation”). In particular, unlike (single-layer) perceptrons, they can learn non-linear functions (e.g. XOR).

Question 4..... 1 point

You have data you want to cluster. You are considering several algorithms. How can you decide which one is the most appropriate?

**Solution:** Three families of strategies:

- Based on the shape of the clusters: cluster tightness/separation, Davies-Bouldin index, silhouette coefficient.
- Based on the stability of the results: under multiple repeats, when perturbing the data with noise or sampling.
- Based on domain knowledge: the clusters match prior knowledge to an extent.

Computational complexity (time/space) is another aspect that it can be necessary to take into account.

Question 5 ..... 2 points

Each tree of a random forest is built on a bootstrap sample of the training data. If  $n$  is the total number of training samples, on how many samples, in average, is each tree built?

**Solution:** Because bootstrap samples are drawn *with replacement*, a sample can be drawn multiple times. The probability  $p_i$  that the  $i$ -th training instance belongs to one bootstrap iteration is one minus the probability that is not picked among all  $n$  instances,  $n$  times. All instances have the same probability of being picked, which is  $1/n$ . Hence

$$p_i = 1 - \left(1 - \frac{1}{n}\right)^n \approx 1 - e^{-1} = 0.632$$

Hence each tree is trained on about 63.2% of the training data.

Question 6 ..... 2 points

What is the VC-dimension of a circle in  $\mathbb{R}^2$ ?

**Solution:** The VC-dimension of a circle in  $\mathbb{R}^2$  is the maximum number of points in the plane that can be shattered by a circle. Three points that are not aligned can be shattered by a circle: however we assign them + and - labels, we can always find a circle such that the points labeled + are inside this circle and the points labeled - are outside this circle. Hence the VC-dimension of a circle is at least 3. However, there are no configurations of 4 points that can be shattered by a circle. (Proof not asked.) Hence the VC-dimension of a circle is exactly 3.

Question 7 ..... 2 points

You are given a test that can determine whether a person is a terrorist or not from the emails they've written and received in the past year. The test is correct 95% of the time. Assume the prevalence of terrorists in this population is 1 in 10,000<sup>1</sup>. If 10,000 people have been labeled "positive" by this test, how many of them are not terrorists?

**Solution:** Denoting belonging to the terrorist class by  $T$  and the test being positive by +, let us apply Bayes rule:

$$P(T|+) = \frac{P(T)P(+|T)}{P(+)}$$

<sup>1</sup>In France, the Prime Minister evaluates the prevalence of terrorists to at most 1,500 (out of 64.410<sup>6</sup> inhabitants), so a prevalence of about  $0.23 \times 10^{-4}$ .

We can compute

$$\begin{aligned} P(+) &= P(+|T)P(T) + P(+|\bar{T})P(\bar{T}) \\ &= 0.95 * 10^{-4} + (1 - 0.95) * (1 - 10^{-4}) \approx 0.05. \end{aligned}$$

Hence:  $P(T|+) = \frac{10^{-4} \times 0.95}{0.05} \approx 0.02$

Finally,  $0.98 \times 10^4 = 9800$  of these people are not terrorists.

Question 8 ..... 2 points

Let us consider  $n$  data points  $\{x_i\}_{i=1,\dots,n}$  in one dimension ( $x \in \mathbb{R}$ ). Assume they are drawn from a normal distribution of mean  $\mu$  and variance  $\sigma^2$ :

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right].$$

Compute the maximum likelihood estimates of  $\mu$  and  $\sigma$ .

**Solution:** Likelihood:  $l(\mu, \sigma|X) = p(X|\mu, \sigma) = \prod_{i=1}^n p(x_i|\mu, \sigma)$ .

Log-likelihood:

$$\begin{aligned} L(\mu, \sigma|X) &= \sum_{i=1}^n \log p(x_i|\mu, \sigma) \\ &= \sum_{i=1}^n \log \left( \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right) \right) \\ &= \sum_{i=1}^n \log \left( \frac{1}{\sqrt{2\pi}\sigma} \right) - \frac{(x_i-\mu)^2}{2\sigma^2} \end{aligned}$$

To get the MLE, take the derivative and set it to 0.

$$\frac{\partial L}{\partial \mu} = \sum_{i=1}^n \left( \frac{-2\mu}{2\sigma^2} + \frac{2x_i}{2\sigma^2} \right) = \frac{1}{\sigma^2} n\mu - \sum_{i=1}^n x_i$$

Hence  $\hat{\mu} = \frac{1}{n} \sum_{i=1}^n x_i$

$$\frac{\partial L}{\partial \sigma} = \sum_{i=1}^n \left( \frac{-\sqrt{2\pi}}{\sqrt{2\pi}\sigma} + \frac{(x_i-\mu)^2}{\sigma^3} \right).$$

Hence

$$-\frac{n}{\hat{\sigma}} + \sum_{i=1}^n \frac{(x_i-\mu)^2}{\hat{\sigma}^3} = 0$$

and finally

$$\hat{\sigma} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \hat{\mu})^2}.$$

Question 9 ..... 2 points

The volume of a  $p$ -dimensional sphere of radius  $r$  is given by

$$\frac{2r^p \pi^{p/2}}{p\Gamma(p/2)}.$$

The  $\Gamma$  function is a continuous generalization of the factorial.

- (a) (1 point) Consider two  $p$ -dimensional spheres (centered at the origin), one (sphere A) of radius 1, and the other (sphere B), slightly smaller, of radius  $1 - \epsilon$ . What is the proportion of the volume of sphere A that lies between sphere A and sphere B?

**Solution:** This proportion is given by

$$\rho = \frac{\frac{2\pi^{p/2}}{p\Gamma(p/2)} - \frac{2(1-\epsilon)^p \pi^{p/2}}{p\Gamma(p/2)}}{\frac{2\pi^{p/2}}{p\Gamma(p/2)}} = 1 - (1 - \epsilon)^p$$

- (b) (1 point) What does this have to do with the curse of dimensionality?

**Solution:** This proportion goes to 0 when  $p$  goes to  $\infty$ . In other words, in high dimension, all the points within a sphere are concentrated at the boundary of this sphere, and there are virtually no points closer than that to the origin. Thus, hyperspace is very big, everything is far apart, and it is hard to rely on nearby points having similar labels because no points are nearby.

Question 10 ..... 4 points

Assume we are given data  $\{(x^1, y^1), \dots, (x^n, y^n)\}$  where  $x^i \in \mathbb{R}^p$  and  $y^i \in \mathbb{R}$ , and a parameter  $t > 0$ . We denote by  $X$  the  $n \times p$  matrix of row vectors  $x^1, \dots, x^n$  and  $y = (y^1, \dots, y^n)$ . The ridge regression estimator is defined as:

$$\hat{\beta}_{\text{ridge}} = \arg \min_{\beta} \|y - X\beta\|_2^2 \text{ s. t. } \|\beta\|_2^2 \leq t.$$

- (a) (2 points) Show there exists a unique  $\lambda > 0$  such that this formulation is equivalent to:

$$\hat{\beta}_{\text{ridge}} = \arg \min_{\beta} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2.$$

**Solution:** We are minimizing a quadratic form  $f(\beta)$  under the constraint that  $g(\beta) \leq 0$ , where  $g(\beta) = \|\beta\|_2^2 - t$ .

Either the unconstrained minimum lies in the feasible region:  $g(\beta) \leq 0$ , or it does not and the solution is at a point where the iso-contour of  $f$  and the feasible region are tangent. In this second case,  $g(\beta) = 0$  and the gradients of  $g$  and  $f$  point towards opposite directions. (Indeed,  $f$  increases towards the feasible region, while  $g$  increases away from the feasible region.)

Hence this problem can be solved by minimizing (in  $\beta$ ) the Lagrangian

$$L(\lambda, \beta) = f(\beta) - \lambda g(\beta)$$

under the constraints that  $\lambda > 0$  and  $\lambda g(\beta) = 0$ .

Minimizing the Lagrangian is equivalent to minimizing (in  $\beta$ ):

$$\|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 - \lambda t$$

and  $\lambda t$  is a constant. QED.

(b) (2 points) Give the explicit form of the solution. Does it always exist?

**Solution:**

$$\begin{aligned}\hat{\beta}_{\text{ridge}} &= \arg \min_{\beta} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \\ &= \arg \min_{\beta} (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta.\end{aligned}$$

Taking the gradient and setting it to 0, we obtain:

$$-2X^T(y - X\hat{\beta}_{\text{ridge}}) + 2\lambda\hat{\beta}_{\text{ridge}} = 0$$

hence

$$(X^T X + \lambda I) \hat{\beta}_{\text{ridge}} = X^T y.$$

Finally:

$$\hat{\beta}_{\text{ridge}} = (X^T X + \lambda I)^{-1} X^T y.$$

Note that  $(X^T X + \lambda I)$  can always be inverted if  $\lambda > 0$ .

Question 11 ..... 4 points

Let us consider the training data  $\{(x^1, y^1), \dots, (x^n, y^n)\}$  where  $x^i \in \mathbb{R}^p$  and  $y^i \in$

$\{-1, +1\}$ . A soft-margin SVM solves

$$\begin{aligned} \arg \min_{w \in \mathbb{R}^p, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \\ \text{s. t. } & y^i(\langle w, x^i \rangle + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad \forall i \end{aligned} \quad (1)$$

(a) (1 point) Is a soft-margin SVM more likely to overfit if  $C$  is large or small?

**Solution:** If  $C$  is large, more importance is given to the error on the training set, and the SVM is more likely to overfit.

(b) (1 point) Give one way of choosing  $C$  in practice.

**Solution:** By cross-validation.

(c) (1 point) What does this mean for a feature  $j$  if the solution  $w_j$  is close to 0?

**Solution:** That this feature is uninformative. The class won't depend on this feature. (Note: we're talking about a feature weight  $w_j$ , not a (support) vector coefficient  $\alpha_i$ .)

(d) (1 point) Give an interpretation of the two terms of Equation (1):  $\|w\|^2$  and  $\sum_{i=1}^n \xi_i$ .

**Solution:**  $\|w\|$  is the inverse of the margin.  
 $\sum_{i=1}^n \xi_i$  is the sum of slacks  $\xi_i$ , which quantify the error for each misclassified training point.

Question 12 ..... 6 points

Figure 1 represents the architecture of a multi-layer perceptron for  $K$  classes. The hidden units are logistic units.

We are given training data  $\{(x^i, y^i)\}_{i=1, \dots, n}$ , with  $x^i \in \mathbb{R}^p$  and  $y^i \in \{0, 1\}^K$ .

(a) (1 point) Each training point belongs to one class only. What is the value of  $\sum_{k=1}^K y_k^i$ ?

**Solution:**  $y_k^i = 1$  if  $x^i$  belongs to class  $k$  and 0 otherwise. It is meant to be 1.

(b) (1 point) The output units are softmax units. If  $o_k$  denotes the linear combination of the signals from the incoming units, the softmax units transform  $o_k$

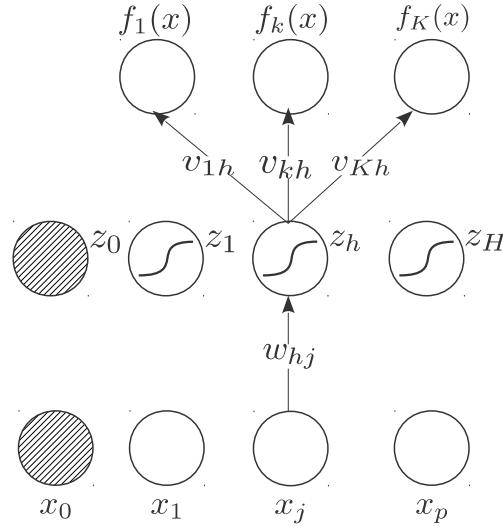


Figure 1: Multi-layer perceptron for  $K$  classes. There are  $p + 1$  input units:  $x_1, \dots, x_p$  and a bias unit  $x_0 = 1$ , and  $H + 1$  hidden units:  $z_1, \dots, z_H$  and a bias unit  $z_0 = 1$ . For clarity, not all connections are represented. Each unit of the input layer is connected to each unit of the hidden layer with a connection weight  $w_{hj}$ , and each unit of the hidden layer is connected to each unit of the output layer with a connection weight  $v_{kh}$ . The hidden units are logistic units.

as follows:

$$f_k(x) = \frac{\exp(o_k)}{\sum_{l=1}^K \exp(o_l)}$$

Why was this function chosen?

**Solution:** The softmax is a “well-behaved” version of the max. When one class gets a larger output than the others, the softmax will push the output of the corresponding unit towards 1, and push the outputs of the other units towards 0. Unlike the max, it is continuous and derivable, which has mathematical advantages.

(c) (2 points) We will use the cross-entropy error to train this network:

$$\text{Error}(f(x), y) = - \sum_{k=1}^K y_k \log f_k(x)$$

Compute the update rule for the weights  $v_{kh}$ .

**Solution:** The output of unit  $t$  is given by:

$$f_t(x) = \frac{\exp(v_t^\top z)}{\sum_{l=1}^K \exp(v_l^\top z)}$$

The error for training point  $(x, y)$  is given by:

$$\text{Error}(f(x), y) = - \sum_{t=1}^K y_t \log \frac{\exp(v_t^\top z)}{\sum_{l=1}^K \exp(v_l^\top z)}$$

We can show that

$$\frac{\partial f_t}{\partial v_{kh}} = \begin{cases} z_h f_k (1 - f_k) & \text{if } t = k \\ -z_h f_k f_t & \text{if } t \neq k \end{cases}$$

Hence

$$\frac{\partial \log(f_t)}{\partial v_{kh}} = \begin{cases} z_h (1 - f_k) & \text{if } t = k \\ -z_h f_k & \text{if } t \neq k \end{cases}$$

The gradient of the error w.r.t.  $v_{kh}$  is given by:

$$\frac{\partial \text{Error}}{\partial v_{kh}} = \left( \sum_{t \neq k} y_t \right) z_h f_k - y_k z_h (1 - f_k)$$

Because if  $y_k = 1$ , then  $\sum_{t \neq k} y_t = 0$  (and conversely), we get:

$$\frac{\partial \text{Error}}{\partial v_{kh}} = \begin{cases} -z_h (1 - f_k) & \text{if } y_k = 1 \\ z_h f_k & \text{if } y_k = 0 \end{cases} = z_h (f_k - y_k)$$

Finally, the update rule (gradient descent) is:  $v_{kh} \leftarrow v_{kh} - \eta z_h (f_k - y_k)$

- (d) (2 points) Write out the chain rule that enables backpropagation and compute the update rule for the weights  $w_{hj}$ .

**Solution:** Backpropagation chain rule:

$$\frac{\partial \text{Error}}{\partial w_{hj}} = \frac{\partial \text{Error}}{\partial z_h} \frac{\partial z_h}{\partial w_{hj}}$$

$$\frac{\partial \text{Error}}{\partial z_h} = - \sum_{k=1}^K y_k \frac{\partial f_k}{\partial z_h} \frac{1}{f_k}$$

And

$$z_h = \frac{1}{1 + e^{-w_h^\top x}}$$

$$\begin{aligned}\frac{\partial f_k}{\partial z_h} &= \frac{v_{kh} \exp(v_k^\top z) \sum_{l=1}^K \exp(v_l^\top z) - \exp(v_k^\top z_h) \sum_{l=1}^K v_{lh} \exp(v_l^\top z)}{\left(\sum_{l=1}^K \exp(v_l^\top z)\right)^2} \\ &= f_k \left( v_{kh} - \sum_{l=1}^K v_{lh} f_l \right)\end{aligned}$$

Hence

$$\frac{\partial \text{Error}}{\partial z_h} = - \sum_{k=1}^K y_k \left( v_{kh} - \sum_{l=1}^K v_{lh} f_l \right) = - \sum_{k=1}^K y_k v_{kh} + \sum_{l=1}^K v_{lh} f_l = \sum_{k=1}^K (f_k - y_k) v_{kh}$$

because  $\sum_{k=1}^K y_k = 1$ .

Because the derivative of the logistic  $\sigma(u)$  can be written as  $\sigma(u)(1 - \sigma(u))$ ,

$$\frac{\partial z_h}{\partial w_{hj}} = z_h (1 - z_h) x_j$$

Finally:

$$\frac{\partial \text{Error}}{\partial w_{hj}} = \sum_{k=1}^K ((f_k - y_k) v_{kh}) z_h (1 - z_h) x_j$$

And hence the update rule (gradient descent) is:

$$w_{hj} \leftarrow w_{hj} + \eta \sum_{k=1}^K ((y_k - f_k) v_{kh}) z_h (1 - z_h) x_j.$$

### Question 13 ..... 6 points

Assume we are given  $n$  points  $\{x^1, \dots, x^n\} \in \mathcal{X}^n$ .

Let  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  be a kernel, with the corresponding feature map  $\Phi : \mathcal{X} \rightarrow \mathcal{H}$ . We denote by  $K$  the  $n \times n$  matrix such that  $K_{ij} = k(x^i, x^j)$ .

We denote by  $\Sigma$  the sample covariance matrix of the images  $\{\Phi(x^1), \dots, \Phi(x^n)\}$  of our data. Let  $\lambda$  and  $V$  be an eigenvalue and corresponding eigenvector of  $\Sigma$ .

- (a) (1 point) Write  $k(x, x')$  as a function of  $\Phi(x)$  and  $\Phi(x')$ .

**Solution:**  $k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}} = \Phi(x)^\top \Phi(x')$ .

- (b) (1 point) Show that  $V$  is a linear combination of  $\{\Phi(x^1), \dots, \Phi(x^n)\}$ .

**Solution:**

$$n\lambda V = \sum_{i=1}^n \Phi(x^i) \Phi(x^i)^\top V$$

and  $\Phi(x^i)^\top V$  is a scalar, hence  $V$  spans  $\{\Phi(x^1), \dots, \Phi(x^n)\}$ .

- (c) (1 point) We can now write  $V$  as  $V = \sum_{i=1}^n \alpha_i \Phi(x^i)$ . Let us denote by  $\alpha$  the  $n$ -dimensional vector  $(\alpha_1, \dots, \alpha_n)$ . Show that  $\lambda, V$  are solution to  $n\lambda K\alpha = K^2\alpha$ .

**Solution:** We are looking for  $V, \lambda$  such that

$$n\lambda V = \frac{1}{n} \sum_{i=1}^n \Phi(x^i) \Phi(x^i)^\top V.$$

As demonstrated above, this implies that  $V$  spans  $\{\Phi(x^1), \dots, \Phi(x^n)\}$ . Let us write  $V = \sum_{i=1}^n \alpha_i \Phi(x^i)$  and reinject in the above equation:

$$n\lambda \sum_{i=1}^n \alpha_i \Phi(x^i) = \frac{1}{n} \sum_{i=1}^n \Phi(x^i) \Phi(x^i)^\top \sum_{j=1}^n \alpha_j \Phi(x^j).$$

We can now left-multiply by  $\Phi(x^k)^\top$  and sum over  $k$  to obtain:

$$n\lambda \sum_{k=1}^n \sum_{i=1}^n \Phi(x^k)^\top \Phi(x^i) \alpha_i = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \Phi(x^k)^\top \Phi(x^i) \Phi(x^i)^\top \Phi(x^j) \alpha_j.$$

Hence:

$$n\lambda \sum_{k=1}^n \sum_{i=1}^n k(x^k, x^i) \alpha_i = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n k(x^k, x^i) k(x^i, x^j) \alpha_j.$$

QED.

- (d) (1 point) Show how PCA in the feature space can be conducted directly in the original space  $\mathcal{X}$ , using only  $k$  and never computing  $\Phi(x)$  for any  $x$ .

**Solution:** All solutions of  $n\lambda\alpha = K\alpha$  satisfy the previous equation.

Finding the eigenvectors/eigenvalues of  $K$  gives us the eigendecomposition of  $\Sigma$ , hence the PCA in feature space.

The projection of  $\Phi(x)$  onto a PC, i.e. an eigenvector  $V$  of  $\Sigma$ , can be computed as:

$$\Phi(x)^\top V = \sum_{i=1}^n \alpha_i \Phi(x)^\top \Phi(x^i) = \sum_{i=1}^n \alpha_i k(x, x^i)$$

and does not require the explicit computation of  $\Phi$  either.

- (e) (1 point) What is the advantage of never having to compute  $\Phi(x)$  explicitly?

**Solution:** This kernel trick means that we don't need to know the form of  $\Phi$ , and we can compute PCA in spaces for which we don't know an exact mapping (e.g infinite-dimensional spaces with RBF kernel). In other cases (e.g. string kernel, Kendall-tau kernel) this can also afford computational advantages.

- (f) (1 point) Can you get an explicit expression of the principal components without using  $\Phi$ ?

**Solution:** No.  $V = \sum_{i=1}^n \alpha_i \Phi(x^i)$ . This is one drawback of kernel PCA.

Question 14 ..... 4 points

Show how to apply the kernel trick to the  $k$ -means algorithm.

**Solution:** The key point here is the ability to compute the distance from the image  $\Phi(x)$  of a data point  $x$  to the centroid of cluster  $C$  in *feature space*, which we'll call  $\nu$ .

By definition,

$$\nu = \frac{1}{|C|} \sum_{z \in C} \Phi(z)$$

Hence:

$$\begin{aligned} \|\Phi(x) - \nu\|^2 &= \|\Phi(x) - \frac{1}{|C|} \sum_{z \in C} \Phi(z)\|^2 \\ &= K(x, x) - \frac{2}{|C|} \sum_{z \in C} K(x, z) + \frac{1}{|C|^2} \sum_{z, z' \in C} K(z, z') \end{aligned} \quad (2)$$

Hence the re-assignment of data points to clusters can be computed in the initial space, without ever needing to compute  $\Phi$  explicitly.

The kernel  $k$ -means algorithm then proceeds as follows:

1. Randomly assign each data point to one of the  $k$  clusters.
2. Until convergence (assignment doesn't change), or a fixed number of iterations is reached, repeat:

- (a) Compute the distance of each data point  $x$  to the centroid of each of the  $k$  clusters, using Eq. 2 (only involving  $K$ ).
- (b) Re-assign each point to the cluster whose centroid it is closest to.

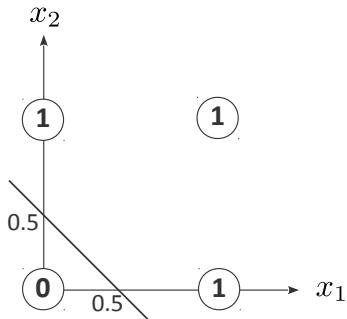
Question 15 ..... 4 points

Table 1 gives the decision table for OR. Give a perceptron that predicts this function.

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	1

Table 1: Decision table for OR.  $x_1$  and  $x_2$  are the inputs, and  $y$  the output.

**Solution:** The perceptron learns  $f(x) = s(w_0 + w_1x_1 + w_2x_2)$  where  $s$  is a threshold function.



$$w_0 = -0.5$$

$$w_1 = 1$$

$$w_2 = 1$$

This is one of many possible solutions.  $w_0, w_1, w_2$  must give the equation of a line that separates  $(0, 0)$  from  $(0, 1), (1, 0)$  and  $(1, 1)$ .

A

**Midterm Exam**  
GU4241/GR5241 Fall 2016

**Name**

---

**UNI**

---

### Problem 0: UNI (2 points)

Write your name and UNI on the first page of the problem sheet. After the exam, please return the problem sheet to us.

### Problem 1: Short questions (2+2+3+3+4+4 points)

Short answers (about one sentence) are sufficient.

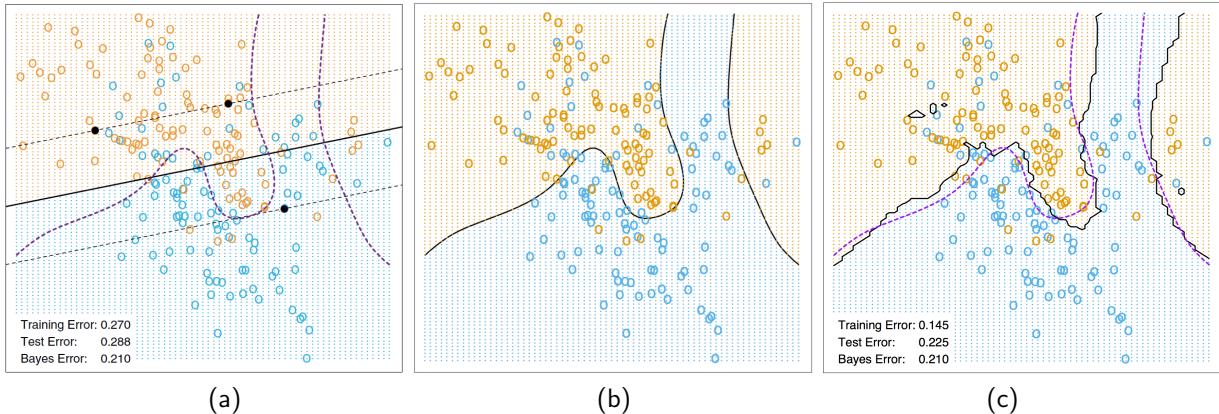
- (a) **(Yes/No)** Does the optimality of the Bayes classifier depend on the choice of loss function? Explain briefly.
- (b) **(Yes/No)** Does the EM algorithm always converge to a global maximizer of the likelihood? Explain briefly.
- (c) Consider a natural cubic spline and a polynomial regression with the same degree of freedom on the same data set. Which is likely to be more stable for extreme values of the predictor?
- (d) Which one of the following algorithms does not rely on the normality assumption:  $K$ -means, LDA, ordinary least squares, principle components analysis. Explain briefly.
- (e) Describe sampling from a finite mixture model  $\pi(x) = \sum_{k=1}^K c_k p(x|\theta_k)$  as a two-step procedure.
- (f) List two regression methods for which it is possible to compute the leave one out cross validation (LOOCV) error analytically without performing  $n$  fits.

### Solution:

- (a) **Yes.** Bayes classifier is optimal under 0-1 loss.
- (b) **No.** EM algorithm always converges. But it may converge to a local maxima.
- (c) The natural cubic spline. **Natural cubic spline requires the fit be linear beyond the boundaries.**
- (d) PCA.  **$K$ -means clustering is a special case of the EM algorithm for mixture models. When the model is finite mixture of Gaussian distributions with the identity as the covariance matrix, EM algorithm is equivalent to  $k$ -means.** LDA assumes the data from each class follow a Gaussian distribution with the same covariance matrix. Ordinary least squares assumes the errors are i.i.d. normal.
- (e)
  - Choose a mixture component at random. Each component  $k$  is selected with probability  $c_k$ .
  - Sample  $x_i$  from  $p(x|\theta_k)$ .
- (f) **Ordinary least squares, ridge regression, cubic splines, natural cubic splines, smoothing splines.**

### Problem 2: Decision boundaries (10 points)

The following pictures, which we have all seen in class, show the output of several different classifiers. Recall that the thick line is the decision boundary determined by the classifier; you can ignore the dashed lines.



For each of the three pictures:

- Name at least one classifier which could have produced this solution. Explain why.
- Name at least one classifier which could not have produced the solution. Explain why not.

#### Solution:

	(a)	(b)	(c)
could be generated by reason	logistic regression or LDA linear boundary, class overlap	Bayes classifier smooth, non-linear boundary	K-nearest-neighbor classifier non-linear boundary
could not be generated by	K-nearest-neighbor classifier Trees or RF (smooth slope)	any linear classifier Trees or RF (smooth)	any linear classifier

**Problem 3: K-means clustering (10 points)**

Perform K-means clustering on the following 2-dimensional observations with  $K = 3$  and initial labels  $(1, 1, 2, 3, 3, 2)$ . Use the *Manhattan* distance between a pair of points:  $d(A, B) = |X_{1A} - X_{1B}| + |X_{2A} - X_{2B}|$ , instead of the Euclidean distance. With this distance, the centroid of a cluster is obtained by taking the median of the samples in each dimension. Show your results after each iteration.

Obs.	$X_1$	$X_2$
A	1	4
B	1	3
C	3	4
D	5	2
E	3	2
F	3	0

**Solution:**

The centroids of the three clusters are  $(1, 3.5)$ ,  $(3, 2)$  and  $(4, 2)$ . The distances between the observations and these centroids is shown in the following table:

dist.	$C_1$	$C_2$	$C_3$
A	0.5	4	5
B	0.5	3	4
C	2.5	2	3
D	5.5	2	1
E	3.5	0	1
F	5.5	2	3

So the new labels after the first iteration are  $(1, 1, 2, 3, 2, 2)$ , and the centroids of the new clusters are  $(1, 3.5)$ ,  $(3, 2)$  and  $(5, 2)$ . Since only one centroid changed, we just need to compute the distance between the third centroid and the data points.

dist.	$C_1$	$C_2$	$C_3$
A	0.5	4	6
B	0.5	3	5
C	2.5	2	4
D	5.5	2	0
E	3.5	0	2
F	5.5	2	4

The clusters remain the same, so the result is  $\{A, B\}$ ,  $\{C, E, F\}$  and  $\{D\}$ .

#### Problem 4: Logistic Regression (10 points)

Suppose we have a dataset with  $N$  observations, and each observation consists of three values:

- $y$ : binary variable that is 1 if a student passed and 0 if a student failed the exam
- $x_1$ : the number of hours spent studying for the exam
- $x_2$ : a binary variable indicating whether or not the student passed the previous exam.

Suppose upon fitting a logistic regression of  $y$  on  $x_1$ ,  $x_2$ , and an intercept, the estimates for  $\beta = (\beta_0, \beta_1, \beta_2)$  are

$$\begin{aligned}\hat{\beta}_0 &= -1.2 \\ \hat{\beta}_1 &= 0.3 \\ \hat{\beta}_2 &= 1.2\end{aligned}$$

Now suppose instead of using the number of hours spent studying, we used the number of minutes spent on the exam preparation. Can you identify what the new  $\beta_0$ ,  $\beta_1$ , and  $\beta_2$  would be? Explain your answer.

#### Solution:

Logistic regression is fit by maximizing the likelihood function, which can be written in the following way:

$$\hat{\beta} = \arg \min_{\beta} \prod_{i:y_i=1} \frac{e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}}}{1 + e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}}} \prod_{i:y_i=0} \frac{1}{1 + e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}}}.$$

This only depends on the parameters through a linear function. After we change the unit, we want to solve the following optimization problem:

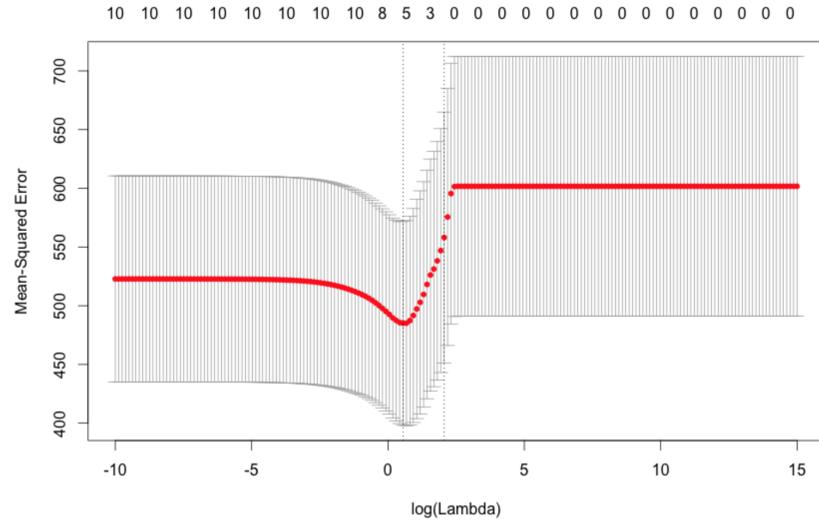
$$\hat{\beta}^{\text{new}} = \arg \min_{\beta} \prod_{i:y_i=1} \frac{e^{\beta_0 + \beta_1 60x_{i1} + \beta_2 x_{i2}}}{1 + e^{\beta_0 + \beta_1 60x_{i1} + \beta_2 x_{i2}}} \prod_{i:y_i=0} \frac{1}{1 + e^{\beta_0 + \beta_1 60x_{i1} + \beta_2 x_{i2}}}.$$

The maximizers  $\hat{\beta}$  and  $\hat{\beta}^{\text{new}}$  are related by

$$\begin{aligned}\hat{\beta}_0^{\text{new}} &= \hat{\beta}_0 = -1.2, \\ \hat{\beta}_1^{\text{new}} &= \frac{1}{60} \hat{\beta}_1 = 0.005, \\ \hat{\beta}_2^{\text{new}} &= \hat{\beta}_2 = 1.2.\end{aligned}$$

### Problem 5: Tuning parameter of lasso

The plot below displays the cross-validation errors of lasso regression computed on a range of tuning parameters  $\lambda$ . There are  $p = 10$  parameters in the model, and the training set and test set has  $n = 100$  observations each.



In terms of the cross validation error, what properties do the two models marked with dashed vertical lines in the plot have? Would you expect an appropriate shrinkage method to give a smaller test MSE than the test MSE of multiple linear regression for this data?

#### Solution:

The two dashed lines mark the model with the smallest cross validation error and the model chosen with the one standard error rule; that is, the simplest model whose cross validation error is within one standard error of the minimum cross validation error. The lasso regression when  $\lambda \rightarrow 0$  corresponds to the linear regression with  $(10 + 1)$  parameters including the intercept and the lasso regression when  $\lambda \rightarrow \infty$  corresponds to the linear regression with intercept only. We observe that the lasso regression with a certain tuning parameter has the smallest cross validation error. **Since the mean-squared errors from cross validation are estimates of the test MSE, we expect that the lasso regression using the best  $\lambda$  from cross-validation will give a smaller test MSE than the test MSE of multiple linear regression.**

**Problem 6: Variable selection (10 points)**

Your colleague fitted a multivariate linear regression model  $Y = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$  and found all but three p-values are significant in the t-test. He decides to drop those three variables and keep all the remaining predictors. Do you think this is a good idea? Briefly explain your answer. If you think this method is inappropriate, could you suggest an alternative?

**Solution:**

The colleague's method is not reasonable because the t-test only tests the marginal effect of each predictor. If there is collinearity, it is possible that after removing one predictor, formerly insignificant predictors become significant. **If  $p$  is large, choosing significant predictors at a fixed significance level could also lead to a large number of false positives. A better approach would be to apply forward or backward stepwise selection, then use cross-validation to select the optimal model.**

what's the meaning of cross-validation here, there is not penalty term

### Problem 7: Step function regression (10 points)

We fit a step function regression on a dataset with a single predictor  $X$ . 3 knots  $c_1, c_2, c_3$  in the range of  $X$  are selected. We construct 4 variables

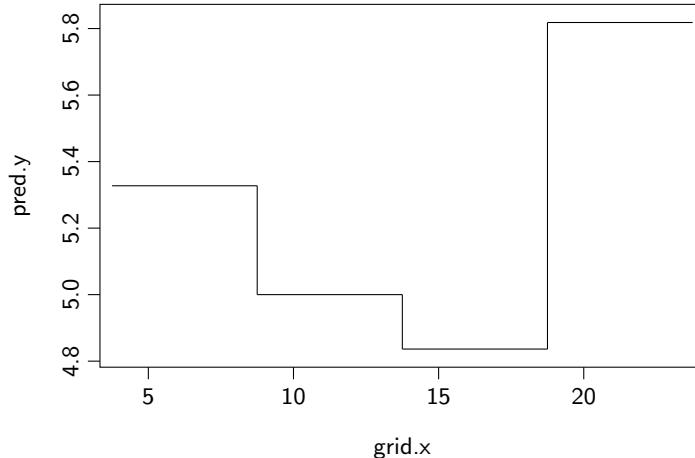
$$C_0(X) = \mathbb{1}_{\{X < c_1\}}, \quad C_1(X) = \mathbb{1}_{\{c_1 \leq X < c_2\}}, \quad C_2(X) = \mathbb{1}_{\{c_2 \leq X < c_3\}}, \quad C_3(X) = \mathbb{1}_{\{c_3 \leq X\}},$$

where  $\mathbb{1}_{\{\cdot\}}$  is an indicator function. For example,  $C_0$  takes value 1 when  $X < c_1$  is true; and is equal to 0 otherwise. Note that the linear model using  $C_0(X), C_1(X), C_2(X), C_3(X)$  as predictors is:

$$Y = \beta_0 C_0(X) + \beta_1 C_1(X) + \beta_2 C_2(X) + \beta_3 C_3(X) + \epsilon, \quad \epsilon \stackrel{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2).$$

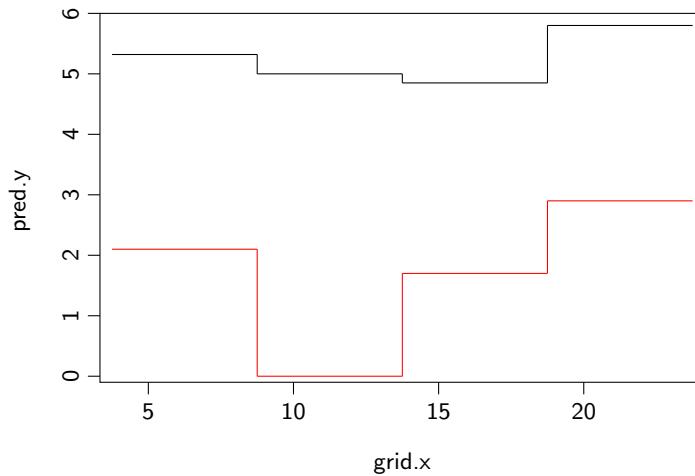
The plot below displays the fitted step function on a dataset.

Now we fit a lasso regression using the same predictors. Using the tuning parameter  $\lambda = 0.17$ , we get 3 nonzero  $\beta_i$ s out of 4 and  $\hat{\beta}_1 = 0$ . Sketch the lasso-fitted step function on the plot below. Briefly explain how you derive this lasso solution.



#### Solution:

If we fit a lasso regression using the same predictors, the estimates will be shrunk to zero. We already know that  $\hat{\beta}_1 = 0$ , which implies that the fitted value is zero when  $c_1 \leq x < c_2$ . The sketch of the lasso-fitted step function is shown by the red curve in the plot below. The black curve represents the fit without the  $L_1$  penalty.



**Problem 8: Principle components regression (10 points)**

The singular value decomposition (SVD) of the input matrix  $\mathbf{X}$  provides us some insights into the nature of the regression methods. We assume that the input matrix  $\mathbf{X}$  is centered and appropriately scaled. The SVD of the  $N \times p$  matrix  $\mathbf{X}$  has the form

$$\mathbf{X} = \mathbf{UDV}^T.$$

We assume all singular values  $d_j$  are positive. Using the singular value decomposition we can write the least squares fitted vector as

$$\begin{aligned}\mathbf{X}\hat{\beta}^{\text{ls}} &= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \\ &= \mathbf{U}\mathbf{U}^T\mathbf{y} \\ &= \sum_{j=1}^p \mathbf{u}_j \mathbf{u}_j^T \mathbf{y},\end{aligned}$$

where  $\mathbf{u}_j$  is the  $j$ th column of  $\mathbf{U}$ . This expression means we can obtain the fitted value directly if we use the orthonormal basis  $\mathbf{u}_1, \dots, \mathbf{u}_p$ .  $\mathbf{u}_1^T \mathbf{y}, \dots, \mathbf{u}_p^T \mathbf{y}$  are the coordinates of  $\mathbf{y}$  with respect to the new basis.

(a) We know that the ridge solutions are

$$\mathbf{X}\hat{\beta}^{\text{ridge}} = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}.$$

Similar to the least squares, can you rewrite the expression using the SVD of  $\mathbf{X}$ ?

(b) The idea of principle components regression is that we replace the original inputs  $X_j$  by a small set of linear combinations of  $X_j$  based on the principle components directions  $v_m$  ( $v_m$  is the  $m$ th column of  $\mathbf{V}$ ). More precisely, we derive the input columns  $\mathbf{z}_m = \mathbf{X}v_m$ , and then regress  $\mathbf{y}$  on  $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_M$  for some  $M \leq p$ . If we write our estimate as

$$\hat{\mathbf{y}}^{\text{pcr}} = \sum_{m=1}^M \hat{\theta}_m \mathbf{z}_m,$$

what are the estimated  $\hat{\theta}_m$ s which minimize the residue sum of squares?

(c) Could you list some pros and cons of principle components regression?

**Solution:**

(a)

$$\begin{aligned}\mathbf{X}\hat{\beta}^{\text{ridge}} &= (\mathbf{UDV}^T)(\mathbf{V}\mathbf{D}^2\mathbf{V}^T + \lambda\mathbf{I})^{-1}\mathbf{V}\mathbf{D}\mathbf{U}^T\mathbf{y} \\ &= \sum_{j=1}^p \mathbf{u}_j \frac{d_j^2}{d_j^2 + \lambda} \mathbf{u}_j^T \mathbf{y}.\end{aligned}$$

(b) Note that  $\mathbf{z}_m$  are orthogonal. Therefore

$$\hat{\theta}_m = \frac{\langle \mathbf{y}, \mathbf{z}_m \rangle}{\langle \mathbf{z}_m, \mathbf{z}_m \rangle}.$$

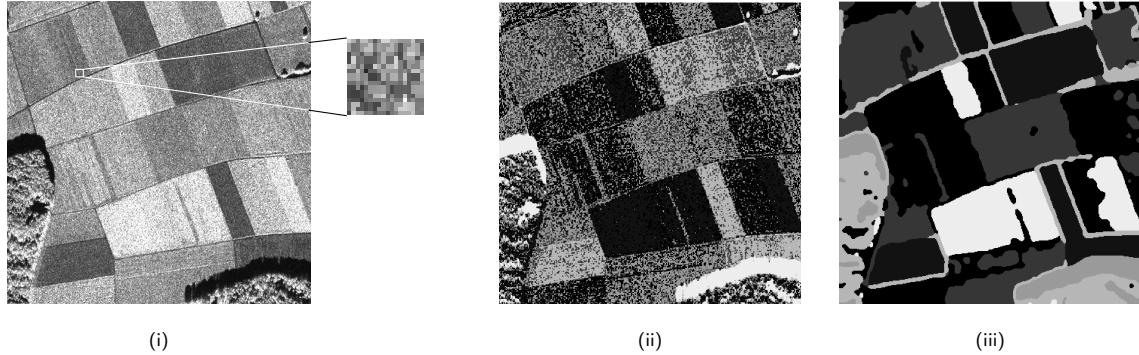
(c) Pros:

- PCR can provide stable estimate when  $\mathbf{X}^T\mathbf{X}$  is almost singular.
- PCR can reduce the dimension of the problem.

Cons: it is difficult to interpret the model.

### Problem 9: Histogram clustering (5+5 points)

In homework 1, you implemented a histogram clustering algorithm for image segmentation. Recall that the histograms you used had been extracted by placing a small window at regularly spaced points in the image and computing a histogram from the points inside the window (figure (i) below):

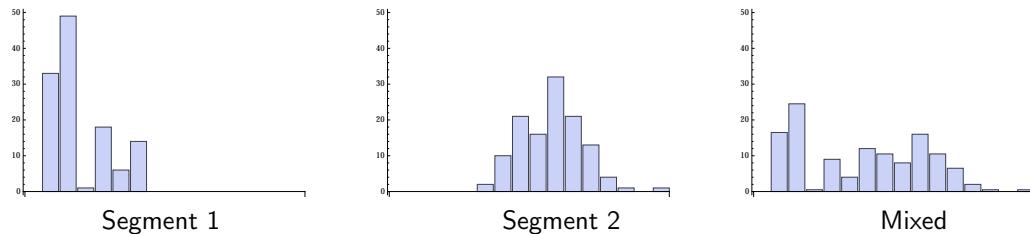


The two segmentation solutions (ii) and (iii) have been obtained by the same algorithm on the same image, with histograms extracted at the same locations. The only difference between the two data sets is the size of the histogram windows: One uses histogram over small windows ( $3 \times 3$  pixels), one is based on large windows ( $19 \times 19$  pixels). The centers of neighboring windows are 4 pixels apart.

- (a) Compare the segmentation solutions (ii) and (iii). Can you tell which one has been generated using small histogram windows, and which using large ones? Please explain your answer.
- (b) In solution (iii), you can see that the borders between some neighboring segments become a segment of their own. Can you explain why that happens?

#### Solution:

- (a) Small histogram windows: (ii) Large histogram windows: (iii)  
The large histograms windows have a large overlap (the overlap with any neighboring window is almost half the window size). The distributions represented by the histograms are therefore more similar (they share almost half their data points), which results in smoother segmentation results.
- (b) At the boundary, the large window overlaps two segments, and the histogram extracted from this window is hence a mixture of the distributions of the segments. For example:



If the segment distributions differ significantly, the mixed distribution differs significantly from both, and becomes a segment in its own right.

# 6.867 Machine Learning

## Problem Set 1 Solutions

Due date: Monday September 27

Please address all questions and comments about this problem set to [6867-staff@csail.mit.edu](mailto:6867-staff@csail.mit.edu). You will need to use MATLAB for some of the problems but essentially all the code is provided. If you are not familiar with MATLAB, please consult <http://www.ai.mit.edu/courses/6.867/matlab.html> and the links therein.

## Part I: background

Suppose we have a probability distribution or density  $p(x; \theta)$ , where  $x$  may be discrete or continuous depending on the problem we are interested in.  $\theta$  specifies the parameters of this distribution such as the mean and the variance of a one dimensional Gaussian. Different settings of the parameters imply different distributions over  $x$ . The available data, when interpreted as samples  $x_1, \dots, x_n$  from one such distribution, should favor one setting of the parameters over another. We need a formal criterion for gauging how well any potential distribution  $p(\cdot|\theta)$  “explains” or “fits” the data. Since  $p(x|\theta)$  is the probability of reproducing any observation  $x$ , it seems natural to try to maximize this probability. This gives rise to the Maximum Likelihood estimation criterion for the parameters  $\theta$ :

$$\hat{\theta}_{\text{ML}} = \arg \max_{\theta} L(x_1, \dots, x_n; \theta) = \arg \max_{\theta} \prod_{i=1}^n p(x_i|\theta)$$

where we have assumed that each data point  $x_i$  is drawn independently from the same distribution so that the likelihood of the data is  $L(x_1, \dots, x_n; \theta) = \prod_{i=1}^n p(x_i|\theta)$ . Likelihood is viewed primarily as a function of the parameters, a function that depends on the data. The above expression can be quite complicated (depending on the family of distributions we are considering), and make maximization technically challenging. However, any monotonically increasing function of the likelihood will have the same maxima. One such function is log-likelihood  $\log L(x_1, \dots, x_n; \theta)$ ; taking the log turns the product into a sum, making derivatives significantly simpler. We will maximize the log-likelihood instead of likelihood.

## Problem 1: Maximum Likelihood Estimation

Consider a sample of  $n$  real numbers  $x_1, x_2, \dots, x_n$  drawn independently from the same distribution that needs to be estimated. Assuming that the underlying distribution belongs to one of the following parametrized families, the goal is to estimate its parameters (each family should be treated separately):

$$\text{Uniform : } p(x; a) = \frac{1}{a} \text{ for } x \in [0, a], 0 \text{ otherwise} \quad (1)$$

$$\text{Exponential : } p(x; \eta) = \frac{1}{\eta} \exp(-x/\eta), \eta > 0 \quad (2)$$

$$\text{Gaussian : } p(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \quad (3)$$

1. (10 points) Derive the maximum likelihood estimators  $\hat{a}_{\text{ML}}$ ,  $\hat{\eta}_{\text{ML}}$ ,  $\hat{\mu}_{\text{ML}}$ ,  $\hat{\sigma}_{\text{ML}}^2$ . The estimators should be obtained by maximizing the log-likelihood of the dataset under each of the families, and should be a function of  $x_1, x_2, \dots, x_n$  only.

### Solution:

- The likelihood of the uniform distribution is given by:

$$L(a) = \prod_{i=1}^n p(x_i; a) = \begin{cases} 0 & \text{if } a < \max_{i=1}^n x_i \\ a^{-n} & \text{if } a \geq \max_{i=1}^n x_i \end{cases}$$

Because  $a^{-n}$  is positive and decreasing in  $a$ , the parameter that maximizes the likelihood is the smallest  $a$  that is not less than  $\max_{i=1}^n x_i$ . Thus

$$\hat{a}_{\text{ML}} = \max_{i=1}^n x_i$$

- The log-likelihood of the exponential distribution is:

$$l(\eta) = -n \log \eta - \frac{1}{\eta} \sum_{i=1}^n x_i$$

In order to maximize it, we compute its derivative:

$$\frac{dl}{d\eta} = -\frac{n}{\eta} + \frac{1}{\eta^2} \sum_{i=1}^n x_i$$

The derivative vanishes at  $\eta = \frac{1}{n} \sum_{i=1}^n x_i$ , and is positive for smaller  $\eta$ 's, and negative for larger. Therefore

$$\hat{\eta}_{\text{ML}} = \frac{1}{n} \sum_{i=1}^n x_i$$

is the parameter that maximizes the likelihood.

(c) The log-likelihood of the normal distribution under the dataset is:

$$l(\mu, \theta) = -\frac{n}{2} \log(2\pi) - n \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^n (x_i - \mu)^2$$

Because the log-likelihood approaches  $-\infty$  if  $\mu \rightarrow \pm\infty$ ,  $\sigma \rightarrow 0$ , or  $\sigma \rightarrow \infty$ , the log-likelihood is maximized for some finite  $(\hat{\mu}_{\text{ML}}, \hat{\sigma}_{\text{ML}})$ ,  $\hat{\sigma}_{\text{ML}} > 0$ . At the maximum

$$\left( \frac{\partial l}{\partial \mu}(\hat{\mu}_{\text{ML}}, \hat{\sigma}_{\text{ML}}), \frac{\partial l}{\partial \sigma}(\hat{\mu}_{\text{ML}}, \hat{\sigma}_{\text{ML}}) \right) = 0$$

The first partial derivative  $\frac{\partial l}{\partial \mu} = \frac{1}{\sigma^2} \sum_{i=1}^n (x_i - \mu)$  vanishes at  $\hat{\mu}_{\text{ML}} = \frac{1}{n} \sum_{i=1}^n x_i$ .

The second partial derivative  $\frac{\partial l}{\partial \sigma} = -n/\sigma + \sigma^{-3} \sum_{i=1}^n (x_i - \mu)^2$  vanishes when  $\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$ . Thus:

$$\begin{aligned} \hat{\mu}_{\text{ML}} &= \frac{1}{n} \sum_{i=1}^n x_i \\ \hat{\sigma}_{\text{ML}} &= \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \hat{\mu}_{\text{ML}})^2} \end{aligned}$$

■

To assess how well an estimator  $\hat{\theta}$  recovers the underlying value of the parameter  $\theta$ , we study its *bias* and *variance*. The bias is defined by the expectation of the deviation from the true value under the true distribution of the sample  $(X_1, X_2, \dots, X_n)$ :

$$\text{bias}(\hat{\theta}) = \mathbb{E}_{X_i \sim P(X|\theta)} \left[ \hat{\theta}(X_1, X_2, \dots, X_n) - \theta \right]$$

Biased (i.e. with a non-zero bias) estimators systematically under-estimate or over-estimate the parameter.

The variance of the estimator

$$\text{var}(\hat{\theta}) = \mathbb{E}_{X_i \sim P(X|\theta)} \left[ \left( \hat{\theta}(X_1, X_2, \dots, X_n) - \mathbb{E} \left[ \hat{\theta}(X_1, X_2, \dots, X_n) \right] \right)^2 \right]$$

measures the anticipated uncertainty in the estimated value due to the particular selection  $(x_1, x_2, \dots, x_n)$  of the sample. Note that the concepts of bias and variance of estimators are similar to the concepts of structural and approximation errors, respectively.

Estimators that minimize both bias and variance are preferred, but typically there is a trade-off between bias and variance.

2. (10 points) Show that  $\hat{a}_{\text{ML}}$  is biased (no need to compute the actual value of the bias),  $\hat{\eta}_{\text{ML}}$  and  $\hat{\mu}_{\text{ML}}$  are unbiased.

**Solution:**

- (a) The expected value of  $\hat{a}_{\text{ML}}$  is given by:

$$E_{p(x;a)} [\max\{X_1, X_2, \dots, X_n\}] = \int_{0 \leq x_1, x_2, \dots, x_n \leq a} a^{-n} \left( \max_{i=1}^n x_i \right) dx_1 dx_2 \dots dx_n$$

Since  $\max_{i=1}^n x_i$  is always  $\leq a$ , and there are regions of positive probability on which  $\max_{i=1}^n x_i$  is strictly less than  $a$  (for example  $0 \leq x_1, x_2, \dots, x_n \leq a/2$  occurs with positive probability), the expected value of  $\max_{i=1}^n x_i$  must be strictly less than  $a$ , and the estimator is *biased*.

- (b) First we show that (or simply state it as a property of the exponential distribution)

$$E_{p(x;\eta)} [X] = \eta$$

Proof:

$$\begin{aligned} \int_0^\infty p(x; \eta) x dx &= \int_0^\infty \frac{1}{\eta} e^{-\frac{x}{\eta}} x dx = - \int_0^\infty x \frac{d}{dx} e^{-\frac{x}{\eta}} dx = \\ &= -x e^{-x/\eta} \Big|_0^\infty + \int_0^\infty e^{-x/\eta} dx = -\eta \int_0^\infty \frac{d}{dx} e^{-x/\eta} dx = \\ &= -\eta e^{-x/\eta} \Big|_0^\infty = \eta \end{aligned}$$

It follows that

$$E[\hat{\eta}_{\text{ML}}] = E \left[ \frac{1}{n} \sum_{i=1}^n X_i \right] = \frac{1}{n} \sum_{i=1}^n E[X_i] = \eta$$

and the estimator is unbiased.

- (c) As in the previous part, for a Gaussian distribution we have

$$E_{p(x;\mu,\sigma)} [X] = \mu$$

If follows that

$$E[\hat{\mu}_{\text{ML}}] = E \left[ \frac{1}{n} \sum_{i=1}^n X_i \right] = \frac{1}{n} \sum_{i=1}^n E[X_i] = \mu$$

and the estimator is unbiased.

■

3. (optional) Show that  $\hat{\sigma}_{\text{ML}}^2$ , equal to the sample variance  $\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$ , is biased. Show that the ML estimator of the variance becomes unbiased after multiplication with  $n/(n-1)$ . Let  $\hat{\sigma}_{n-1}^2$  be this new estimator.

**Solution:** Let's compute the bias of the ML estimator of the variance by expanding the square under the sum:

$$\begin{aligned}
E[\hat{\sigma}_{ML}^2] &= E\left[\frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2\right] = \frac{1}{n} E\left[\sum_{i=1}^n X_i^2 - 2\bar{X} \sum_{i=1}^n X_i + n\bar{X}^2\right] = \\
&= \frac{1}{n} E\left[\sum_{i=1}^n X_i^2 - \frac{1}{n} \left(\sum_{i=1}^n X_i\right)^2\right] = \\
&= \frac{1}{n} E\left[\sum_{i=1}^n X_i^2 - \frac{1}{n} \left(\sum_{i=1}^n X_i^2 + 2 \sum_{1 \leq i < j \leq n} X_i X_j\right)\right] = \\
&= \frac{1}{n} \left[ \left(1 - \frac{1}{n}\right) \sum_{i=1}^n E[X_i^2] - \frac{2}{n} \sum_{1 \leq i < j \leq n} E[X_i X_j] \right]
\end{aligned}$$

Because the  $n$  samples are drawn independently, for  $i \neq j$  we have  $E[X_i X_j] = E[X_i] E[X_j] = E[X]^2$ . Similarly,  $E[X_i^2]$  is the same for all  $i$ . It follows that:

$$E[\hat{\sigma}_{ML}^2] = \frac{n-1}{n} (E[X^2] - E[X]^2) = \frac{n-1}{n} \sigma^2$$

Thus  $\hat{\sigma}_{ML}^2$  is biased, but  $E[\hat{\sigma}_{ML}^2 n/(n-1)] = \sigma^2$  is the expectation of an unbiased estimator. ■

4. (optional) A standard way to balance the tradeoff between bias and variance is to choose estimators of lower *mean squared error*:  $MSE(\hat{\theta}) = E_{X_i \sim P(X|\theta)} [(\hat{\theta} - \theta)^2]$ . Show that  $MSE(\hat{\theta}) = \text{bias}(\hat{\theta})^2 + \text{var}(\hat{\theta})$  and that  $MSE(\hat{\sigma}_{ML}^2) < MSE(\hat{\sigma}_{n-1}^2)$  even though  $\hat{\sigma}_{ML}^2$  is biased.

**Solution:** First let's split  $MSE(\hat{\theta})$  into bias and variance:

$$\begin{aligned}
MSE(\hat{\theta}) &= E_{X_i \sim P(X|\theta)} [(\hat{\theta} - \theta)^2] = E\left[\left(\hat{\theta} - E[\hat{\theta}] + E[\hat{\theta}] - \theta\right)^2\right] = \\
&= E\left[\left(\hat{\theta} - E[\hat{\theta}]\right)^2\right] + 2E[\hat{\theta} - E[\hat{\theta}]](E[\hat{\theta}] - \theta) + (E[\hat{\theta}] - \theta)^2
\end{aligned}$$

The first term in the right-hand side is the definition of  $\text{var}(\hat{\theta})$ ; the second term is 0 because  $E[\hat{\theta} - E[\hat{\theta}]] = 0$ ; finally, the third term is  $\text{bias}(\hat{\theta})^2$ . Therefore  $MSE(\hat{\theta}) = \text{bias}(\hat{\theta})^2 + \text{var}(\hat{\theta})$ .

To prove the second assertion we compute:

$$\begin{aligned}
MSE(\hat{\sigma}_{n-1}^2) - MSE(\hat{\sigma}_{ML}^2) &= E[(\hat{\sigma}_{n-1}^2 - \sigma^2)^2 - (\hat{\sigma}_{ML}^2 - \sigma^2)^2] = \\
&= E[(\hat{\sigma}_{n-1}^2)^2] - E[(\hat{\sigma}_{ML}^2)^2] - 2\sigma^2 (E[\hat{\sigma}_{n-1}^2] - E[\hat{\sigma}_{ML}^2]) = \\
&= \left(\frac{n^2}{(n-1)^2} - 1\right) E[(\hat{\sigma}_{ML}^2)^2] - \frac{2}{n} \sigma^4
\end{aligned}$$

Thus to show that  $\text{MSE}(\hat{\sigma}_{n-1}^2) > \text{MSE}(\hat{\sigma}_{\text{ML}}^2)$  we need to prove that:

$$\text{E} [(\hat{\sigma}_{\text{ML}}^2)^2] > \frac{2(n-1)^2}{n(2n-1)} \sigma^4$$

For a Gaussian distribution we can evaluate  $\text{E} [(\hat{\sigma}_{\text{ML}}^2)^2]$  directly:

$$\begin{aligned} \text{E} [(\hat{\sigma}_{\text{ML}}^2)^2] &= \text{E} \left[ \frac{1}{n^2} \left( \sum_{i=1}^n (X_i - \bar{X})^2 \right)^2 \right] = \\ &= \frac{1}{n^2} \text{E} \left[ \left( \frac{n-1}{n} \sum_{i=1}^n X_i^2 - \frac{2}{n} \sum_{1 \leq i < j \leq n} X_i X_j \right)^2 \right] = \\ &= \frac{1}{n^3} \left\{ (n-1)^2 \text{E} [X^4] + [2(n-1) + (n-1)^3] \text{E} [X^2]^2 + \right. \\ &\quad + (n-1)(n-2)(n-3) \text{E} [X]^4 - 2(n-1)(n-2)(n-3) \text{E} [X^2] \text{E} [X]^2 - \\ &\quad \left. - 4(n-1)^2 \text{E} [X^3] \text{E} [X] \right\} \end{aligned}$$

Because  $X$  is normal of mean  $\mu$  and variance  $\sigma^2$ , we have

$$\begin{aligned} \text{E} [X] &= \mu \\ \text{E} [X^2] &= \mu^2 + \sigma^2 \\ \text{E} [X^3] &= \mu^3 + 3\mu\sigma^2 \\ \text{E} [X^4] &= \mu^4 + 6\mu^2\sigma^2 + 3\sigma^4 \end{aligned}$$

It follows that

$$\text{E} [(\hat{\sigma}_{\text{ML}}^2)^2] = \left( 1 - \frac{1}{n^2} \right) \sigma^4$$

It is easy to verify that  $1 - 1/n^2 > 2(n-1)^2/(n(2n-1))$  for all  $n > 1$ , therefore  $\text{MSE}(\hat{\sigma}_{n-1}^2) > \text{MSE}(\hat{\sigma}_{\text{ML}}^2)$ . ■

## Problem 2: Maximum A-Posteriori Estimation

We want to determine the bias of an unfair coin for “heads” or “tails” from observing the outcome of a series of tosses. We model the coin by a single parameter  $\theta$  that represents the probability of tossing heads.

Given  $n$  independent observed tosses  $\mathcal{D} = \{x_1, \dots, x_n\}$  out of which  $n_H$  are “heads”, the likelihood function is:

$$p(\mathcal{D}|\theta) = \theta^{n_H} (1-\theta)^{n-n_H}$$

1. (5 points) Show that  $\hat{\theta}_{\text{ML}} = n_H/n$ . Thus if we toss the coin only once and we see “tails” ( $n = 1$  and  $n_H = 0$ ), according to maximum likelihood flipping the coin should always result in “tails”.

**Solution:** To calculate  $\hat{\theta}_{\text{ML}}$  we maximize on  $[0, 1]$  the logarithm of the likelihood:

$$l(\theta) = n_H \log \theta + (n - n_H) \log(1 - \theta)$$

We compute the derivative of the log-likelihood:

$$\frac{dl}{d\theta} = \frac{n_H}{\theta} - \frac{n - n_H}{1 - \theta}$$

and solve for its zeros:

$$\frac{n_H}{\theta} = \frac{n - n_H}{1 - \theta} \Rightarrow \theta = \frac{n_H}{n}$$

If  $0 < n_H < n$  then  $l(\theta) \rightarrow -\infty$  if  $\theta \rightarrow 0$  or  $\theta \rightarrow 1$ . It follows that the maximum is achieved on  $(0, 1)$  and  $\hat{\theta}_{\text{ML}} = n_H/n$ . The special cases  $n_H = 0$  and  $n_H = n$  result in the same formula for the estimator.

Alternatively, one can apply Jensen’s inequality on the concave logarithm:

$$\frac{n_H}{n} \log \frac{\theta}{n_H} + \frac{n - n_H}{n} \log \left( \frac{1 - \theta}{n - n_H} \right) \leq \log \left[ \frac{n_H}{n} \frac{\theta}{n_H} + \frac{n - n_H}{n} \frac{1 - \theta}{n - n_H} \right] = \log \frac{1}{n}$$

with equality when

$$\frac{n_H}{\theta} = \frac{n - n_H}{1 - \theta}$$

■

While the maximum likelihood estimator is accurate on large training samples, if data is very scarce the estimated value is not that meaningful (on small samples the variance of the estimator is very high and it overfits easily). In contrast, in **Maximum A-Posteriori (MAP) estimation** we compensate for the lack of information due to limited observations with an *a priori* preference on the parameters based on prior knowledge we might have. In the case of the coin toss for instance, even without seeing any tosses we can assume the coin should be able to show both “heads” and “tails” ( $\theta \neq 0$ ).

We express the prior preference/knowledge about  $\theta$  by a distribution  $p(\theta)$  (the *prior*). Assuming that  $\theta$  and the observed sample are characterized by an underlying joint probability  $p(\theta, \mathcal{D})$ , we can use the Bayes rule to express our adjusted belief about the parameters after observing the trials (the *posterior*):

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

where  $p(\mathcal{D}) = \int p(\mathcal{D}|\theta')p(\theta')d\theta'$  normalizes the posterior. Maximization of the posterior distribution gives rise to the *Maximum A-Posteriori* (MAP) estimate of the parameters:

$$\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} p(\theta|\mathcal{D}) = \arg \max_{\theta} p(\mathcal{D}|\theta)p(\theta)$$

As in maximum likelihood, to compute the MAP estimate it is often easier to maximize the logarithm  $\log p(\theta) + \log p(\mathcal{D}|\theta)$ .

For the coin toss we will consider separately each of the following priors:

$$\text{Discrete : } p^1(\theta) = \begin{cases} 0.5 & \text{if } \theta = 0.5 \\ 0.5 & \text{if } \theta = 0.4 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$\text{Beta : } p^2(\theta) = \frac{1}{Z} \theta^{\alpha-1} (1-\theta)^{\beta-1} \quad (5)$$

Here  $\alpha$  and  $\beta$  are *hyperparameters* that should be given, not estimated, and  $Z$  is a normalization constant needed to make  $p^2(\theta)$  integrate to 1 whose actual value is not important.

2. (10 points) Prior  $p^1(\theta)$  translates into a strong belief that the coin is either fair, or biased towards “tails” with a “heads” probability of 0.4. Express the MAP estimate  $\hat{\theta}^1_{\text{MAP}}$  under this prior as a function of  $n_H/n$ .

**Solution:** Since the prior assigns positive probability only to  $\theta = 0.4$  or  $\theta = 0.5$ , to maximize the posterior we need to compare only two quantities:

$$\hat{\theta}_{\text{MAP}} = \arg \max_{\theta \in \{0.4, 0.5\}} \theta^{n_H} (1-\theta)^{n-n_H} 0.5$$

Thus  $\hat{\theta}_{\text{MAP}} = 0.5$  when

$$\begin{aligned} 0.4^{n_H} 0.6^{n-n_H} &< 0.5^{n_H} 0.5^{n-n_H} \text{ or} \\ 0.4^{n_H/n} 0.6^{1-n_H/n} &< 0.5 \\ \frac{n_H}{n} \log 0.4 + \left(1 - \frac{n_H}{n}\right) \log 0.6 &< \log 0.05 \\ \log \frac{0.6}{0.5} &< \frac{n_H}{n} \log \frac{0.6}{0.4} \end{aligned}$$

It follows that the maximum-a-posteriori estimate is given by

$$\hat{\theta}_{\text{MAP}} = \begin{cases} 0.4 & \text{if } n_H/n < \log 1.2 / \log 1.5 \\ 0.5 & \text{if } n_H/n > \log 1.2 / \log 1.5 \end{cases}$$

The boundary case  $n_H/n = \log 1.2 / \log 1.5$  never occurs because the right-hand side is irrational. ■

3. (10 points) The Beta prior expresses the belief that  $\theta$  is likely to be near  $\alpha/(\alpha + \beta)$ . The larger  $\alpha + \beta$  is, the more peaked the prior, and the stronger the bias that  $\theta$  is close to  $\alpha/(\alpha + \beta)$ . Derive  $\hat{\theta}_{\text{MAP}}^2$  under the Beta prior and show that when  $n$  approaches infinity the MAP estimate approaches the ML estimate, thus the prior becomes irrelevant given a large number of observations.

**Solution:** The MAP estimate maximizes the following posterior:

$$\hat{\theta}_{\text{MAP}} = \arg \max_{\theta \in [0,1]} \theta^{n_H + \alpha - 1} (1 - \theta)^{n - n_H + \beta - 1}$$

Note that the posterior is identical to the likelihood of the coin after observing a series of exactly  $n_H + \alpha - 1$  heads out of a total of  $n + \alpha + \beta - 2$  tosses. We have already derived the ML estimator in Part 1, therefore:

$$\hat{\theta}_{\text{MAP}} = \frac{n_H + \alpha - 1}{n + \alpha + \beta - 2}$$

As  $n \rightarrow \infty$  we have:

$$\lim_{n \rightarrow \infty} \frac{\hat{\theta}_{\text{MAP}}}{\hat{\theta}_{\text{ML}}} = \lim_{n \rightarrow \infty} \frac{1 + \frac{\alpha - 1}{n_H}}{1 + \frac{\alpha + \beta - 2}{n}} = 1 + \frac{\alpha - 1}{\lim_{n \rightarrow \infty} n_H} = 1$$

(If  $\theta = 0$  the last equality is not true. In that case  $n_H$  must always be 0, therefore  $\hat{\theta}_{\text{MAP}} = (\alpha - 1)/(n + \alpha + \beta - 2) \rightarrow 0 = \hat{\theta}_{\text{ML}}$ .  $\blacksquare$

4. (optional) Compare qualitatively  $\hat{\theta}_{\text{MAP}}^1$  and  $\hat{\theta}_{\text{ML}}$ . Assuming that the coin has a true “heads” probability of 0.41, which of the two estimators is likely to learn it faster? If data is sufficient, which of the two estimators is better?

**Solution:**  $\hat{\theta}_{\text{MAP}}^1$  is able to get close to  $\theta = 0.41$  with fewer samples because it only needs to rule out  $\theta = 0.5$  out of the two feasible choices for the parameter  $\theta$ . However, if data is sufficient  $\hat{\theta}_{\text{ML}}$  is better because the MAP estimate will never be able to get closer than 0.01 to the true value of the parameter, whereas in the limit  $\hat{\theta}_{\text{ML}}$  recovers 0.41 exactly.  $\blacksquare$

## Part II: Polynomial Regression

### Problem 3

In this problem, we explore the behavior of polynomial regression methods when only a small amount of training data is available. We use polynomial regression models of the form

$$y = w_0 + w_1 x + w_2 x^2 + \dots + w_m x^m + \epsilon \quad (6)$$

$$= \mathbf{x}^T \mathbf{w} + \epsilon \quad (7)$$

where  $\epsilon \sim N(0, \sigma^2)$  (zero mean Gaussian noise) and  $\mathbf{x} = [1 \ x \ x^2 \ \dots \ x^m]^T$ . In a matrix form for all the training outputs, the model can be written as

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{e} \quad (8)$$

where  $\mathbf{y} = [y_1, \dots, y_n]^T$ ,  $\mathbf{X} = [\mathbf{x}_1^T; \dots; \mathbf{x}_n^T]$  depends on the polynomial order, and  $\mathbf{e} \sim N(\mathbf{0}, \sigma^2 \mathbf{I})$ . In other words, the outputs  $\mathbf{y}$  are normally distributed with mean vector  $\mathbf{X}\mathbf{w}$  and covariance matrix  $\sigma^2 \mathbf{I}$ . The likelihood of the outputs, given the inputs, can therefore be expressed as

$$p(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2) = N(\mathbf{y}; \mathbf{X}\mathbf{w}, \sigma^2 \mathbf{I}) \quad (9)$$

where  $N(\mathbf{y}; \mu, \Sigma)$  is a multi-variate (here  $n$ -variate) Gaussian

$$N(\mathbf{y}; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} \det(\Sigma)^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{y} - \mu)^T \Sigma^{-1} (\mathbf{y} - \mu) \right\} \quad (10)$$

We will begin by using a maximum likelihood estimation criterion for the parameters  $\mathbf{w}$  that reduces to least squares fitting.

1. Consider a simple 1D regression problem. The data in `housing.data` provides information of how 13 different factors affect house price in the Boston area. (Each column of data represents a different factor, and is described in brief in the file `housing.names`.) To simplify matters (and make the problem easier to visualise), we consider predicting the house price (the 14th column) from the LSTAT feature (the 13th column).

We split the data set into two parts (in `testLinear.m`), train on the first part and test on the second. We have provided you with the necessary MATLAB code for training and testing a polynomial regression model. Simply edit the script (`ps1_part2.m`) to generate the variations discussed below.

- (5 points) Use `ps1_part2.m` to calculate and plot training and test errors for polynomial regression models as a function of the polynomial order (from 1 to 7). Use 250 training examples (set `numtrain=250`).

**Solution:** See attached plot (Figure 1). ■

- (10 points) Briefly explain the qualitative behavior of the errors. Which of the regression models are over-fitting to the data? Provide a brief justification.

**Solution:** The training error is monotonically decreasing (non-increasing) with polynomial order. This is because higher order models can fully represent any lower order model by adequate setting of parameters, which in turn implies that the former can do no worse than the latter when fitting to the same training data.

(Note that this monotonicity property need not hold if the training sets to which the higher and lower order models were fit were *different*, even if these were drawn from the same underlying distribution.)

The test error mostly decreases with model order till about 5th order, and then increases. This is an indication (but not proof) that higher order models (6th and 7th) might be overfitting to the data. Based on these results, the best choice of model for training on the given data is the 5th order model, since it has lowest error on an independent test set of around 250 examples. ■

- (c) (10 points) Rerun `ps1_part2.m` with only 50 training examples (set `numtrain=50`). Briefly explain key differences between the resulting plot and the one from part a). Which of the models are over-fitting this time?

**Solution:** See attached plot (Figure 1).

We note the following differences between the plots for 250 and 50 examples:

- The training errors are lower in the present case. This is because we are having to fit fewer points with the same model. In this examples, in particular, we are fitting only a subset of the points we were previously fitting (since there is no randomness in drawing points for training).
- The test errors for most models are higher. This is evidence of systematic overfitting for all model orders, relative to the case where there were many more training points.
- The model with the lowest test error is now the third order model. From 4th order onwards, the test error generally increases (though the 7th order is an exception, perhaps due to the particular choice of training and test sets). This tells us that with fewer training examples, our preference should switch towards lower-order models (in the interest of achieving low generalisation error), even though the *true* model responsible for generating the underlying data might be of much higher order. This relates to the trade-off between bias and variance. We typically want to minimise the mean-square error, which is the sum of the bias and variance. Low-order models typically have high bias but low variance. Higher order models may be unbiased, but have higher variance.

There are many ways of trying to avoid over-fitting. One way is to use a maximum a posteriori (MAP) estimation criterion rather than maximum likelihood. MAP criterion allows us to penalize parameter choices that we would not expect to lead to good generalization. For example, very large parameter values in linear regression make predictions very sensitive to slight variations in the inputs. We can express a preference against such large parameter values by assigning a prior distribution over the parameters such as simple Gaussian

$$p(\mathbf{w}; \alpha^2) = \mathcal{N}(\mathbf{0}, \alpha^2 \mathbf{I}) \quad (11)$$

This prior decreases rapidly as the parameters deviate from zero. The single variance (hyper-parameter)  $\alpha^2$  controls the extent to which we penalize large parameter values.

This prior needs to be combined with the likelihood to get the MAP criterion. The MAP parameter estimate maximizes

$$\log(p(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2)p(\mathbf{w}; \alpha^2)) = \log p(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2) + \log p(\mathbf{w}; \alpha^2) \quad (12)$$

The resulting parameter estimates are biased towards zero due to the prior. We can find these estimates as before by setting the derivatives to zero.

2. (15 points) Show that

$$\hat{\mathbf{w}}_{MAP} = (\mathbf{X}^T \mathbf{X} + \frac{\sigma^2}{\alpha^2} \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}, \quad (13)$$

**Solution:** Given training input matrix  $X$  and column vectors for weights  $w$  and training targets  $y$ , our objective function is

$$\begin{aligned} Q &= \log(p(y|Xw, \sigma^2)) + \log(p(w; \alpha^2)) \\ &= \frac{-1}{2}(y - Xw)^T (\sigma^2 I)^{-1} (y - Xw) \frac{-1}{2} w^T (\alpha^2 I)^{-1} w \\ &\quad + \text{constant (w.r.t. } w) \\ &= \frac{-1}{2\sigma^2} (y^T y - 2y^T Xw - w^T X^T Xw) + \frac{-1}{2\alpha^2} (w^T w) + \text{const.} \end{aligned}$$

Differentiating with respect to weights,

$$\begin{aligned} \frac{\partial Q}{\partial w} &= 0 \\ \Rightarrow \frac{-1}{2\sigma^2} (2X^T y - 2X^T Xw) + \frac{-1}{2\alpha^2} (2w) &= 0 \\ \Rightarrow \left( \frac{X^T X}{\sigma^2} + \frac{I}{\alpha^2} \right) w &= \frac{X^T y}{\sigma^2} \\ \Rightarrow (X^T X + \frac{\sigma^2}{\alpha^2} I) w &= X^T y \\ \Rightarrow w &= (X^T X + \frac{\sigma^2}{\alpha^2} I)^{-1} X^T y. \end{aligned}$$

(Note: we should also take the second derivative to make sure we truly have a maximum. For the sake of this problem set, we did not require you to perform this check.)

■

3. (5 points). In the above solution, show that in the limit of infinitely large  $\alpha$ , the MAP estimate is equal to the ML estimate, and explain why this happens

**Solution:** As  $\alpha \rightarrow \infty$ , the variance ratio approaches zero, so  $w = (X^T X)^{-1} X^T y$ .

In terms of regularisation, this means that the prior has infinite variance (about mean zero), and is therefore flat. All weight vectors are equally likely *a priori*, and we are back to the ML setting where the weights are determined solely by the data. ■

4. Let us see how the MAP estimate changes our solution in the housing-price estimation problem. The MATLAB code you used above actually contains a variable corresponding to the variance ratio  $\text{var\_ratio} = \frac{\sigma^2}{\alpha^2}$  for the MAP estimator. This has been set to a default value of zero to simulate the ML estimator discussed in class. In this part, you should vary this value from  $1e-8$  to  $1e-4$  in multiples of 10 (*i.e.*  $1e-8, 1e-7, \dots, 1e-4$ ). A larger ratio corresponds to a stronger prior (smaller values of  $\alpha^2$  constrain the parameters  $\mathbf{w}$  to lie closer to origin).

(a) (10 points) Plot the training and test errors as a function of the polynomial order using the above 5 MAP estimators and 250 and 50 training points.

**Solution:** See attached plots (Figures 2 and 3). ■

(b) (10 points) Describe how the prior affects the estimation results.

**Solution:** We make the following observations:

- As the variance ratio (*i.e.* the strength of the prior) increases, the training error increases (slightly). This is because we are no longer solely interested in obtaining the best fit to the training data.
  - The test error for higher order models decreases dramatically with strong priors. This is because we are no longer allowing these models to overfit to training data by restricting the range of weights possible.
  - Test error generally decreases with increasing prior.
  - As a consequence of the above two points, the *best* model changes slightly with increasing prior in the direction of more complex models.
  - For 50 training samples, the difference in test error between ML and MAP is more significant than with 250 training examples. This is because overfitting is a more serious problem in the former case.
-

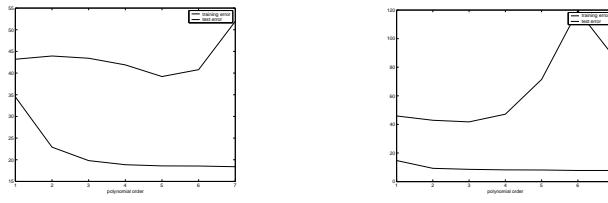


Figure 1: Plots for 3.1(a) (left) and 3.1(c) (right).

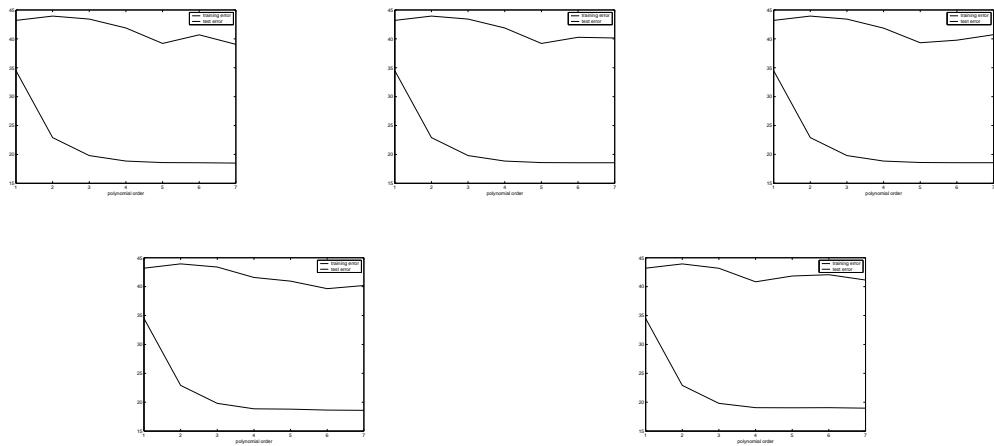


Figure 2: Plots for 3.4(b), for 250 training examples. Left to right, (then) top to bottom, variance ratio = 1e-8 to 1e-4

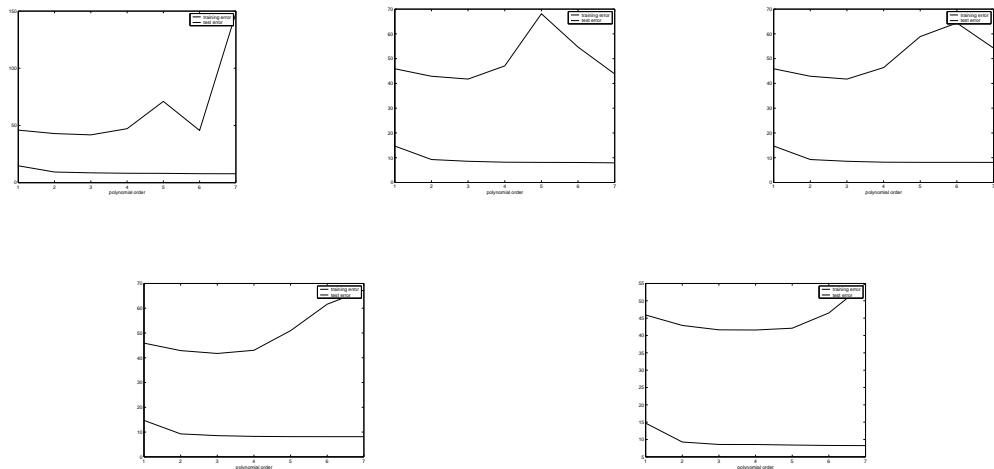


Figure 3: Plots for 3.4(b), for 50 training examples. Left to right, (then) top to bottom, variance ratio = 1e-8 to 1e-4

# 10702/36702 Statistical Machine Learning, Spring 2008

## Homework 2 Solutions

February 20, 2008

### 1 [15 points]

(a) Let  $\pi_n$  be a sequence of priors and  $\tilde{\theta}_n$  the corresponding Bayes estimators. Suppose that

$$\int R(\theta, \tilde{\theta}_n) \pi_n(\theta) d\theta \rightarrow c$$

for some finite  $c$ . Suppose that  $\hat{\theta}$  is an estimator such that

$$\sup_{\theta} R(\theta, \hat{\theta}) \leq c$$

Show that  $\hat{\theta}$  is minimax.

★ SOLUTION: For any estimator  $T$

$$\begin{aligned} \sup_{\theta} R(\theta, T) &\geq \int R(\theta, T) \pi_n(\theta) d\theta \\ &\geq \int R(\theta, \tilde{\theta}_n) \pi_n(\theta) d\theta \end{aligned}$$

Let  $n$  goes to infinity on both sides of the inequality, we have

$$\lim_{n \rightarrow \infty} \sup_{\theta} R(\theta, T) = \sup_{\theta} R(\theta, T) \geq \lim_{n \rightarrow \infty} \int R(\theta, \tilde{\theta}_n) \pi_n(\theta) d\theta = c \geq \sup_{\theta} R(\theta, \hat{\theta})$$

Therefore,  $\hat{\theta}$  is minimax.

(a) Let  $X \sim N(\theta, 1)$ . Show that  $\hat{\theta} = X$  is minimax.

Hint: Let  $\pi_n$  be  $N(0, n)$ . Check that

$$\int R(\theta, \hat{\theta}_n) \pi_n(\theta) d\theta \rightarrow 1$$

Next show that  $R(\theta, X) = 1$ . Consider from part (a) that  $X$  is minimax.

★ SOLUTION: Let  $\pi_n$  be  $N(0, n)$ . The posterior distribution

$$\begin{aligned} \pi(\theta | X = x) &\propto \exp\left(-\frac{(x - \theta)^2}{2}\right) \cdot \exp\left(-\frac{\theta^2}{2n}\right) \\ &\propto \exp\left(-\frac{(n+1)(\theta - \frac{n}{n+1}x)^2}{2n}\right) \end{aligned}$$

Therefore,  $\pi(\theta|X=x) \sim N(\frac{n}{n+1}X, \frac{n}{n+1})$ , and the Bayes estimator  $\tilde{\theta}_n = \frac{n}{n+1}X$ . The risk

$$R(\theta, \tilde{\theta}_n) = E_\theta(\theta - \tilde{\theta}_n)^2 = (E_\theta \tilde{\theta}_n - \theta)^2 + Var_\theta(\tilde{\theta}_n) = \frac{\theta^2 + n^2}{(n+1)^2}$$

The Bayes risk

$$\int R(\theta, \tilde{\theta}_n) \pi_n(\theta) d\theta = \int \frac{\theta^2 + n^2}{(n+1)^2} \cdot \frac{\exp(-\frac{\theta^2}{2n})}{\sqrt{2\pi n}} d\theta = \frac{n + n^2}{(n+1)^2} = \frac{n}{n+1}$$

Therefore, as  $n \rightarrow \infty$ , the Bayes risk goes to 1. On the other hand,

$$R(\theta, X) = (E_\theta(X) - \theta)^2 + Var_\theta X = 1$$

Therefore,  $\sup_\theta R(\theta, X) = 1$ . According to (a),  $X$  is minimax.

## 2 [16 points]

The following is a list of some loss functions commonly used for large-margin classification algorithms. For each loss function  $\phi(x)$  determine whether  $\phi$  is a convex function, and then calculate its conjugate  $\phi^*$ . Plot  $\phi$  and  $\phi^*$ .

(a) Exponential loss:  $\phi(x) = \exp(-x)$

★ SOLUTION:  $\phi''(x) = \exp(-x) \succeq 0 \quad \forall x$ . Hence  $\phi(x)$  is convex.

Conjugate function:  $\phi^*(y) = \sup_x (xy - \exp(-x))$ .

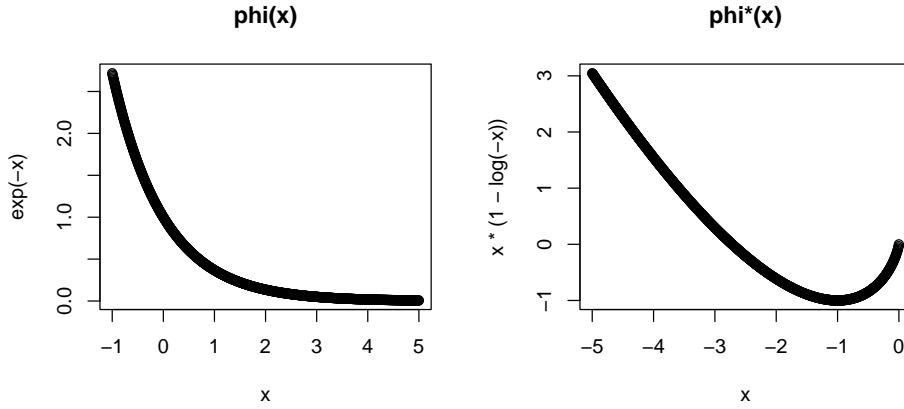
For  $y > 0$ ,  $\phi^*(y)$  is unbounded.

For  $y = 0$ ,  $\phi^*(y) = 0$ .

For  $y < 0$ ,  $\frac{\partial}{\partial x}(xy - \exp(-x)) = y - \exp(-x) = 0$ .

Or,  $x^* = -\log(-y)$ . Substituting  $x$ , we get  $\phi^*(y) = -y\log(-y) + y$ .

$$\phi^*(y) = \begin{cases} \infty & y > 0 \\ 0 & y = 0 \\ -y\log(-y) + y & y < 0 \end{cases}$$



(b) Truncated quadratic loss:  $\phi(x) = [\max(1-x, 0)]^2$

★ SOLUTION:  $1 - x$  and  $0$  are both convex. Hence  $\max[(1 - x), 0]$  is also convex. Square of convex function also convex. Hence  $\phi(x)$  convex.

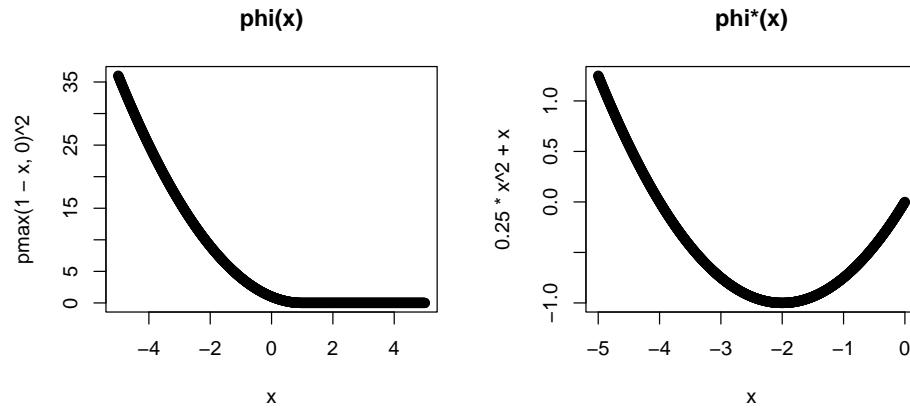
Conjugate computation:

For  $y > 0$ ,  $\phi^*(y)$  is unbounded since  $x^* \rightarrow \infty$ .

Consider  $y \leq 0$ . Then for  $x \leq 1$  we have  $\phi(x) = (1 - x)^2$  and  $\phi^*(y) = \sup(xy - (1 - x)^2)$ . To find the maximum we differentiate to get  $y + 2(1 - x) = 0$  or  $x^* = 1 + y/2$ . Substituting for  $x$  in  $\phi^*(y)$  we get  $\phi^*(y) = (1 + y/2)y - y^2/4 = y^2/4 + y$ . For  $x > 1$ , the conjugate is unbounded.

Hence we have the conjugate as:

$$\phi^*(y) = \begin{cases} y^2/4 + y & y \leq 0 \\ \infty, & \text{otherwise} \end{cases}$$



(c) Hinge loss:  $\phi(x) = \max(1 - x, 0)$

★ SOLUTION:  $\phi(x)$  is convex. see part (b).

Conjugate computation:

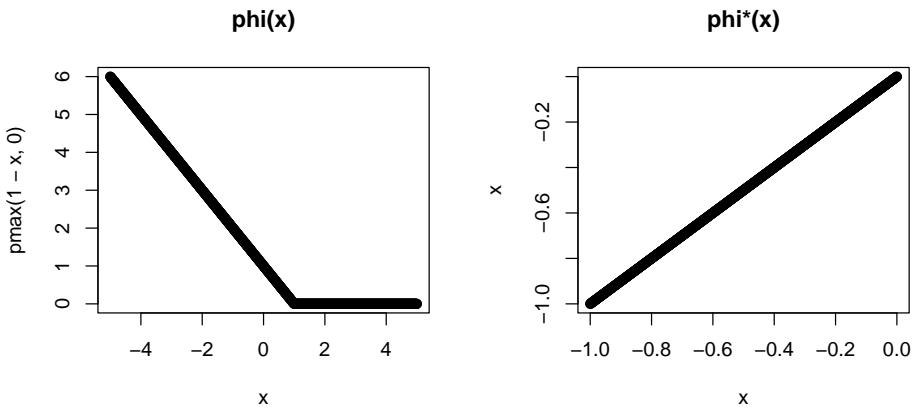
For  $y > 0$ ,  $\phi^*(y)$  is unbounded since  $x^* \rightarrow \infty$ .

For  $y < -1$ ,  $\phi^*(y)$  is unbounded since  $x^* \rightarrow -\infty$ .

Consider the case when  $-1 \leq y \leq 0$ . We have  $\phi(x) = (1 - x)$  when  $x \leq 1$  and  $0$  otherwise. Hence,  $\phi^*(y) = \sup(xy - (1 - x))$ . Differentiating we get,  $y + 1 = 0$  or  $y = -1$ . Or,  $\phi^*(y) = y$

Hence we have the conjugate as:

$$\phi^*(y) = \begin{cases} y & -1 \leq y \leq 0 \\ \infty, & \text{otherwise} \end{cases}$$



(d) Sigmoid loss:  $\phi(x) = 1 - \tanh(\kappa x)$ , for fixed  $\kappa > 0$

★ SOLUTION:  $\phi''(x) = 2\kappa^2 \operatorname{sech}^2(\kappa x) \tanh(\kappa x)$ . Hence,  $\phi''(x)$  has the same sign as  $\tanh(\kappa x)$  which can be positive or negative. Hence  $\phi(x)$  is not convex.

Conjugate computation:

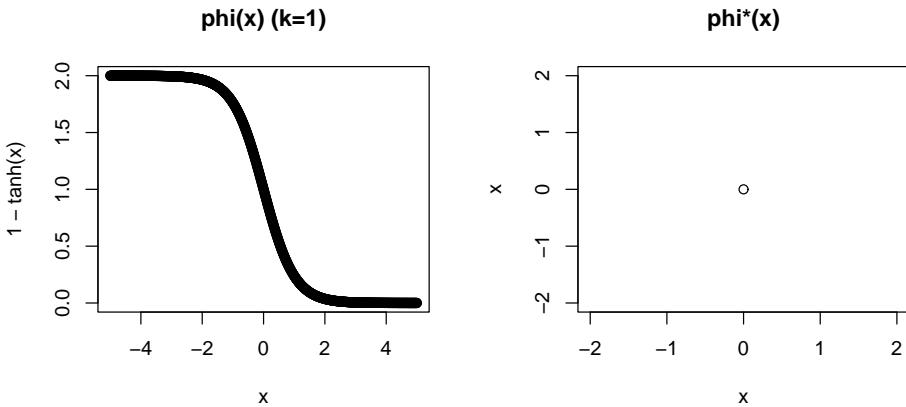
For  $y > 0$ ,  $\phi^*(y)$  is unbounded since  $x^* \rightarrow \infty$ .

For  $y < 0$ ,  $\phi^*(y)$  is unbounded since  $x^* \rightarrow -\infty$ .

Only when  $y = 0$ , we have  $\phi^*(y) = -1 + \tanh(\kappa x)$ . Since hyperbolic tan is bound between -1 and 1, we have  $\phi^*(0) = 0$ .

Hence we have the conjugate as:

$$\phi^*(y) = \begin{cases} 0 & y = 0 \\ \infty, & \text{otherwise} \end{cases}$$



### 3 [14 points]

If  $f(x, y) = f_1(x) + f_2(y)$ , with  $f_1$  and  $f_2$  convex, show that

$$f^*(x, y) = f_1^*(x) + f_2^*(y)$$

Does this hold if  $f_1$  and  $f_2$  are not convex?

★ SOLUTION:

$$\begin{aligned}
f^*(x, y) &= \sup(< u, v > < x, y >' - f(u, v)) \\
&= \sup(ux + vy - f(u) - f(v)) \\
&= \sup(ux - f(u)) + \sup(vy - f(y)) \\
&= f^*(x) + f^*(y)
\end{aligned}$$

Since we didn't need the convexity condition, this also holds for non-convex functions.

## 4 [15 points]

The following is called the *probit regression model*. Suppose  $Y \in \{0, 1\}$  is a random variable given by

$$Y = \begin{cases} 1 & a^T X + b + V \leq 0 \\ 0 & a^T X + b + V > 0 \end{cases}$$

where  $X \in \mathbb{R}^p$  is a vector of explanatory variables and  $V \sim N(0, 1)$  is a latent (unobserved) random variable. Formulate the maximum likelihood estimation problem of estimating  $a$  and  $b$ , given data consisting of pairs  $(X_i, Y_i)$ ,  $i = 1, \dots, n$ , as a convex optimization problem.

★ SOLUTION:

$$P(Y = 1|X) = P(a^T X + b + V \leq 0) = P(V \leq -a^T X - b) = \Phi(-a^T X - b)$$

where  $\Phi(\cdot)$  is the standard normal cdf. Therefore,  $P(Y = 0|X) = 1 - P(Y = 1|X) = \Phi(a^T X + b)$ .

The log-likelihood

$$\begin{aligned}
l(a, b) &= \sum_{i=1}^n [Y_i \log(P(Y = 1|X_i)) + (1 - Y_i) \log(P(Y = 0|X_i))] \\
&= \sum_{i=1}^n [Y_i \log \Phi(-a^T X - b) + (1 - Y_i) \log \Phi(a^T X + b)]
\end{aligned} \tag{1}$$

According to the notes on log-concavity,  $\Phi(\cdot)$  is log-concave. Thus,  $\log \Phi(\cdot)$  is a non-decreasing concave function. Since  $-a^T X - b$  and  $a^T X + b$  are concave functions of  $a$  and  $b$ ,  $\log \Phi(-a^T X - b)$  and  $\log \Phi(a^T X + b)$  are both concave functions of  $a$  and  $b$ . According to equation (1),  $l(a, b)$  is a non-negative weighted combination of concave functions. Therefore,  $l(a, b)$  is a concave function.

The maximum likelihood estimation problem is to maximize the concave function  $l(a, b)$ , which is equivalent to minimize the convex function  $-l(a, b)$ , so it is a convex optimization problem.

## 5 [15 points]

For  $x \in \mathbb{R}^n$  define the  $L_p$  norm

$$\|x\|_p = \left( \sum_{j=1}^n |x_j|^p \right)^{1/p}$$

for  $p > 0$ . Let

$$C = \{x : \|x\|_p \leq 1\}$$

Show that  $C$  is convex if and only if  $p \geq 1$ .

★ SOLUTION: First, we prove that if  $p \geq 1$ , then  $C$  is convex. Let  $f(y) = y^p$ , where  $y > 0$  and  $p \geq 1$ . It is easily verified that  $f(y)$  is a non-decreasing convex function of  $y$ . Let  $g(y) = |y|$ , where  $y \in \mathbb{R}$ .  $g(y)$  is also a convex function. Therefore,  $f(g(y)) = |y|^p$  is a convex function, where  $y \in \mathbb{R}$  and  $p \geq 1$ .

$\forall x^1, x^2 \in C$ ,  $\sum_{j=1}^n |x_j^1|^p \leq 1$  and  $\sum_{j=1}^n |x_j^2|^p \leq 1$ . According to Jensen's inequality,  $\forall c \in [0, 1]$

$$\sum_{j=1}^n |cx_j^1 + (1-c)x_j^2|^p \leq \sum_{j=1}^n [c|x_j^1|^p + (1-c)|x_j^2|^p] \leq c \sum_{j=1}^n |x_j^1|^p + (1-c) \sum_{j=1}^n |x_j^2|^p = 1$$

Therefore,  $cx^1 + (1-c)x^2 \in C$ , and  $C$  is convex.

Next, we prove that if  $p < 1$ , then  $C$  is not convex. If  $n = 1$ , it can be easily verified that  $C$  is not convex. If  $n > 1$ , let  $x^1 = [1, 0, \dots, 0]^T$ ,  $x^2 = [0, 1, \dots, 0]^T$  and  $c = 0.5$ .

$$\sum_{j=1}^n |cx_j^1 + (1-c)x_j^2|^p = 2 \times 0.5^p > 2 \times 0.5 = 1$$

Therefore,  $cx^1 + (1-c)x^2 \notin C$ , and  $C$  is not convex.

To summarize,  $C$  is convex if and only if  $p \geq 1$ .

## 6 [14 points]

Linear regression in R. Add brief comments to this code, and to the output, to explain what the code does and what the output means.

★ SOLUTION:

```
par(mfrow=c(2,2),bg='cornsilk')      # plots will be drawn on a 2x2 grid on cornsilk background
n = 100
sigma = 1
x = rnorm(n)                         # generate 'n' random numbers from N(0,1)
x = sort(x)
y = 5 + 3*x + rnorm(n,0,sigma)       # y is linear function of x with normal noise
plot(x,y,col="blue",lwd=3)            # plot (x,y) with blue color
out = lm(y~x)                        # fit a linear model
summary(out)                         # show summary of fitted linear model
abline(out,col="red",lwd=3)           # show predicted values of 'y' with 'red' lines
abline(a=5,b=3,col="green",lwd=2)     # show true value of 'y' with 'green' lines

y = 5 + 3*x + rcauchy(n,0,sigma)     # y is linear function of x with cauchy noise
plot(x,y,col="blue",lwd=3)
out = lm(y~x)
summary(out)
abline(out,col="red",lwd=3)
abline(a=5,b=3,col="green",lwd=2)

nsim = 100
b = rep(0,nsim)
for(i in 1:nsim){                   # generate random x and y 100 times with normal noise
  x = rnorm(n)                      # and record the predicted slope of regression line
  x = sort(x)                       # for each simulation
  y = 5 + 3*x + rnorm(n,0,sigma)
  out = lm(y~x)
```

```

b[i] = out$coef[2]
}
summary(b)
hist(b)                                # plot the histogram of predicted slope values
abline(v=3,lwd=3,col="red")              # show true value in the histogram
print(mean((b-3)^2))                   # print MSE value

#####
# Output of summary(out) with normal noise
#####
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q      Max
-2.91252 -0.63554 -0.04475  0.62950  2.61872

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.1733     0.1052   49.19   <2e-16 ***
x            3.0534     0.1128   27.08   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.037 on 98 degrees of freedom
Multiple R-Squared: 0.8821,    Adjusted R-squared: 0.8809
F-statistic: 733.2 on 1 and 98 DF,  p-value: < 2.2e-16

As seen above, the estimates of regression coefficients are 5.17 and
3.05. These are very close to the true values and very significant (<2e-16).
The plot shows that the ranlge of 'y' is very large as compared to normal.
#####

#####
# Output of summary(out) with cauchy noise
#####
Call:
lm(formula = y ~ x)

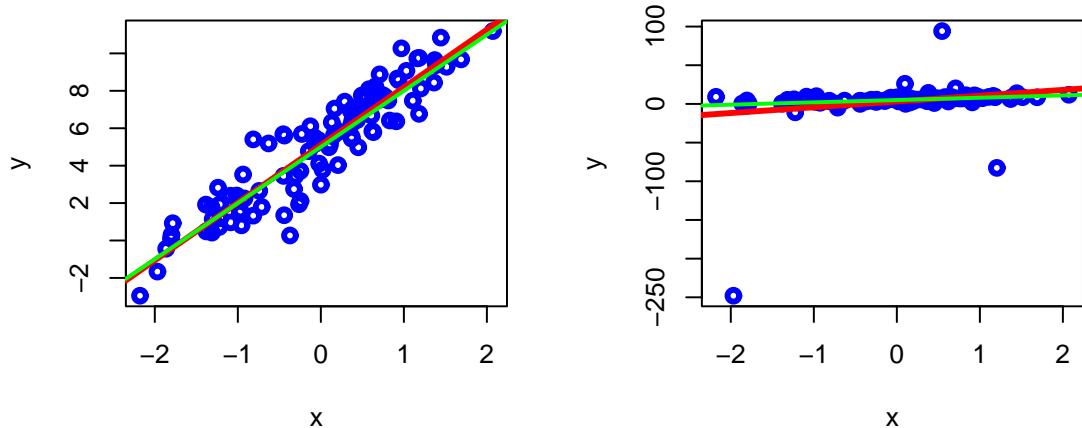
Residuals:
    Min      1Q  Median      3Q      Max
-20.2287 -0.9422  0.1061   1.0327  7.8915

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.0879     0.3379  15.058   < 2e-16 ***
x            2.6615     0.3683   7.227 1.09e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

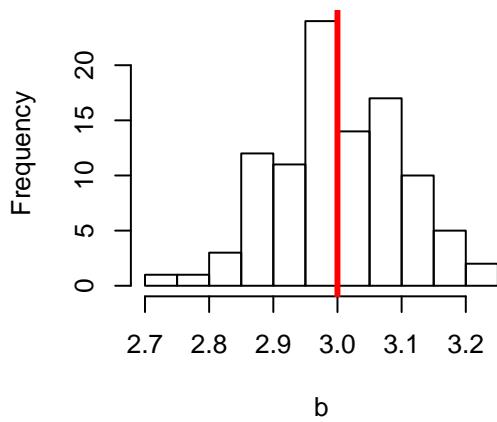
Residual standard error: 3.356 on 98 degrees of freedom
Multiple R-Squared: 0.3477,    Adjusted R-squared: 0.341
F-statistic: 52.23 on 1 and 98 DF,  p-value: 1.087e-10

```

As seen above, the estimates with cauchy noise are not as good as from normal noise. This is due to the fact that cauchy has heavier tails.



**Histogram of  $b$**



## 7 [14 points]

Prove the leave-one-out cross-validation identity:

$$\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_{(-i)})^2 = \frac{1}{n} \sum_{i=1}^n \left( \frac{Y_i - \hat{Y}_i}{1 - H_{ii}} \right)^2$$

★ SOLUTION: Consider  $Z$  such that

$$Z_j = \begin{cases} Y_j & j \neq i \\ \hat{Y}_j^{(-i)} & j = i \end{cases}$$

$\text{SSE} = \sum (Y_j^{(-i)} - Z_j)^2$ . This is also the minimal SSE for the regression on  $Y$  without the  $i^{th}$  data point.

$$\begin{aligned} \text{Hence, } \hat{Y}_i^{(-i)} &= (HZ)_i \\ &= \sum_{k \neq i} H_{ik} Z_k + H_{ii} Z_i \\ &= \sum H_{ik} Y_k - H_{ii} Y_i + H_{ii} \hat{Y}_i^{(-i)} \\ &= (HY)_i - H_{ii} Y_i + H_{ii} \hat{Y}_i^{(-i)} \\ &= \hat{Y}_i - H_{ii} Y_i + H_{ii} \hat{Y}_i^{(-i)} \\ &= \frac{\hat{Y}_i - H_{ii} Y_i}{1 - H_{ii}} \end{aligned}$$

Using this we get,

$$\frac{1}{n} \sum (Y_i - \hat{Y}_i^{(-i)})^2 = \frac{1}{n} \sum \left( \frac{Y_i - Y_i H_{ii} - \hat{Y}_i + H_{ii} Y_i}{1 - H_{ii}} \right)^2 = \frac{1}{n} \sum \left( \frac{Y_i - \hat{Y}_i}{1 - H_{ii}} \right)^2$$

Hence proved.

# 10-601 Machine Learning, Fall 2012

## Homework 2

Instructors: Tom Mitchell, Ziv Bar-Joseph

TA in charge: Selen Uguroglu  
email: sugurogl@cs.cmu.edu

### — SOLUTIONS —

## 1 Naive Bayes, 20 points

### Problem 1. Basic concepts, 10 points

Naive Bayes reduces the number of parameters that must be estimated for a Bayesian classifier, by making a conditional independence assumption when modeling  $P(X|Y)$ . The definition for conditional independence is the following:

*Definition:* Given random variables X, Y and Z, X is conditionally independent of Y given Z, denoted by  $X \perp Y|Z$ , if and only if :

$$P(X = x_i|Y = y_j, Z = z_k) = P(X = x_i|Z = z_k), \forall i, j, k \quad (1)$$

Given this definition, please answer the following questions:

a. (1 point) Given  $X \perp Y|Z$ , can we say  $P(X, Y|Z) = P(X|Z)P(Y|Z)$ ? Explain.

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

b. (1 point) Given  $X \perp Y|Z$ , can we say  $P(X, Y) = P(X)P(Y)$ ? Explain.

**SOLUTION:** No, it has to be conditioned on Z,  $P(X, Y) = P(X|Y)P(Y)$ ,  $P(X|Y)$  is not equal to  $P(X)$

c. (2 points) Suppose  $X$  is a vector of  $n$  boolean attributes and  $Y$  is a single discrete-valued variable that can take on  $J$  possible values.

Let  $\theta_{ij} = P(X_i|Y = y_j)$ . What is the number of independent  $\theta_{ij}$  parameters?

**SOLUTION:**  $nJ$

d. (2 points) Consider the same problem, but now suppose  $X$  is a vector of  $n$  real-valued attributes, where each of these  $X_i$  follows a Normal (Gaussian) distribution:  $P(X_i = x_i|Y = y_j) \sim N(x_i|\mu_{ij}, \sigma_{ij})$ . How many distinct  $\mu_{ij}, \sigma_{ij}$  are there?

**SOLUTION:**  $nJ$  pairs (  $nJ \mu_{ij}$  and  $nJ \sigma_{ij}$  )

We can write the classification rule for Naive Bayes as:

$$y^* = \operatorname{argmax}_{y_k} \frac{P(Y = y_k) \prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)} \quad (2)$$

e. (2 points) We often do not compute the denominator when estimating Y. Explain why.

**SOLUTION:** Denominator does not depend on  $y_j$ , so we don't have to compute it during maximization.

f. (2 points) Is it possible to calculate  $P(X)$  from the parameters estimated by Naive Bayes?

**SOLUTION:** Yes, NB is a generative classifier. We can obtain  $P(X)$  by marginalizing  $P(X|Y)$  over the class variable, e.g.  $P(X) = \sum_y P(X|Y = y)P(Y = y)$ .

**Problem 2. Parameter estimation for Naive Bayes, 10 points**

Whether  $X$  takes discrete or continuous inputs, Naive Bayes can be used for classification with the same conditional independence assumptions. In this question, we'll discuss how to estimate the parameters using MLE for both of the cases.

a. (4 points)

Let  $X = \langle X_1, X_2 \dots X_n \rangle$  be a vector of  $n$  Boolean values where the random variable  $X_i$  denotes the  $i^{th}$  attribute of  $X$ . Suppose we are interested in estimating the parameters for the first attribute  $X_1$ . We typically model  $P(X_1|Y = y_k)$  with a Bernoulli distribution:

$$P(X_1 = x_{1j} | Y = y_k) = \theta_{1k}^{x_{1j}} (1 - \theta_{1k})^{(1-x_{1j})} \quad (3)$$

where  $j = 1 \dots M$  refers to the  $j^{th}$  training instance ( $M$  is the number of training samples), and where  $x_{1j}$  refers to the value of  $X_1$  in the  $j^{th}$  training instance. Assume that the  $M$  training instances are independent and identically distributed (iid). Write down the MLE for  $\hat{\theta}_{1k}$ . (you need not derive it - just write it down).

**SOLUTION:**

$$P(X_{1j} = x_{1j} | \theta_{1k}) = \theta_{1k}^{x_{1j}} (1 - \theta_{1k})^{(1-x_{1j})} \text{ for one instance}$$

Lets write down the likelihood:

$$L(\theta_{1k}) = \prod_{j=1}^M P(X_{1j} | \theta_{1k})^{I(Y^j = y_k)}$$

where  $I(Y^j = y_k) = 1$ , if  $Y^j = y_k$  ;  $I(Y^j = y_k) = 0$  otherwise

Taking the log:

$$\begin{aligned} \ell(\theta_{1k}) &= \ln \prod_{j=1}^M P(X_{1j} | \theta_{1k})^{I(Y^j = y_k)} \\ &= \sum_{j=1}^M I(Y^j = y_k) \ln P(X_{1j} | \theta_{1k}) \\ &= \sum_{j=1}^M I(Y^j = y_k) \left[ x_{1j} \ln \theta_{1k} + (1 - x_{1j}) \ln (1 - \theta_{1k}) \right] \end{aligned}$$

taking derivative with respect to  $\theta_{1k}$ :

$$\frac{\partial \ell(\theta_{1k})}{\partial \theta_{1k}} = \sum_{j=1}^M I(Y^j = y_k) \left[ x_{1j} \frac{1}{\theta_{1k}} + (1 - x_{1j}) \frac{1}{(1 - \theta_{1k})} \right]$$

Setting to 0:

$$0 = \frac{1}{\theta_{1k}} \sum_{j=1}^M I(Y^j = y_k) x_{1j} + \frac{1}{1 - \theta_{1k}} \sum_{j=1}^M I(Y^j = y_k) (1 - x_{1j})$$

Lets denote  $\sum_{j=1}^M I(Y^j = y_k) x_{1j} = \#n_{1jk}$  and  $\sum_{j=1}^M I(Y^j = y_k) = \#n_{1k}$   
then:

$$\hat{\theta}_{1k} = \frac{\sum_{j=1}^M I(Y^j = y_k) x_{1j}}{\sum_{j=1}^M I(Y^j = y_k)}$$

$$\hat{\theta}_{1k} = \frac{\#n_{1jk}}{\#n_{1k}}$$

**b. (6 points)**

Now suppose each  $X_i$  is distributed normally, i.e.

$$P(X_i = x_{ij} | Y = y_k) = \frac{1}{\sigma_{ik} \sqrt{2\pi}} \exp\left(\frac{-(x_{ij} - \mu_{ik})^2}{2\sigma_{ik}^2}\right) \quad (4)$$

Suppose the variance is independent of the class variable Y, and  $X_i$ , i.e.  $\sigma_{ik} = \sigma$  Derive the MLE estimator for  $\mu_{ik}$ .

**SOLUTION:**

$$L(\mu_{ik}; \sigma) = \prod_{j=1}^M \left[ \frac{1}{\sigma \sqrt{2\pi}} \exp\left(\frac{-(x_{ij} - \mu_{ik})^2}{2\sigma^2}\right) \right]^{I(Y^j = y_k)}$$

Taking the log:

$$\ell(\mu_{ik}, \sigma) = \sum_{j=1}^M I(Y^j = y_k) \left[ \ln\left(\frac{1}{\sigma \sqrt{2\pi}}\right) + \left(\frac{-(x_{ij} - \mu_{ik})^2}{2\sigma^2}\right) \right]$$

Taking derivative with respect to  $\mu_{ik}$  and setting it to 0:

$$0 = \frac{\partial}{\partial \mu_{ik}} \sum_{j=1}^M I(Y^j = y_k) \left[ \ln\left(\frac{1}{\sigma \sqrt{2\pi}}\right) + \left(\frac{-(x_{ij} - \mu_{ik})^2}{2\sigma^2}\right) \right]$$

$$\hat{\mu}_{ik} = \frac{\sum_{j=1}^M I(Y^j = y_k) x_{ij}}{\sum_{j=1}^M I(Y^j = y_k)}$$

## 2 Regularized Multi-Class Logistic Regression, 20 points

We can easily extend the binary Logistic Regression model to handle multi-class classification. Let's assume we have K different classes, and posterior probability for class k is given by:

$$P(Y = k|X = \mathbf{x}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{1 + \sum_{t=1}^{K-1} \exp(\mathbf{w}_t^T \mathbf{x})} \text{ for } k = 1 \dots K-1$$

$$P(Y = K|X = \mathbf{x}) = \frac{1}{1 + \sum_{t=1}^{K-1} \exp(\mathbf{w}_t^T \mathbf{x})}$$

where  $\mathbf{x}$  is a n dimensional vector,  $\mathbf{w}_t^T$  is the transpose of  $\mathbf{w}_t$ . Notice that we ignored  $w_{t0}$  to simplify the expression. Our goal is to estimate the weights using gradient ascent. We will also define priors on the parameters to avoid overfitting and very large weights.

**a. (12 points)** Assume that you are given a D by N training matrix, where D is the number of training examples, and N is the number of dimensions. Please explicitly write down the log likelihood,  $L(\mathbf{w}_1, \dots, \mathbf{w}_K)$  with  $L_2$  regularization on the weights. Show your steps.

**HINT:** You can simplify the multi class logistic regression expression above by introducing a fixed parameter vector  $\mathbf{w}_K = \mathbf{0}$ .

**b. (4 points)** Note that there is not a closed form solution to maximize the log conditional likelihood,  $L(\mathbf{w}_1, \dots, \mathbf{w}_K)$ , with respect to  $\mathbf{w}_K$ . However, we can still find the solution with gradient ascent by using partial derivatives. Derive the expression for the  $i^{th}$  component in the vector gradient  $L(\mathbf{w}_1, \dots, \mathbf{w}_K)$  with respect to  $\mathbf{w}_i$ , which is the partial derivative of  $L(\mathbf{w}_1, \dots, \mathbf{w}_K)$  with respect to  $\mathbf{w}_i$ .

**c. (2 points)** Beginning with the initial weights of 0, write down the update rule for  $\mathbf{w}_K$ , using  $\nu$  for the step size.

**d. (2 points)** Will the solution converge to a global maximum?

**SOLUTION:** Let  $I_{lk}$  be an indicator function, where  $I_{lk} = 1$  if  $Y^l = k$ , otherwise  $I_{lk} = 0$ . Then we can write the likelihood as:

$$L(\mathbf{w}_1, \dots, \mathbf{w}_K) = \prod_{l=1}^D \prod_{k=1}^K P(\mathbf{Y}^l = k | \mathbf{X}^l = \mathbf{x}; \mathbf{w})^{I_{lk}}$$

$$= \prod_{l=1}^D \prod_{k=1}^K \left( \frac{\exp(\mathbf{w}_k^T \mathbf{x}^l)}{\sum_r \exp(\mathbf{w}_r^T \mathbf{x}^l)} \right)^{I_{lk}}$$

Taking log:

$$\ell(\mathbf{w}_1, \dots, \mathbf{w}_K) = \sum_{l=1}^D \sum_{k=1}^K I_{lk} \left[ \mathbf{w}_k^T \mathbf{x}^l - \ln \sum_r \exp(\mathbf{w}_r^T \mathbf{x}^l) \right]$$

Adding the  $L_2$  regularization term:

$$\ell(\mathbf{w}_1, \dots, \mathbf{w}_K) = \sum_{l=1}^D \sum_{k=1}^K I_{lk} \left[ \mathbf{w}_k^T \mathbf{x}^l - \ln \sum_r \exp(\mathbf{w}_r^T \mathbf{x}^l) \right] - \frac{\lambda}{2} \|\mathbf{w}_K\|^2$$

Taking derivative with respect to  $w_i$ :

$$\begin{aligned}\partial \frac{\ell(\mathbf{w}_1, \dots, \mathbf{w}_K)}{\partial \mathbf{w}_i} &= \sum_{l=1}^D \left[ I_{li} \mathbf{x}^l - \frac{\mathbf{x}^l \exp(\mathbf{w}_i^T \mathbf{x}^l)}{\sum_r \exp(\mathbf{w}_i^T \mathbf{x}^l)} \right] - \lambda \mathbf{w}_i \\ \partial \frac{l(\mathbf{w}_1, \dots, \mathbf{w}_K)}{\partial \mathbf{w}_i} &= \sum_{l=1}^D \left[ I_{li} - P(Y^l = i | X^l) \right] \mathbf{x}^l - \lambda \mathbf{w}_i\end{aligned}$$

Then the update rule with gradient ascent for  $w_i$  is:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \nu \sum_{l=1}^D \left[ I_{li} - P(Y^l = i | X^l) \right] \mathbf{x}^l - \nu \lambda \mathbf{w}_i$$

This will converge to a global maximum since it is a concave function

### 3 Generative-Discriminative Classifiers, 20 points

In class, we learned that when  $Y$  takes Boolean values and  $X$  is a  $n$  dimensional vector of  $X = \langle X_1, X_2 \dots X_n \rangle$  continuous variables, where each  $X_i, i = 1 \dots n$  is distributed normally (i.e.  $P(X_i | Y = y_k) = N(\mu_{ik}, \sigma_i)$ ), then Logistic Regression is the discriminative equivalent of Naive Bayes under the Naive Bayes assumptions.

**a. (14 points)** Consider instead the case where  $X = \langle X_1, X_2 \dots X_n \rangle$  is a vector of *boolean* variables. Prove that even in this case,  $P(Y|X)$  follows the same logistic function form (and hence that Logistic Regression is also the discriminative counterpart to a Naive Bayes classifier over boolean features). [Hint: see Exercise 3 in the Mitchell reading on Naive Bayes and Logistic Regression. ]

**SOLUTION:** In the lecture we derived:

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

Prior for  $P(Y = 1) = \pi$  and  $P(Y = 0) = 1 - \pi$ . Also, each  $X_i$  has binomial distribution:

$$P(X_i | Y = 0) = \theta_{i0}^{X_i} (1 - \theta_{i0})^{(1-X_i)}$$

$$P(X_i | Y = 1) = \theta_{i1}^{X_i} (1 - \theta_{i1})^{(1-X_i)}$$

Inserting this back to the equation:

$$\begin{aligned}P(Y = 1 | X) &= \frac{1}{1 + \exp \left( \ln \frac{1 - \pi}{\pi} + \sum_i \ln \frac{\theta_{i0}^{X_i} (1 - \theta_{i0})^{(1-X_i)}}{\theta_{i1}^{X_i} (1 - \theta_{i1})^{(1-X_i)}} \right)} \\ &= \frac{1}{1 + \exp \left( \ln \frac{1 - \pi}{\pi} + \sum_i X_i \ln \frac{\theta_{i0}}{\theta_{i1}} + (1 - X_i) \ln \frac{(1 - \theta_{i0})}{(1 - \theta_{i1})} \right)} \\ &= \frac{1}{1 + \exp \left( \ln \frac{1 - \pi}{\pi} + \frac{(1 - \theta_{i0})}{(1 - \theta_{i1})} + \sum_i X_i \left[ \ln \frac{\theta_{i0}}{\theta_{i1}} - \ln \frac{(1 - \theta_{i0})}{(1 - \theta_{i1})} \right] \right)}\end{aligned}$$

If we set:

$$w_0 = \ln \frac{1 - \pi}{\pi} + \sum_i \ln \frac{(1 - \theta_{i0})}{(1 - \theta_{i1})} \text{ and}$$

$$w_i = \ln \frac{\theta_{i0}}{\theta_{i1}} - \ln \frac{(1 - \theta_{i0})}{(1 - \theta_{i1})}$$

then we can reach:

$$P(Y = 1|X) = \frac{1}{1 + \exp \left( \sum_i w_i X_i \right)}$$

which is equivalent to the LR formulation.

**b. (2 points)** Suppose the data satisfies the conditional independence assumption of Naive Bayes. As the number of training examples approaches infinity, which classifier produces better results, NB or LR? Justify your answer in one sentence.

**SOLUTION:** Under conditional independence assumptions, we showed that Logistic regression is discriminative counterpart of Naive Bayes. Therefore, if the data satisfies CI assumptions, Naive Bayes and Logistic Regression will produce equivalent results.

**c. (2 points)** Suppose the data does not satisfy the conditional independence assumption of Naive Bayes. As the number of training examples approaches infinity, which classifier produces better results, NB or LR? Justify your answer in one sentence.

**SOLUTION:** Logistic Regression will produce better results, since it doesn't assume that data satisfies conditional independence.

**d. (2 points)** Is it possible to calculate  $P(X)$  from the parameters estimated by Logistic Regression? Explain.

**SOLUTION:** No it is not, LR is a discriminative classifier, that estimates  $P(Y|X)$ , not  $P(X|Y)$ . In order to calculate  $P(X)$ , we need to know  $P(X|Y)$ .

## 4 Programming, 40 points

We will now learn how to use Naive Bayes and Logistic Regression to solve a real world problem: text categorization. Text categorization (also referred as text classification) is the task of assigning documents to one or more topics. For our homework, we will use a benchmark dataset that is frequently used in text categorization problems. This dataset, Reuters-21578, consists of documents that were appeared in Reuters newswire in 1987. Each document was then manually categorized into a topic among over 100 topics. In this homework we are only interested in earn and acquisition (acq) topics, so we will be using a shortened version of the dataset (documents assigned to topics other than "earn" or "acq" are not in the dataset provided for the homework). As features, we will use the frequency (counts) of each word occurred in the document. This model is known as bag of words model and it is frequently used in text categorization.

You can download HW2.data from the class website. In this folder you will find:

**train.csv:** Training data. Each row represents a document, each column separated by commas represents

features (word counts). There are 4527 documents and 5180 words.

**train\_labels.txt:** labels for the training data

**test.csv:** Test data, 1806 documents and 5180 words

**test\_labels.txt:** labels for the test data

**word\_indices:** words corresponding to the feature indices.

For your convenience we also included a version of this dataset in .mat format, (reuters.mat) so that you can directly import it to Matlab.

Implement regularized Logistic Regression (LR) using gradient descent. Use step size  $\nu = 0.001$ , and regularization constant,  $\lambda = 0.01$ . Choose an appropriate threshold value as stopping criteria to decide if the weights are converged.

Implement Naive Bayes. To avoid 0 probabilities, choose a Beta distribution with equal valued parameters as a prior when estimating Naive Bayes parameters using MAP. You may need to implement with log probabilities to avoid underflow.

- Train your classifiers on the training set that is given. For each of the classifier, report training accuracy, testing accuracy and the amount of time spent training the classifier. For logistic regression, plot log likelihood with respect to iterations needed to converge.

### **SOLUTION:**

#### **Naive Bayes:**

Elapsed time: 0.025 seconds.

Training Accuracy: 0.97

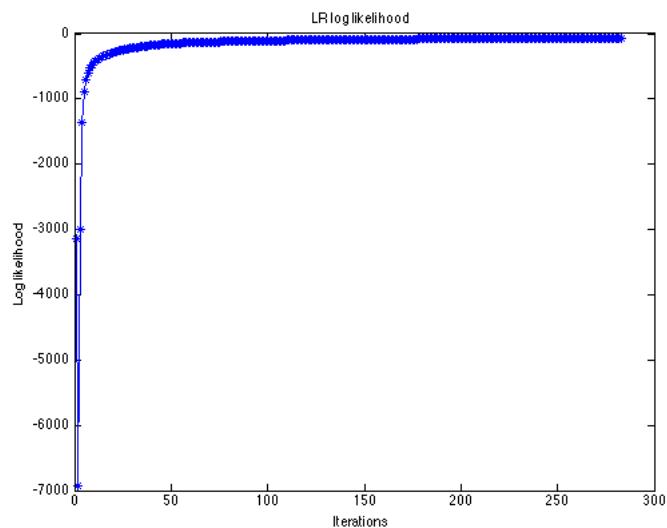
Test Accuracy: 0.98

#### **Logistic Regression:**

Elapsed time: is 407.4 seconds.

Training Accuracy: 0.996

Test Accuracy: 0.99



## 4.1 Feature selection with Mutual Information

Feature selection usually improves the classification performance on text categorization tasks. In this question, we are going to select top 1000 most informative features with mutual information (MI). Mutual information measures the contribution of a term on the correct classification decision. We can define the expected mutual information between the word  $W_i$  and the class variable Y by:

$$I(w_i; Y) = \sum_{w_i \in \{0,1\}} \sum_{y_j \in \{0,1\}} P(W_i = w_i, Y = y_j) \log_2 \frac{P(W_i = w_i, Y = y_j)}{P(W_i = w_i)P(Y = y_j)} \quad (5)$$

- b. Implement mutual information feature selection method, and reduce the dataset to include only the top 1000 most informative features. Run Naive Bayes and Logistic Regression on this new dataset and report training, testing accuracies and the amount of time spent training the classifiers. For logistic regression, plot log likelihood with respect to iterations needed to converge. Did feature selection improve classification accuracy on the test set?

### SOLUTION:

First 10 features:

'vs'  
'ct'  
'shr'  
'net'  
'qtr'  
'rev'  
'note'  
'loss'  
'mth'  
'avg'

### Naive Bayes:

Elapsed time: 0.02 seconds.

Training Accuracy: 0.96

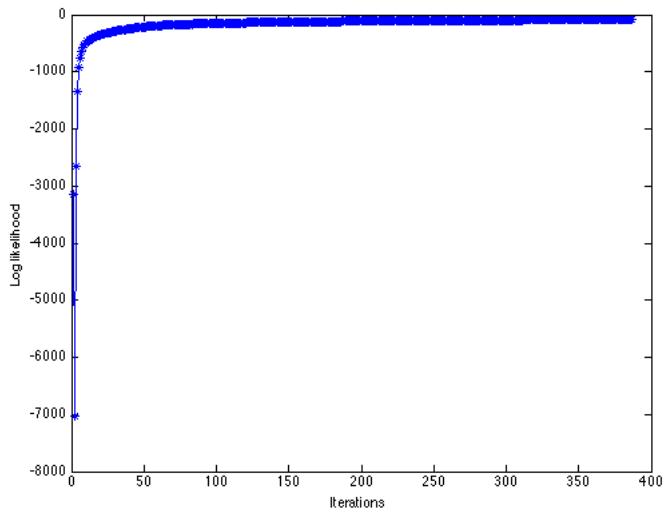
Test Accuracy: 0.98

### Logistic Regression:

Elapsed time is 70.86 seconds.

Training Accuracy: 0.995

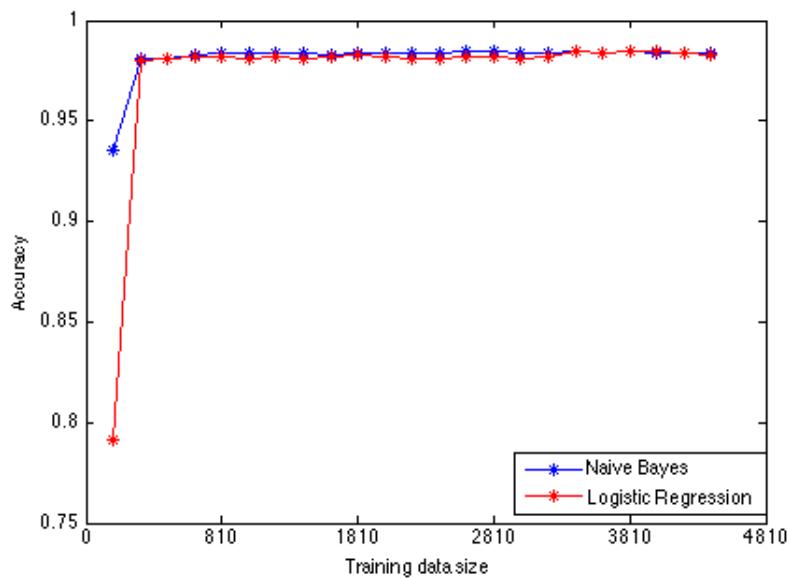
Test Accuracy: 0.985



c. We will now try to see if Naive Bayes assumptions hold for this dataset. Lets first reduce the number of features to 100, using MI as the feature selection method as described above. Then, randomly select 20 instances from the training set, and train your classifiers on the reduced training dataset. Test both classifiers on the same test dataset, and report accuracies. Then add 50 more randomly selected instances from the rest of the training set to the reduced sample and train both classifiers. Repeat this until you use all training instances for training. Include a plot of Naive Bayes and Logistic Regression accuracies on the same figure, y axis should be the classification accuracy, x axis should be the number of training samples. What do you observe? What can you tell about the Naive Bayes assumptions? Do they hold in this dataset?

**SOLUTION:**

As training data goes infinity, Naive Bayes and Logistic regression converges to the same value, so we can say that the NB assumptions hold on this particular dataset.



# 6.867 Machine Learning

## Problem Set 2 Solutions

Due date: Wednesday October 6

### Problem 1: Active Learning

1. **Solution:** The covariance matrix of the parameter vector, after selecting the first two training examples,  $x_1$  and  $x_2$ , is given by

$$\begin{aligned}\mathbf{C} &= \sigma^2(\mathbf{X}^T \mathbf{X})^{-1} = \left( \begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \right)^{-1} = \begin{bmatrix} 2 & x_1 + x_2 \\ x_1 + x_2 & x_1^2 + x_2^2 \end{bmatrix}^{-1} \\ &= \frac{1}{2(x_1^2 + x_2^2) - (x_1 + x_2)^2} \begin{bmatrix} x_1^2 + x_2^2 & -(x_1 + x_2) \\ -(x_1 + x_2) & 2 \end{bmatrix} \\ \text{Tr}(\mathbf{C}) &= \frac{1}{(x_1 - x_2)^2} (x_1^2 + x_2^2 + 2) = 1 + \frac{2(1 + x_1 x_2)}{(x_1 - x_2)^2}\end{aligned}$$

To minimise the trace, the second term should be as small as possible. Since  $x_1 x_2$  lies between 1 and  $-1$  in the input region of interest, the minimum occurs when  $x_1 x_2 = -1$ . Thus,  $x_1$  and  $x_2$  take the extreme values 1 and  $-1$ .  $\blacksquare$

2. **Solution:** After using two training examples, output variance at a test point  $x_0$  is given by

$$\begin{aligned}\text{output variance}(x_0) &= [1 \ x_0] \mathbf{C} \begin{bmatrix} 1 \\ x_0 \end{bmatrix} \\ &= \frac{1}{4} [1 \ x_0] \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ x_0 \end{bmatrix} \\ &= \frac{1}{2}(1 + x_0^2).\end{aligned}$$

See attached plot (Figure 1).  $\blacksquare$

3. **Solution:** We should either choose  $x_3 = 1$  or  $x_3 = -1$  as these are the points where output variance is maximum for  $x \in [-1, 1]$ .

After adding  $x_3 = -1$ , output variance at a test point  $x_0$  is given by

$$\begin{aligned}\text{output variance}(x_0) &= [1 \ x_0] \left( \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 \\ x_0 \end{bmatrix} \\ &= \frac{1}{8} [1 \ x_0] \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ x_0 \end{bmatrix} \\ &= \frac{1}{8}(3 + 2x_0 + 3x_0^2).\end{aligned}$$

(Note: had we picked  $x_3 = 1$  instead, we would have got  $\frac{1}{8}(3 - 2x_0 + 3x_0^2)$ .)

See attached plot (Figure 1). ■

4. **Solution:** Output variance at  $x = 0$  after adding  $x_3$  according to the sequential selection criterion is 0.375 (for both  $x_3 = 1$  and  $x_3 = -1$ ).

Output variance at  $x = 0$  after adding  $x_3 = 0$  is given by

$$\begin{aligned}\text{output variance}(x_0) &= [1 \ 0] \left( \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \frac{1}{6} [1 \ 0] \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= 0.333\end{aligned}$$

This shows that while the sequential selection criterion is good for reducing the uncertainty at all test points, it does not guarantee that the uncertainty at any specific point (other than the point queried) will be minimised. We can increase our confidence about the output at a specific point by querying that point itself instead. ■

5. **Solution:** See attached plot (Figure 2).

We make two observations from this plot:

1. For very few training examples, active learning does much better than passive learning in terms of test error on an independent data set.
2. When there are a large number of training examples, active learning does somewhat worse than passive learning.

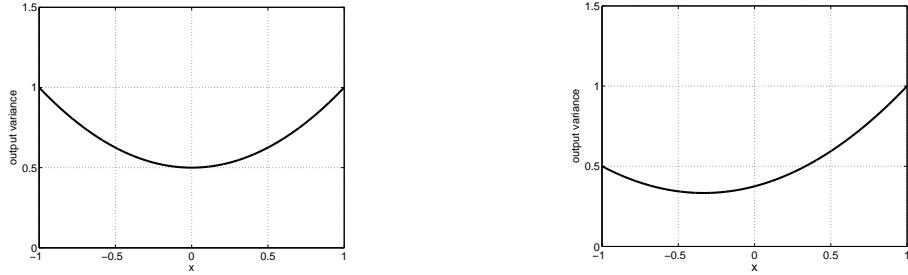


Figure 1: Plots for Problems 1.2 (left) and 1.3 (right).

The explanation for this is as follows. We have assumed a linear regression model for active learning. However, for the real data that we are working with, we have no guarantee of how good accurate a linear model is. When only a small number of examples (4 or 5) are available, the actively learned model seeks out the most informative training examples. It is thus more resistant to overfitting than the passively learned model. However, this resistance is gained at the cost of a strict assumption of linearity of the underlying model. If the data truly fit a non-linear model, the passive learning algorithm will eventually find the best linear fit. The active learning algorithm will not, as it will repeatedly query extreme training inputs in an effort to minimise variance (for the assumed linear model), and never query the large set of points lying in intermediate regions. Thus, for a large number of training examples (when both models have low output variance), the passive model is expected to have a lower ‘bias’ and hence lower generalisation error than the actively learned model.

There are secondary effects caused by the facts that querying the same input produces exactly the same output, and the noise distribution is not truly Gaussian. However, these effects are not very significant in the present context, since the test error of the actively learned linear model would still be expected to be greater than that for the passive model with infinite possible queries from a non-linear model with true Gaussian noise. (Our conclusion of the non-linearity of the model is supported by plotting the outputs against each of the input features and observing the nature of these (two) plots.) ■

## Problem 2: Optimality of the Linear Discriminant

1. **Solution:** As mentioned in the text of the problem, the points on the decision boundary satisfy:

$$\log \frac{p(\mathbf{x}|y=1)P(y=1)}{p(\mathbf{x}|y=0)P(y=0)} = 0$$

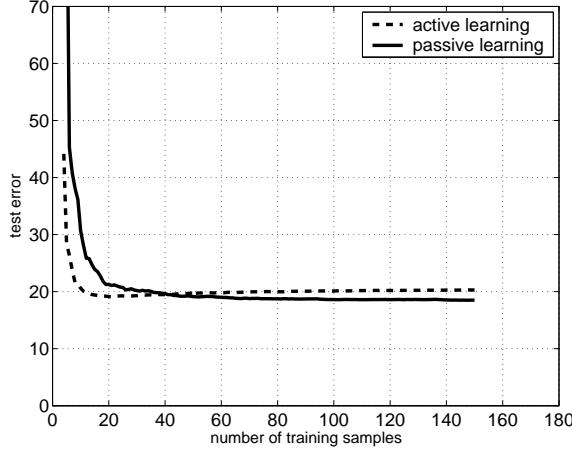


Figure 2: Plot for Problem 1.5.

The  $\sqrt{2\pi|\Sigma|}$  in the normal distributions cancel because of the ratios, while the exponentials are canceled by the logarithm:

$$\begin{aligned}
 \frac{1}{2} [(\mathbf{x} - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) - (\mathbf{x} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_0)] + \log \frac{P(y=1)}{P(y=0)} &= 0 \\
 \frac{1}{2} [2\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \mathbf{x} - 2\boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1} \mathbf{x} + \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1] + \log \frac{P(y=1)}{P(y=0)} &= 0 \\
 (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1} \mathbf{x} + \frac{1}{2} [\boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_0 - \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1] + \log \frac{P(y=1)}{P(y=0)} &= 0 \\
 (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0) + \log \frac{P(y=1)}{P(y=0)} &= 0
 \end{aligned}$$

and the assertion follows.

The decision boundary of logistic regression is also linear, and with the appropriate sample of training points, any linear separation can be the result of training logistic regression on some data. Thus the optimal decision boundary can be the decision boundary of a linear logistic regression model.

In fact, if the two Gaussians have the same covariance and we sample a large number of points as training data, the linear logistic regression decision boundary converges to the optimal decision. ■

2. **Solution:** The following optimization problem:

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \frac{(\hat{\boldsymbol{\mu}}_1^T \mathbf{w} - \hat{\boldsymbol{\mu}}_0^T \mathbf{w})^2}{n_0 \mathbf{w}^T \hat{\boldsymbol{\Sigma}}_0 \mathbf{w} + n_1 \mathbf{w}^T \hat{\boldsymbol{\Sigma}}_1 \mathbf{w}}$$

can be easily written as:

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \frac{[(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T \mathbf{w}]^2}{\mathbf{w}^T [n_0 \hat{\boldsymbol{\Sigma}}_0 + n_1 \hat{\boldsymbol{\Sigma}}_1] \mathbf{w}}$$

Therefore

$$\begin{aligned}\mathbf{m} &= \hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0 \\ \mathbf{S} &= n_0 \hat{\boldsymbol{\Sigma}}_0 + n_1 \hat{\boldsymbol{\Sigma}}_1\end{aligned}$$

$\mathbf{S}$  is symmetric and positive semi-definite because  $\hat{\boldsymbol{\Sigma}}_0$  and  $\hat{\boldsymbol{\Sigma}}_1$  are and  $n_0, n_1 \geq 0$ . ■

3. **Solution:** To write the criterion in terms of  $\mathbf{v}$ , we substitute  $\mathbf{w}$  by  $\mathbf{R}^{-1}\mathbf{v}$ :

$$\frac{(\mathbf{m}^T \mathbf{w})^2}{\mathbf{w}^T \mathbf{S} \mathbf{w}} = \frac{(\mathbf{m}^T \mathbf{R}^{-1} \mathbf{v})^2}{(\mathbf{R}^{-1} \mathbf{v})^T \mathbf{S} \mathbf{R}^{-1} \mathbf{v}} = \frac{((\mathbf{R}^{-T} \mathbf{m})^T \mathbf{v})^2}{\mathbf{v}^T \mathbf{v}} = \left[ (\mathbf{R}^{-T} \mathbf{m})^T \frac{\mathbf{v}}{\|\mathbf{v}\|} \right]^2$$

where we have used  $\mathbf{S} = \mathbf{R}^T \mathbf{R}$ .

The criterion takes the form of the square of a dot product between the fixed vector  $\hat{\mathbf{v}} = \mathbf{R}^{-T} \mathbf{m}$  and the vector of norm 1 given by  $\mathbf{v}/\|\mathbf{v}\|$ . The only degree of freedom over which to optimize is the angle between the two vectors. But if two vectors have fixed norms, their dot product is maximized when they have the same direction (the inequality  $\mathbf{v}^T \mathbf{u} \leq \|\mathbf{v}\| \cdot \|\mathbf{u}\|$  holds). Thus  $\mathbf{v} \equiv \hat{\mathbf{v}}$  maximizes the criterion (as well as any scalar multiple of  $\hat{\mathbf{v}}$ ). Moreover:

$$\hat{\mathbf{w}} = \mathbf{R}^{-1} \hat{\mathbf{v}} = \mathbf{R}^{-1} \mathbf{R}^{-T} \mathbf{m} = \mathbf{S}^{-1} \mathbf{m} = \left( n_0 \hat{\boldsymbol{\Sigma}}_0 + n_1 \hat{\boldsymbol{\Sigma}}_1 \right)^{-1} (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)$$

■

### Problem 3: Linear Discriminant vs. Logistic Regression

1. **Solution:**

	Set 1	Set 2	Set 3
logistic	0.067	0.17	0.2125
Fisher disc	0.07	0.1650	0.22

For the plots see Figure 3 at the end. ■

2. **Solution:** Yes, the performance of logistic regression and classification with the Fisher linear discriminant should be similar if data truly comes from Gaussian classes

of equal covariance. In this situation in the limit of infinite training data both logistic regression and the Fisher discriminant converge to the decision-theoretical optimal boundary, thus the only differences should arise because of the randomness of the finite training sample. ■

### 3. Solution:

	Trained on train1	Trained on train1_2
logistic	0.067	0.067
Fisher disc	0.07	0.1510

We observe that the addition of the outlier does not affect logistic regression, but has a strong negative impact on the performance of the Fisher linear discriminant classifier.

Since the new point is correctly classified and far from the boundary, the logistic model assigns to the outlier a  $P(y|\mathbf{x})$  probability exponentially close to 1. Adding this probability to the likelihood has almost not effect on the criterion, because the probability is already almost maximum at the point. Thus logistic regression is not affected.

On the other hand the Fisher discriminant sees the data as if each class is Gaussian, and the addition of a single point very far from the current mean can greatly affect the estimate of the mean and variance of that Gaussian. We can distinguish two effects on the decision boundary:

- a translation of the decision boundary because the location of the mean of one class shifts with the addition of the outlier
  - a rotation of the decision boundary because the variance in one direction increases by a large amount, while the variance in the perpendicular direction remains small. Thus the projection performed by the Fisher discriminant has to be rotated to keep the variance small.
- 

### 4. Solution:

	64 features	2D projection
logistic	0.1425	0.2125
Fisher disc	0.23	0.22

We observe that Fisher discrimination and logistic regression achieve similar classification performance on the reduced 2D representation, but while the performance of logistic regression improves significantly on the full set of features, that of Fisher discrimination remains at best the same (if not even worse than that on 2D features).

Several properties of the given representation of digits violate the Gaussian assumption. Think about averaging together all digit images in one class as if they were

printed on transparent playing cards and placed in a deck. If digits were Gaussian, the average image should be well defined (the mean of the multivariate Gaussian), and the variability around that mean image should be distributed in all directions as if its noise. In reality:

- some digits, like 4 and 7, are commonly written in more than one way, so when we look through the transparent deck of cards we see more than one defined outline
- the same basic outline can be transformed by slight rotations, shear, scaling, or translation, and while the digit remains the same, the Gaussianity is violated by such operations

Logistic regression is not that sensitive to the Gaussian assumption. If fact, in logistic regression we only model  $P(y|\mathbf{x})$  as if it is the decision boundary between Gaussians, but we do not assume that  $P(\mathbf{x})$  is a Gaussian distribution.

The Fisher discriminant is more sensitive to the Gaussian assumption because it is derived by modeling directly the means and covariances of the training data as if classes were Gaussian. If data is not Gaussian the means and covariances alone do not fully capture data structure. ■

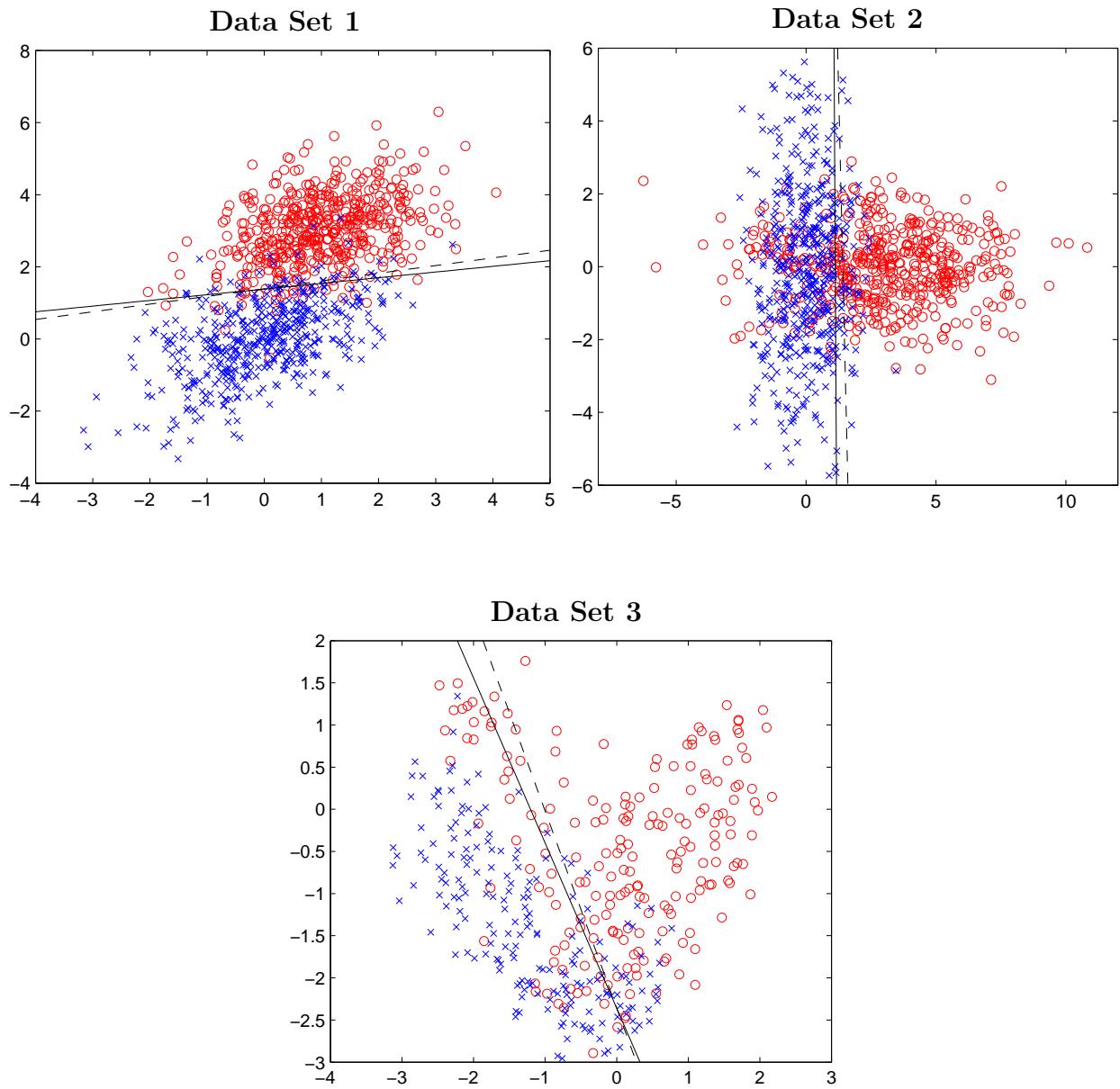


Figure 3: Plots for Problem 3 Part 1

# 10702/36702 Statistical Machine Learning, Spring 2008: Homework 3 Solutions

March 24, 2008

## 1 [25 points], (Jingrui)

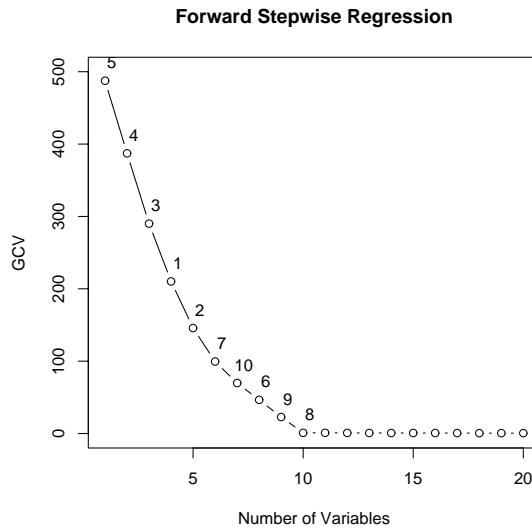
- (a) Generate data as follows.

```
 $n = 100$ 
 $p = 1000$ 
 $X = matrix(rnorm(n * p), n, p)$ 
 $beta = c(rep(10, 5), rep(5, 5), rep(0, 990))$ 
 $y = X \% * \% beta + rnorm(n)$ 
```

(1)

- (b) Write an R function to do forward stepwise regression. Apply it to your dataset and summarize the output. The output should include the cross-validation score for the sequence of models.

★ **SOLUTION:** The first 20 selected features and their corresponding GCV scores are as follows.

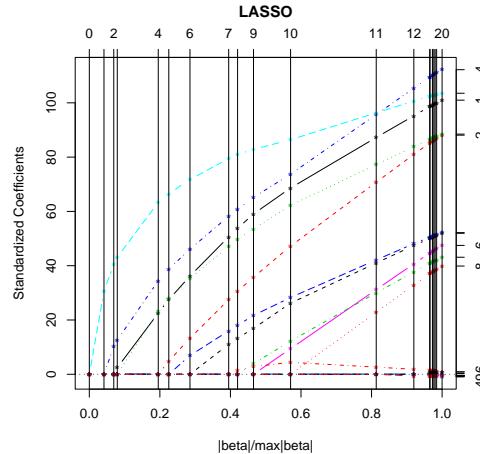
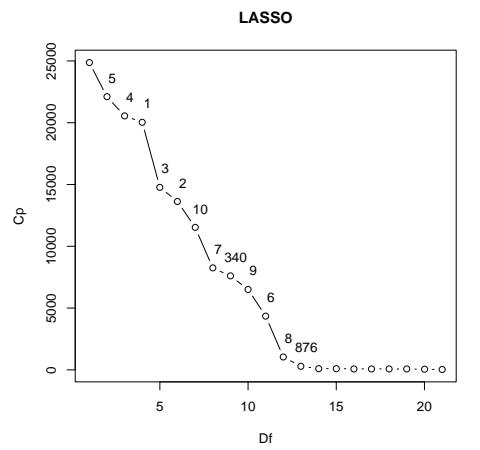


If we train a linear regression model using the 20 features, the coefficients are as follows.

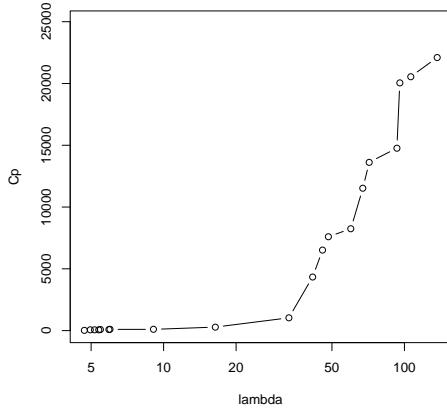
Index of Variables	5	4	3	1	2	7	10	6	9	8
Coefficient	10.22	9.84	10.12	10.21	9.97	5.11	5.12	5.19	4.93	4.79
Index of Variables	403	885	320	400	152	618	292	531	376	392
Coefficient	-0.46	-0.40	-0.23	0.29	-0.29	-0.31	-0.28	-0.22	-0.27	0.22

With forward stepwise regression, the first 10 covariates are selected in the first 10 steps, which roughly recovers the true underlying model. After the first 10 steps, the GCV score keeps decreasing, which is due to the large variance of the noise. However, the decrease in the score is not as significant as in the first 10 steps.

- (c) Run the lasso on these data and compare to the results from stepwise.

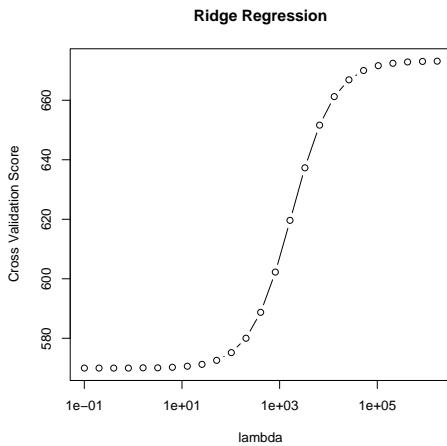


★ SOLUTION:



Based on the above figures, the first 10 covariates are among the first few features selected by Lasso, and the Cp scores stops decreasing significantly after the 10 covariates are included in the model. Compared with forward stepwise regression, with Lasso, two noise features are also included in the model.

- (d) Write a function to do ridge regression. Run your code on these data and summarize the results.



★ **SOLUTION:** Based on the above figure, we train the ridge regression model with  $\lambda = 0.1$ , and the coefficients of the first 10 covariates are as follows.

Index of Variables	1	2	3	4	5	6	7	8	9	10
Coefficient	0.94	0.62	0.88	1.09	1.37	0.39	0.84	0.32	0.51	0.70
Index of Variables	11	12	13	14	15	16	17	18	19	20
Coefficient	-0.21	0.09	-0.01	0.03	0.53	0.09	-0.17	0.10	-0.12	-0.03

Therefore, with ridge regression, the coefficients of the noise features may be very close to 0, but not exactly 0. So ridge regression can not perform feature selection.

## 2 [25 points], (Robin)

In this problem you will fit a logistic regression model to the UCI Pima Indians diabetes database. The data, and a description of the data, can be downloaded from

<http://www.ics.uci.edu/~mlearn/databases/pima-indians-diabetes>.

It is a binary classification problem with 768 instances having eight features each.

- (a) Fit a maximum likelihood logistic regression model using Newton's method (iteratively reweighted least squares). Summarize your findings.

### ★ SOLUTION:

```
# load data
data = read.table("pima-indians-diabetes.data",sep=",");
# split data into input and output
x = as.matrix(data[,1:8]);
y = as.matrix(data[,9]);

# run logistic regression
out = glm(y ~ x, family = "binomial")
summary.glm(out);

#####
# output of summary.glm(out)
#####
Call:
glm(formula = y ~ x, family = "binomial")

Deviance Residuals:
    Min      1Q      Median      3Q      Max
-2.5566 -0.7274 -0.4159  0.7267  2.9297

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -8.4046964  0.7166359 -11.728 < 2e-16 ***
xV1          0.1231823  0.0320776   3.840 0.000123 ***
xV2          0.0351637  0.0037087   9.481 < 2e-16 ***
xV3         -0.0132955  0.0052336  -2.540 0.011072 *
xV4          0.0006190  0.0068994   0.090 0.928515
xV5         -0.0011917  0.0009012  -1.322 0.186065
xV6          0.0897010  0.0150876   5.945 2.76e-09 ***
xV7          0.9451797  0.2991475   3.160 0.001580 **
xV8          0.0148690  0.0093348   1.593 0.111192
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 993.48 on 767 degrees of freedom
Residual deviance: 723.45 on 759 degrees of freedom
AIC: 741.45
```

Number of Fisher Scoring iterations: 5

The output shows that only a few features:  $x_1, x_2, x_6, x_7$  are important for classification.

- (b) Now use sparse logistic regression. Use coordinate-wise descent. Plot the estimates as a function of  $\lambda$ . Also, plot the AIC score as a function of  $\lambda$ .

★ SOLUTION:

```

# sparse logistic regression
# assumed that the first column of x is all 1's
sparse <- function(x,y,lambda){
  # Initialization
  n <- length(y);
  numFeats <- dim(x)[2];
  beta <- 1/numFeats/colMeans(x);
  epsilon <- 1;

  while ( epsilon > 1e-6 ){
    betaOld <- beta;
    for ( j in 1:numFeats ){
      betaTmp <- beta;
      betaTmp[j] <- 0;
      pTilda <- as.vector(1 - 1/(1+exp(x%*%betaTmp)));

      if ( abs( t(y-pTilda)%*%x[,j] ) < lambda ){
        beta[j] = 0;
      }
      else {
        delta <- 1;
        while ( delta > 1e-3 ){
          p <- as.vector(1 - 1/(1+exp(x%*%beta)));
          delta <- (t(y-p)%*%x[,j]-lambda*sign(beta[j]))/(t(p*(1-p))%*%(x[,j]^2));
          beta[j] <- beta[j]+delta;
        }
      }
    } # j loop terminates

    epsilon <- sum(abs(beta-betaOld));
  } # while (epsilon < 1e-6) loop end

  ## compute log-likelihood
  ll <- y %*% (x%*%beta) - sum(log(1+exp(x%*%beta)));
  S <- sum(beta!=0);

  AIC1 <- ll - S;
  AIC2 <- -2*ll + 2*S;

  return (list(beta=beta,AIC1=AIC1,AIC2=AIC2));
} # sparse function end

# main program
# load data
data = read.table("pima-indians-diabetes.data",sep=",");

```

```

# split data into input and output
x = as.matrix(data[,1:8]);
y = as.vector(data[,9]);

# Add column for intercept beta_0
x = cbind(rep(1,length(y)),x);

size <- 500;
maxLambda <- 250;
lambda <- seq(0,maxLambda,length=size);
beta <- matrix(rep(0,size*dim(x)[2]),nrow=dim(x)[2]);
aic <- matrix(rep(0,size*2),nrow=2);

for ( i in 1:size){
  out <- sparse(x,y,lambda[i]);
  beta[,i] <- out$beta;
  aic[1,i] <- out$AIC1;
  aic[2,i] <- out$AIC2;
  cat(i,"\\n");
}

# plotting results
par(mfrow=c(1,3));

# plotting the estimates
plot(lambda,beta[1,],type='l',col=10,ylab="Beta")
for ( i in 2:9 )
  lines(lambda,beta[i,],type='l',col=10*i);

#plotting AIC
plot(lambda,aic[1,],type='l',ylab="AIC=ll-|S|");
plot(lambda,aic[2,],type='l',ylab="AIC=-2*ll+2*|S|");

```

The figure shows that  $\beta$  values reduce to zeroes for higher values of  $\lambda$ . You can either maximize  $AIC = ll - |S|$  or minimize  $AIC = -2(ll + |S|)$ . Either case the figure shows the optimal value of  $\lambda$  is 0.

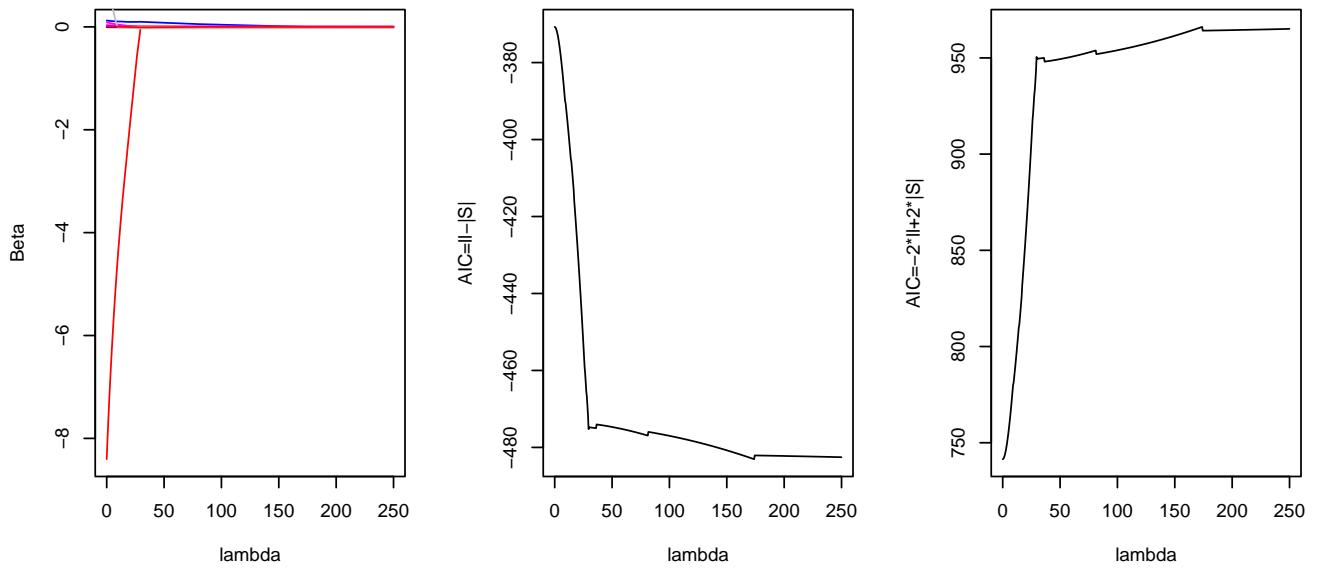
### 3 [25 points], (Jingrui)

Let  $x = (x_1, \dots, x_d)^T \in \mathbb{R}^d$  and let

$$g(x) = \|x\|_p = \left( \sum_{j=1}^d |x_j|^p \right)^{1/p}$$

where  $0 < p < 1$ . Find the subgradient of  $g$ .

★ **SOLUTION:** Subgradient/subderivative is only defined for convex functions. As we saw in the last homework, when  $0 < p < 1$ ,  $g(x)$  is not convex, so it does not have subgradient. When  $p = 1$ ,  $g(x) = \sum_{j=1}^d |x_j|$  is a convex function. Its subdifferential at  $x_j = 0$  is  $[-1, 1]$ ; its subdifferential at  $x_j < 0$  is  $-1$ ; and its subdifferential at  $x_j > 0$  is  $1$ .



## 4 [25 points], (Robin)

- (a) Write an R program to fit a mixture of multivariate Normals using the EM algorithm (with the number of components  $k$  known).

### ★ SOLUTION:

```
# mixture of multi-variate norms using EM algo.
mvg <- function(X, mu, Sigma, i, j)
{
  d <- dim(X)[2];
  p <- 1/((2*pi)^(d/2))/(det(Sigma)^(1/2))*exp(-0.5*t(X-mu) %*% solve(Sigma) %*% (X-mu));
  return(p);
}

EMmmn <- function(X, k){
  n <- dim(X)[1];
  d <- dim(X)[2];

  m <- rep(0, d);
  for ( j in 1:d )
    m[j] <- mean(X[, j]);

  # initialize parameters
  w <- matrix(rep(0, n*k), nrow = n);
  mu <- matrix(runif(d*k), nrow=k);
  p <- matrix(rep(1/k, k), nrow=k);

  Sinit <- 1/n*t(X-m) %*% (X-m);
```

```

S <- matrix(rep(0,d*d*k),nrow=k);

for ( j in 1:k ) S[j,] <- as.vector(Sinit);

tol <- 1;
# repeat until convergence
while ( tol > 1e-03*k*d )
{
  ll <- 0;
  oldmu <- mu;
  # compute weights
  for ( j in 1:k )
  {
    Sj <- matrix(S[j,],nrow=d);
    for ( i in 1:n )
    {
      prob <- mvg(as.matrix(X[i,]),as.matrix(mu[j,]),Sj);
      w[i,j] <- p[j]*prob;
      ll <- ll + log(prob);
    }
  }

  # normalize weights
  for ( i in 1:n )
  {
    sumWi <- sum(w[i,]);
    w[i,] <- w[i,]/sumWi;
  }

  # update parameters
  for ( j in 1:k )
  {
    sumWj <- sum(w[,j]);
    p[j] <- 1/n*sumWj;

    for ( i in 1:d )
      mu[j,i] <- sum(X[,i]*w[,j])/sumWj;

    S[j,] <- as.vector( (t(X-mu[j,])%*%((X-mu[j,])*w[,j]))/sumWj );
  }

  tol <- sum(abs(mu-oldmu));
  % cat("tol: ",tol,"\n");
}

# compute log-likelihood
ll <- 0;
for ( j in 1:k )
{
  Sj <- matrix(S[j,],nrow=d);
  for ( i in 1:n )
    ll <- ll + w[i,j]*log(mvg(as.matrix(X[i,]),as.matrix(mu[j,]),Sj));
}

```

```

s <- k+d*k+k*d*d;
aic <- ll - s;
bic <- ll - s/2*log(n);
return(list(prob=p,mu=mu,var=S,aic=aic,bic=bic));
}

```

(b) Simulate 1000 observations from the model

$$\frac{1}{5}N(0, I) + \frac{4}{5}N(\mu, I)$$

where  $\mu = (3, 3, 3, 3, 3)$  and  $I$  is the 5 by 5 identity matrix. Use your program to find the MLE.

★ SOLUTION:

```

# load library
library(MASS);

d <- 5; # number of dimensions
n <- 1000; # number of points
S <- diag(d); # identity matrix

# generate data
X <- matrix(rep(0,n*d),nrow=n);

for ( i in 1:n ) {
  if ( runif(1,0,1) < 0.2 ) {
    X[i,] <- mvrnorm(1,mu=rep(0,d),Sigma=S);
  }
  else { X[i,] <- mvrnorm(1,mu=rep(3,d),Sigma=S); }
}

res <- EMmmn(X,2);

cat("cluster probs\n");
res$prob
cat("cluster means\n");
res$mu[1,]
res$mu[2,]
cat("cluster vars\n");
matrix(res$var[1,],nrow=d)
matrix(res$var[2,],nrow=d)

#####
# output of code
#####
cluster probs
> res$prob
[,1]
[1,] 0.8061191
[2,] 0.1938809
> cat("cluster means\n");
cluster means
> res$mu[1,]

```

```

[1] 2.970932 3.049995 3.012737 2.900860 2.986491
> res$mu[2,]
[1] 0.10066126 0.02896716 0.04202755 0.04309818 0.06456465
> cat("cluster vars\n");
cluster vars
> matrix(res$var[1,],nrow=d)
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.94865362 -0.010098981 0.048339147 -0.036357260 0.093404195
[2,] -0.01009898 1.026635129 -0.005396046 -0.054604952 -0.015011003
[3,] 0.04833915 -0.005396046 0.978004876 0.002056654 0.007193553
[4,] -0.03635726 -0.054604952 0.002056654 1.034921146 -0.035570629
[5,] 0.09340420 -0.015011003 0.007193553 -0.035570629 1.007609017
> matrix(res$var[2,],nrow=d)
      [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.85527994 0.05907938 0.022014719 -0.040955944 -0.01855187
[2,] 0.05907938 1.08175250 -0.062179874 0.101668376 0.01957809
[3,] 0.02201472 -0.06217987 1.061560303 -0.004966484 0.04057524
[4,] -0.04095594 0.10166838 -0.004966484 0.799421239 0.05533264
[5,] -0.01855187 0.01957809 0.040575241 0.055332636 1.01671623

```

(c) Now repeat but take  $k$  as unknown. Use AIC and BIC to choose  $k$ .

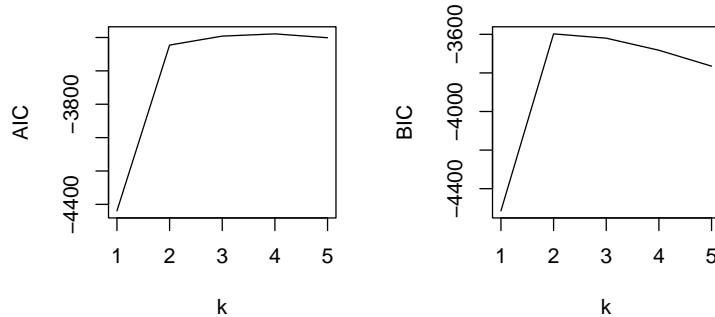
```

AIC = rep(0,5);
BIC = rep(0,5);

# Run EM for multiple k
for ( k in 1:5 )
{
  res <- EMmmn(X,k);
  AIC[k] <- res$aic;
  BIC[k] <- res$bic;
  cat(k,"\\n");
}

#plotting results
par(mfrow=c(1,2));
plot(AIC,xlab='k',ylab='AIC',type='l');
plot(BIC,xlab='k',ylab='BIC',type='l');

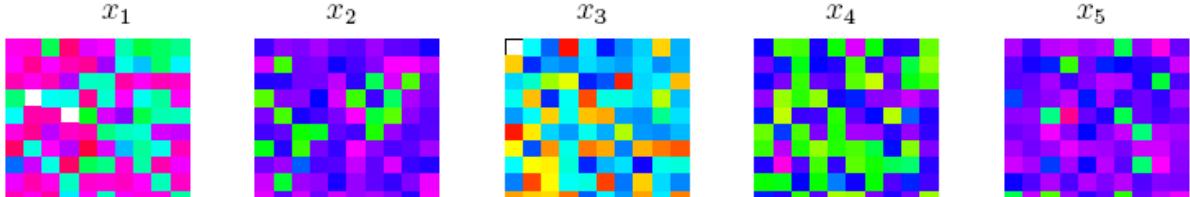
```



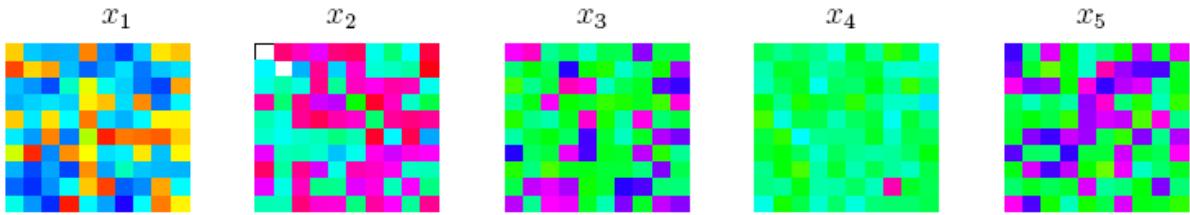
Using the graphs of AIC and BIC we notice that optimal  $k$  using AIC and BIC are 4 and 2 respectively. Hence BIC is a better criteria to estimate the number of mixtures.

## 5 [25 points], (Jingrui)

In this problem you will use mixture models for a classification task. The training data consist of 100 examples  $x_i, y_i$ , where  $x_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5})$  and each  $x_{ij} \in \mathbb{R}^{100}$  is a 100-dimensional vector, and  $y_i = \{c_{ij}\}$  is an unordered set of up to five labels, with each  $c_{ij} \in \{1, 2, \dots, 10\}$ . Two examples are shown below (check them out in color in the electronic version):



$$y = \{1, 5, 6, 9\}$$



$$y = \{1, 7, 9\}$$

In the first example, there are four classes  $y = \{1, 5, 6, 9\}$ . Each of the five  $x_i$  was generated from exactly one of these four classes; thus, one of the four classes must have generated two of the  $x_i$ . Considering the alignment of each of the five images with one of the classes as a latent variable  $z = (z_1, z_2, z_3, z_4, z_5)$ , a possible alignment is  $z = (9, 5, 9, 1, 6)$  where class 9 generates  $x_1$ , class 5 generates  $x_2$ , class 9 generates  $x_3$ , and so on. The test data consist of 100 unlabeled examples  $x_i$ . The task is to label each of the test examples with an *ordered* set of labels  $y_i = (c_{i1}, c_{i2}, c_{i3}, c_{i4}, c_{i5})$  where  $c_{ij}$  is the predicted class of  $x_{ij}$ .

You may use any method and model you choose, but it should make use of mixture models in some way. Give a detailed explanation of your approach and methodology. The training data and test data are on the course website.

Describe your method clearly and report a summary of how well you do on the test data.

**★ SOLUTION:** Let the hidden variables be  $z_{ij}^k$ .  $z_{ij}^k = 1$  iff the  $j^{\text{th}}$  image of the  $i^{\text{th}}$  example is from the  $k^{\text{th}}$  component, and  $z_{ij}^k = 0$  otherwise. The parameters in the mixture model include: the priors of the components  $p_1, \dots, p_{10}$ , the mean vector of the components  $\mu_1, \dots, \mu_{10}$ , and the covariance matrices of the components  $\Sigma_1, \dots, \Sigma_{10}$ . For the sake of simplicity, assume that the features are independent given the component. In this case, the covariance matrices are diagonal matrices, and we only need to estimate the diagonal element  $\sigma_{kd}^2$ ,  $k \in \{1, \dots, 10\}, d \in \{1, \dots, 100\}$ , which is the variance of  $j^{\text{th}}$  feature in the  $i^{\text{th}}$  component.

The EM algorithm works as follows.

1. E-step: if  $k > 0$  and  $k \in y_i$ , then  $w_{ij}^k = \mathbb{P}(z_{ij}^k = 1 | x_1, \dots, x_n) = \frac{p_k \phi(x_{ij}^k; \mu_k, \Sigma_k)}{\sum_{l \in y_i, l > 0} p_l \phi(x_{ij}^l; \mu_l, \Sigma_l)}$ ; otherwise,  $w_{ij}^k = 0$ .

2. M-step:

$$p_k = \frac{1}{5n} \sum_{i=1}^n \sum_{j=1}^5 w_{ij}^k$$

$$\mu_k = \frac{\sum_{i=1}^n \sum_{j=1}^5 x_{ij} w_{ij}^k}{\sum_{i=1}^n \sum_{j=1}^5 w_{ij}^k}$$

$$\sigma_{kd}^2 = \frac{\sum_{i=1}^n \sum_{j=1}^5 (x_{ij}^d - \mu_k^d)^2 w_{ij}^k}{\sum_{i=1}^n \sum_{j=1}^5 w_{ij}^k}$$

where  $\mu_k^d$  is the  $d^{\text{th}}$  component of  $\mu_k$ , and  $x_{ij}^d$  is the  $d^{\text{th}}$  component of  $x_{ij}$ .

The test error of the above model is 35.6%.

# 10-601 Machine Learning, Fall 2012

## Homework 3

Instructors: Tom Mitchell, Ziv Bar-Joseph

TA in charge: Mehdi Samadi  
email: msamadi@cs.cmu.edu

Due: Monday October 15, 2012 by 4pm

**Instructions** There are 4 questions on this assignment – no programming. Please hand in a hard copy of your completed homework to Sharon Cavlovich (GHC 8215) by 4 PM on Monday, October 15th, 2012. Don’t forget to include your name and email address on your homework.

## 1 Neural Networks

### 1.1 Expressiveness of Neural Networks [10 points]

As discussed in class, neural networks are built out of units with real-valued inputs  $X_1 \dots X_n$ , where the unit output  $Y$  is given by

$$Y = \frac{1}{1 + \exp(-(w_0 + \sum_i w_i X_i))}$$

Here we will explore the expressiveness of neural nets, by examining their ability to represent boolean functions. Here the inputs  $X_i$  will be 0 or 1. Of course the output  $Y$  will be real-valued, ranging anywhere between 0 and 1. We will interpret  $Y$  as a boolean value by interpreting it to be a boolean 1 if  $Y > 0.5$ , and interpreting it to be 0 otherwise.

1. Give 3 weights for a single unit with two inputs  $X_1$  and  $X_2$ , that implements the logical OR function  $Y = X_1 \vee X_2$ .

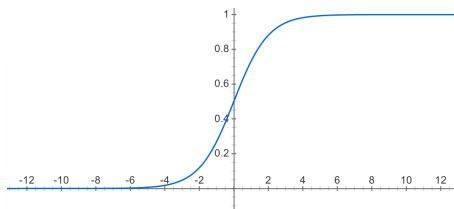


Figure 1:  $\frac{1}{1+e^{-x}}$ .

★ **SOLUTION:** Figure 1 shows the value of  $y = \frac{1}{1+e^{-x}}$  for different values of  $x$ . Note that  $y \geq 0.5$  if  $x \geq 0$ , and  $y \leq 0.5$  if  $x \leq 0$ . Given this, we need to choose  $w_i$  so that  $w_0 + w_1 * x_1 + w_2 * x_2$  will be greater than 0 when  $x_1 \vee x_2$  is equal to 1. One candidate solution is  $[w_0 = -0.5, w_1 = 1, w_2 = 1]$ .

2. Can you implement the logical AND function  $Y = X_1 \wedge X_2$  in a single unit? If so, give weights that achieve this. If not, explain the problem.

★ **SOLUTION:** Similar to previous part, we can obtain  $[w_0 = -1.5, w_1 = 1, w_2 = 1]$

3. It is impossible to implement the EXCLUSIVE-OR function  $Y = X_1 \oplus X_2$  in a single unit. However, you can do it using a multiple unit neural network. Please do. Use the smallest number of units you can. Draw your network, and show all weights of each unit.

★ **SOLUTION:** It can be represented by a neural network with two nodes in the hidden layer. Input weights for node 1 in the hidden layer would be  $[w_0 = -0.5, w_1 = 1, w_2 = -1]$ , input weights for node 2 in the hidden layer would be  $[w_0 = -0.5, w_1 = -1, w_2 = 1]$ , and input weights for the output node would be  $[w_0 = -0.8, w_1 = 1, w_2 = 1]$ .

4. Create a neural network with only one hidden layer (of any number of units) that implements  $(A \vee \neg B) \oplus (\neg C \vee \neg D)$ . Draw your network, and show all weights of each unit.

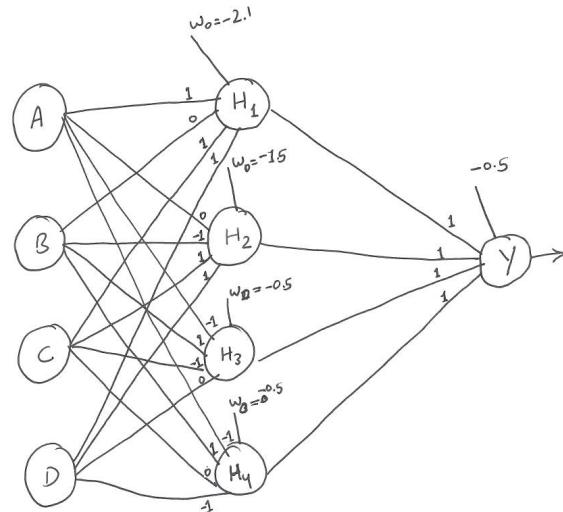


Figure 2: An example of neural network for problem 1.4

★ **SOLUTION:** Note that XOR operation can be written in terms of AND and OR operations:  $p \oplus q = (p \wedge \neg q) \vee (\neg p \wedge q)$ . Given this, we can rewrite the formula as  $(A \wedge C \wedge D) \vee (\neg B \wedge C \wedge D) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg D)$ . This formula can be represented by a neural network with one hidden layer and four nodes in the hidden layer (one unit for each parenthesis). An example is shown in Figure 2.

## 1.2 MCLE, MAP, Gradient descent [15 points]

In class we showed the derivation of the gradient descent rule to train a *single* logistic (sigmoid) unit to obtain a Maximum Conditional Likelihood Estimate for the unit weights  $w_0 \dots w_n$ . (See the slides from the lecture on neural networks: [http://www.cs.cmu.edu/~tom/10601\\_fall2012/slides/NNets-9\\_27\\_2012.pdf](http://www.cs.cmu.edu/~tom/10601_fall2012/slides/NNets-9_27_2012.pdf), especially the slides on pages 4 and 5).

1. The slide at the top of page 5 claims that if we want to place a Gaussian prior on the weights, to obtain a MAP estimate instead of a Maximum likelihood estimate, then we must choose weights that minimize the expression  $E$ :

$$E = c \sum_i w_i^2 + \sum_l (y^l - \hat{f}(x^l))^2$$

where  $w_i$  is the  $i^{th}$  weight for our logistic unit,  $y^l$  is the target output for the  $l^{th}$  training example,  $x^l$  is the vector of inputs for the  $l^{th}$  training example,  $\hat{f}(x^l)$  is the unit output for input  $x^l$ , and  $c$  is some constant.

Show that this claim is correct, by showing that minimizing  $E$  is equivalent to maximizing the expression:  $\ln P(W) \prod_l P(Y^l | X^l; W)$ . Here  $W$  is the weight vector  $\langle w_0 \dots w_n \rangle$ . In particular, assume each weight  $w_i$  in the single unit follows a zero-mean Gaussian prior, of the form:

$$p(w_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \left(\frac{w_i - 0}{\sigma}\right)^2\right)$$

So that  $P(W) = P(w_0, \dots, w_n) = \prod_{i=0}^n P(w_i)$ .

### ★ SOLUTION:

$$\begin{aligned} \hat{W}_{MAP} &= \arg \max_W \ln \left( P(W) \prod_l P(Y^l | X^l; W) \right) \\ &= \arg \max_W \ln P(W) + \ln \prod_l P(Y^l | X^l; W) \\ &= \arg \max_W \ln P(W) + \ln \prod_l \left( \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \left(\frac{Y^l - f(X^l)}{\sigma}\right)^2\right) \right) \end{aligned} \quad (1)$$

From the definition of  $P(W)$  we can write:

$$\begin{aligned} P(W) &= \prod_{i=0}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \left(\frac{w_i - 0}{\sigma}\right)^2\right) \\ \ln P(W) &= \frac{n \ln(2\pi\sigma^2)}{2} + \left( \frac{-\sum_{i=0}^n w_i^2}{2\sigma^2} \right) \end{aligned} \quad (2)$$

By removing constant values from the above two formulas (since they don't change the maximization) and removing the negative sign, we would have:

$$\begin{aligned} \hat{W}_{MAP} &= \arg \min_W c \sum_{i=0}^n w_i^2 + \sum_{l=1}^M (Y^l - f(X^l))^2 \\ &= \arg \min_W E \end{aligned} \quad (3)$$

2. Derive the gradient you would use to obtain the map estimate, for a single unit with two inputs  $X_1$  and  $X_2$ . In other words, give formulas for each of the three partial derivatives

$$\left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2} \right]$$

*Hint: the slide at the bottom of page 5 of the handout does part of your task. If you get stuck, the slides on linear regression might also be helpful.*

★ SOLUTION:

$$\begin{aligned}
 E &= c \sum_{i=0}^n w_i^2 + \sum_{l=1}^M (Y^l - f(X^l))^2 \\
 &= cw_0^2 + cw_1^2 + cw_2^2 + \sum_l (Y^l - f(X^l))^2 \\
 \frac{\partial E}{\partial w_0} &= 2cw_0 - \sum_l 2(Y^l - f(X^l)) \frac{\partial}{\partial w_0}(f(X^l))
 \end{aligned} \tag{4}$$

In class we showed that  $\frac{\partial}{\partial w_0}(f(X^l)) = f(X^l) \cdot (1 - f(X^l))$ . So we can write:

$$\frac{\partial E}{\partial w_0} = 2cw_0 - \sum_l 2(Y^l - f(X^l))f(X^l)(1 - f(X^l)) \tag{5}$$

Similarly, we could derive formulas for  $w_1$  and  $w_2$ .

## 2 Bayesian Networks [20 points]

### 2.1 Representation and Inference [12 points]

Consider the following Bayes net:

$$A \rightarrow B \rightarrow C \leftarrow D$$

1. Write the joint probability  $P(A, B, C, D)$  for this network as the product of four conditional probabilities.

★ SOLUTION:

$$P(A, B, C, D) = P(A)P(B|A)P(D)P(C|B, D) \tag{6}$$

2. How many independent parameters are needed to fully define this Bayesian Network?

★ SOLUTION: We need 8 independent variables.

3. How many independent parameters would we need to define the joint distribution  $P(A, B, C, D)$  if we made *no* assumptions about independence or conditional independence?

★ SOLUTION:  $2^4 - 1$

4. [6 pts] Consider the even simpler 3-node Bayes Net

$$A \rightarrow B \rightarrow C$$

Give an expression for  $P(B = 1|C = 0)$  in terms of the *parameters of this network*. Use notation like  $P(C = 1|B = 0)$  to represent individual Bayes net parameters.

★ **SOLUTION:** Using Bayes' Rule, we have:

$$\begin{aligned} P(B = 1|C = 0) &= \frac{P(C = 0|B = 1)P(B = 1)}{P(C = 0)} \\ &= \frac{(1 - P(C = 1|B = 1))P(B = 1)}{P(C = 0)} \end{aligned} \tag{7}$$

Equations for  $P(B = 1)$  and  $P(C = 0)$  can be derived by:

$$\begin{aligned} P(B = 1) &= P(B = 1|A = 0)P(A = 0) + P(B = 1|A = 1)P(A = 1) \\ P(C = 0) &= 1 - P(C = 1) \\ &= 1 - (P(C = 1|B = 0)P(B = 0) + P(C = 1|B = 1)P(B = 1)) \end{aligned} \tag{8}$$

Note that 5 independent parameters for this Bayesian Network are:  $P(A = 1)$ ,  $P(B = 1|A = 0)$ ,  $P(B = 1|A = 1)$ ,  $P(C = 1|B = 0)$ ,  $P(C = 1|B = 1)$ . The above equations can be written in terms of these parameters using complement rule in probability.

## 2.2 Learning Bayes Nets [8 points]

Suppose you want to learn a Bayes net over two binary variables  $X_1$  and  $X_2$ . You have  $N$  training pairs of  $X_1$  and  $X_2$ , given as  $\{(x_1^1, x_2^1), (x_1^2, x_2^2), (x_1^3, x_2^3), \dots, (x_1^N, x_2^N)\}$ . Given two datasets  $A$  and  $B$ , we know that the data in  $B$  is generated by  $x_2^j = F(x_1^j, \theta) + \epsilon$  for all training instances  $j$  where  $\theta$  and  $\epsilon$  are two unknown parameters. We don't have any information on how dataset  $A$  is generated. Let  $BN$  denote the Bayes Net with no edges, and  $BN'$  denote the BN with an edge from  $X_1$  to  $X_2$ . For both of these Bayes net, we learn its parameters using maximum likelihood estimation.

1. Which Bayes net is better to model the dataset  $A$ ? Explain your answer.

★ **SOLUTION:**  $BN'$ . We shouldn't make any independence assumption between variables since we don't know how data has been generated.

2. Which Bayes net is better to model the dataset  $B$ ? Explain your answer.

★ **SOLUTION:**  $BN'$  since we know that  $X_1$  and  $X_2$  are not independent.

## 3 Expectation Maximization (EM) [15 points]

Consider again the simple Bayes Network from question 2:  $A \rightarrow B \rightarrow C$ . You must train this network from partly observed data, using EM and the following training examples:

example 1: A=1, B=1, C=0  
example 2: A=1, B=?, C=0  
example 3: A=0, B=0, C=1  
example 4: A=0, B=1, C=1

Assume that we begin with *each independent parameter of this network initialized to 0.6* (recall that you enumerated these in question 2).

1. As we execute the EM algorithm, what gets calculated during the first E step?

★ SOLUTION: For each training example  $k$ :

$$E[B_k] = P(B_k = 1 | A_k, C_k, \theta) = \frac{P(B_k = 1, A_k, C_k | \theta)}{P(B_k = 1, A_k, C_k | \theta) + P(B_k = 0, A_k, C_k | \theta)} \quad (9)$$

2. Give the value for this quantity, as calculated by the first E step.

★ SOLUTION: For  $k = 2$ :

$$\begin{aligned} E[B_2] &= \frac{\theta_{C=0|B=1}\theta_{B=1|A=1}\theta_{A=1}}{\theta_{C=0|B=1}\theta_{B=1|A=1} + \theta_{C=0|B=0}\theta_{B=0|A=1}\theta_{A=1}} \\ &= \frac{0.4 * 0.6 * 0.6}{0.4 * 0.6 * 0.6 + 0.4 * 0.4 * 0.6} = 0.6 \end{aligned} \quad (10)$$

3. What gets calculated during the first M step?

★ SOLUTION:

$$\begin{aligned} \theta_{A=1} &= \frac{\sum_{k=1}^N \delta(A_k = 1)}{N} \\ \theta_{B=1|A=0} &= \frac{\sum_{k=1}^N \delta(A_k = 0)E[B_k]}{\sum_{k=1}^N \delta(A_k = 0)} \\ \theta_{B=1|A=1} &= \frac{\sum_{k=1}^N \delta(A_k = 1)E[B_k]}{\sum_{k=1}^N \delta(A_k = 1)} \\ \theta_{C=1|B=1} &= \frac{\sum_{k=1}^N \delta(C_k = 1)E[B_k]}{\sum_{k=1}^N E[B_k]} \\ \theta_{C=1|B=0} &= \frac{\sum_{k=1}^N \delta(C_k = 1)(1 - E[B_k])}{\sum_{k=1}^N (1 - E[B_k])} \end{aligned} \quad (11)$$

4. Give the value for this set of quantities, as calculated by the first M step.

★ SOLUTION:

$$\begin{aligned} \theta_{A=1} &= 0.5 \\ \theta_{B=1|A=0} &= 0.5 \\ \theta_{B=1|A=1} &= 0.8 \\ \theta_{C=1|B=0} &= 0.71 \\ \theta_{C=1|B=1} &= 0.38 \end{aligned} \quad (12)$$

## 4 Midterm Review Questions [15 points]

Here are short questions (some from previous midterm exams) intended to help you review for our midterm on October 18.

### 4.1 True or False Questions [9 points]

If true, give a 1-2 sentence explanation. If false, a counterexample.

1. As the number of training examples grows toward infinity, the MLE and MAP estimates for Naive Bayes parameters converge to the same value in the limit.

★ **SOLUTION:** False, since we have not made any assumption about the prior. A simple counterexample is the prior which assigns probability 1 to a single choice of parameter  $\theta$ .

2. As the number of training examples grows toward infinity, the probability that logistic regression will overfit the training data goes to zero.

★ **SOLUTION:** True.

3. In decision tree learning with noise-free data, starting with the wrong attribute at the root can make it impossible to find a tree that fits the data exactly.

★ **SOLUTION:** False.

### 4.2 Short Questions [6 points]

1. The Naive Bayes algorithm selects the class  $c$  for an example  $x$  that maximizes  $P(c|x)$ . When is this equivalent to selecting the  $c$  that maximizes  $P(x|c)$ ?

★ **SOLUTION:**  $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$ , so finding the  $c$  that maximizes  $P(c|x)$  is equivalent to finding the  $c$  that maximizes  $P(x|c)$ , if the prior  $P(c)$  is uniform.

2. Imagine you have a learning problem with an instance space of points on the plane. Assume that the target function takes the form of a line on the plane where all points on one side of the line are positive and all those on the other are negative. If you are asked to choose between using a decision tree or a neural network with no hidden layer, which would you choose? Why?

★ **SOLUTION:** Neural network. A decision tree is not able to learn some of the linear functions on a plane (e.g.,  $y = x$ ).

# 6.867 Machine Learning

## Problem Set 3 Solutions

Due date: Wednesday October 20

### Problem 1: Kernels, features and maximum margin

1. **Solution:** For any  $\alpha$ ,

$$\begin{aligned}\alpha^T \mathbf{K} \alpha &= \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ &= \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j) \\ &= (\sum_{i=1}^n \alpha_i \Phi(\mathbf{x}_i))^T (\sum_{j=1}^n \Phi(\mathbf{x}_j)) \\ &= (\sum_{i=1}^n \alpha_i \Phi(\mathbf{x}_i))^2 \\ &\geq 0\end{aligned}$$

Hence,  $\mathbf{K}$  is a positive semi-definite matrix. ■

2. **Solution:**

- (a) For a pair of input points  $\mathbf{x}$  and  $\mathbf{x}'$ , and two possible feature vectors  $\Phi^{(1)}(\cdot)$  and  $\Phi^{(2)}(\cdot)$  for each point, of lengths  $n_1$  and  $n_2$  respectively, the product kernel value

(which is a scalar) is given by:

$$K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') \times K_2(\mathbf{x}, \mathbf{x}') \quad (1)$$

$$= (\Phi^{(1)}(\mathbf{x})^T \Phi^{(1)}(\mathbf{x}')) \times (\Phi^{(2)}(\mathbf{x})^T \Phi^{(2)}(\mathbf{x}')) \quad (2)$$

$$= \left( \sum_{i=1}^{n_1} \Phi_i^{(1)}(\mathbf{x}) \Phi_i^{(1)}(\mathbf{x}') \right) \times \left( \sum_{j=1}^{n_2} \Phi_j^{(2)}(\mathbf{x}) \Phi_j^{(2)}(\mathbf{x}') \right) \quad (3)$$

$$= \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \Phi_i^{(1)}(\mathbf{x}) \Phi_j^{(2)}(\mathbf{x}) \Phi_i^{(1)}(\mathbf{x}') \Phi_j^{(2)}(\mathbf{x}') \quad (4)$$

$$= \sum_{k=1}^{n_1 \times n_2} \Phi_k(\mathbf{x}) \Phi_k(\mathbf{x}') \quad (5)$$

$$= \Phi(\mathbf{x})^T \Phi(\mathbf{x}'), \quad (6)$$

which is a valid inner product. Thus, the product kernel,  $K(\mathbf{x}, \mathbf{x}')$ , is a valid kernel.

(b) The rules we apply at different steps are as follows:

Step 1: Scaling. (Using  $f(\mathbf{x}) = (1/||\mathbf{x}||)$ , we get a new kernel  $K_3(\mathbf{x}, \mathbf{x}')$ )

Step 2: Sum. ( $K_4(\mathbf{x}, \mathbf{x}') = 1 + K_3(\mathbf{x}, \mathbf{x}')$ )

Step 3: Product, twice. (to obtain  $K_4(\mathbf{x}, \mathbf{x}')^3$ )

■

3. **Solution:** Input vectors  $x_1 = 0$  and  $x_2 = 1$  are mapped to feature vectors  $\Phi(x_1) = [1 \ 0 \ 0]^T$  and  $\Phi(x_2) = [1 \ 2 \ 2]^T$ .

(a) For a training set consisting of one positive and one negative example, the direction of the weight vector  $\hat{\mathbf{w}}_1$  is the same as the direction of a vector from the negative example to the positive example, in the feature space. Thus, direction of  $\hat{\mathbf{w}}_1 = [0 \ 2 \ 2]^T$

(b) Margin = distance from each support vector to decision boundary (in feature space) = distance from each training point to a point midway between the 2 points (since there are only 2 training examples) =  $\sqrt{2}$ .

(c)  $||\hat{\mathbf{w}}_1|| = 1/(\text{margin})$ . We have now specified both the norm and the direction of the weight vector. Therefore, we can calculate the weight vector as  $\hat{\mathbf{w}}_1 = [0 \ 1/2 \ 1/2]^T$ .

Constraints are satisfied as equalities at the support vectors (*i.e.*, the two training points). Thus,

$$\begin{aligned} -1(w_0 + [1 \ 0 \ 0] \hat{\mathbf{w}}_1) &= 1 \\ +1(w_0 + [1 \ 2 \ 2] \hat{\mathbf{w}}_1) &= 1 \end{aligned}$$

Solving,  $w_0 = -1$ .

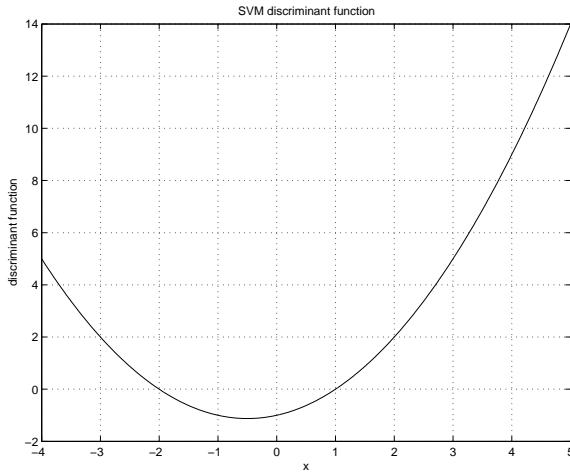


Figure 1: Plot for Problem 1.3 (d)

- (d) The discriminant function, as a function of input  $x$ , is given by  $w_0 + \hat{\mathbf{w}}_1^T \Phi(x) = x^2/2 + x/\sqrt{2} - 1$ . See Figure 1.

For large negative inputs  $x$ , the SVM classifier output is positive.

The decision boundary is equidistant from the support vectors (*i.e.* the two training points) in feature space, not in input space. There is a non-linear mapping from input space to feature space. Thus, points equidistant from the training points in the input space need not lie on the decision boundary.

■

## Problem 2: Support Vector Machines

### 1. Solution:

**Set 1** The best kernel is the linear one. Rationale: samples seem to be drawn from two Gaussians of the same covariance, whose decision boundary is linear. It is always best to choose the simplest model to discourage overfitting. You can also choose by the error rate on the test set.

**Set 2** The best kernel is the second order polynomial one. Rationale: the decision boundary between the two classes seems to be a parabola, a curve of degree 2. It is certainly not linear, and the Gaussian kernel overfits. You can also choose by the error rate on the test set.

**Set 3** The best kernel is the Gaussian kernel. Rationale: the clusters are clearly not separable with curves of degree one or two. Since the Gaussian kernel always

separates points, it is the best choice here. Also, points are similar under the Gaussian kernel if they are close to each other in the original space. It is clear the small distance to the cluster center is the defining property of the classes. You can also choose by the error rate on the test set.

Plots are shown in Figure 2. ■

2. Solution:	linear	2 <sup>nd</sup> order	Gaussian
	0.1375	0.12	0.085

All the three kernels perform better than logistic regression (which was 0.1425). ■

### Problem 3: Non-Separable SVM's

1. **Solution:** Because  $L$  is linear in  $w_0$ ,  $\xi$ , and  $\rho$ , and  $w_0$ ,  $\xi$ , and  $\rho$  are unconstrained, the coefficients in front of these variables must be 0. Otherwise we could drive  $L$  to  $-\infty$  by choosing appropriate large values for  $w_0$ ,  $\xi$ , or  $\rho$ . Thus  $\alpha$ ,  $\beta$ ,  $\gamma$  must satisfy the following conditions in order for  $J$  to be finite:

$$\sum_{i=1}^n \alpha_i y_i = 0 \quad (7)$$

$$\alpha_i + \beta_i = \frac{1}{n} \quad \text{for every } i \in \{1, \dots, n\} \quad (8)$$

$$\sum_{i=1}^n \alpha_i = \gamma + \nu \quad (9)$$

When these conditions are satisfied,  $L$  becomes:

$$L(w_0, \mathbf{w}_1, \xi, \rho; \alpha, \beta, \gamma) = \frac{1}{2} \|\mathbf{w}_1\|^2 - \left( \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i^T \right) \mathbf{w}_1$$

To obtain  $J$  under these conditions, we must minimize the above formula for  $L$  with respect to  $\mathbf{w}_1$  (the only variable left). Either by taking the derivative, or simply by seeing it as a quadratic function, the minimizing  $\mathbf{w}_1$  is  $\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ . At this  $\mathbf{w}_1$ ,  $J$  is:

$$J(\alpha, \beta, \gamma) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

For completeness,  $J(\alpha, \beta, \gamma) = -\infty$  if any of the constraints (7), (8), (9) is not satisfied. It follows that the maximum of  $J(\alpha, \beta, \gamma)$  with respect to  $\alpha, \beta, \gamma$  will not be achieved if any of (7), (8), (9) is not satisfied.

Since  $\beta$  does not appear in the criterion, the constraints  $\alpha_i + \beta_i = \frac{1}{n}$ ,  $\beta_i \geq 0$  can be rewritten as  $\frac{1}{n} - \alpha_i \geq 0$ . Similarly  $\sum_{i=1}^n \alpha_i = \gamma + \nu$ ,  $\gamma \geq 0$  becomes  $\sum_{i=1}^n \alpha_i - \nu \geq 0$ . ■

**2. Solution:** For this part we assume  $\alpha$  is known as a result of solving the dual optimization, and we must express  $w_0$  and  $\rho$  in terms of  $V\alpha$  and the training data.  $\mathbf{w}_1$  is already known from the derivation of the dual optimization:  $\mathbf{w}_1 = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ . Let  $i$  be a positive support vector ( $y_i = 1$ ) and  $j$  be a negative one ( $y_j = -1$ ). Then:

$$\begin{aligned} w_0 + \mathbf{x}_i^T \mathbf{w}_1 &= \rho \\ -w_0 - \mathbf{x}_j^T \mathbf{w}_1 &= \rho \end{aligned}$$

Adding the two equations together we get

$$\rho = \frac{1}{2} (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{w}_1$$

To get a robust estimate  $\hat{\rho}$ , we can consider all positive-negative pairs of SV's, and average the values we obtain from the above equation. An alternative is to take the median.

Now given  $\hat{\rho}$ , we can solve for  $w_0$  from and SV constraint:

$$w_0 = y_i \hat{\rho} - \mathbf{x}_i^T \mathbf{w}_1$$

To get a robust estimate, we can average all the above equations for all SV's, or take the median.

An alternative solution is to view the problem as solving an over-constrained system of linear equations with 2 unknowns. A robust solutions to such a system that minimizes the error between predicted and real values in the least square sense is given by the *pseudo*-inverse of a (non necessarily square) matrix. ■

**3. Solution:** The code for solving the  $\nu$ -SVM is shown in Figure 3.

$\nu$	0.01	0.1	0.3	0.5	0.7
training error	0	0.01	0.05	0.07	0.18
test error	0.013	0.014	0.073	0.105	0.195

$\nu$  has the expected effect in the sense the number of training errors roughly increases with  $\nu$ . The fraction of training errors is not exactly  $\nu$  because of the limited sample on which the SVM is trained (in the limit they should become equal). ■

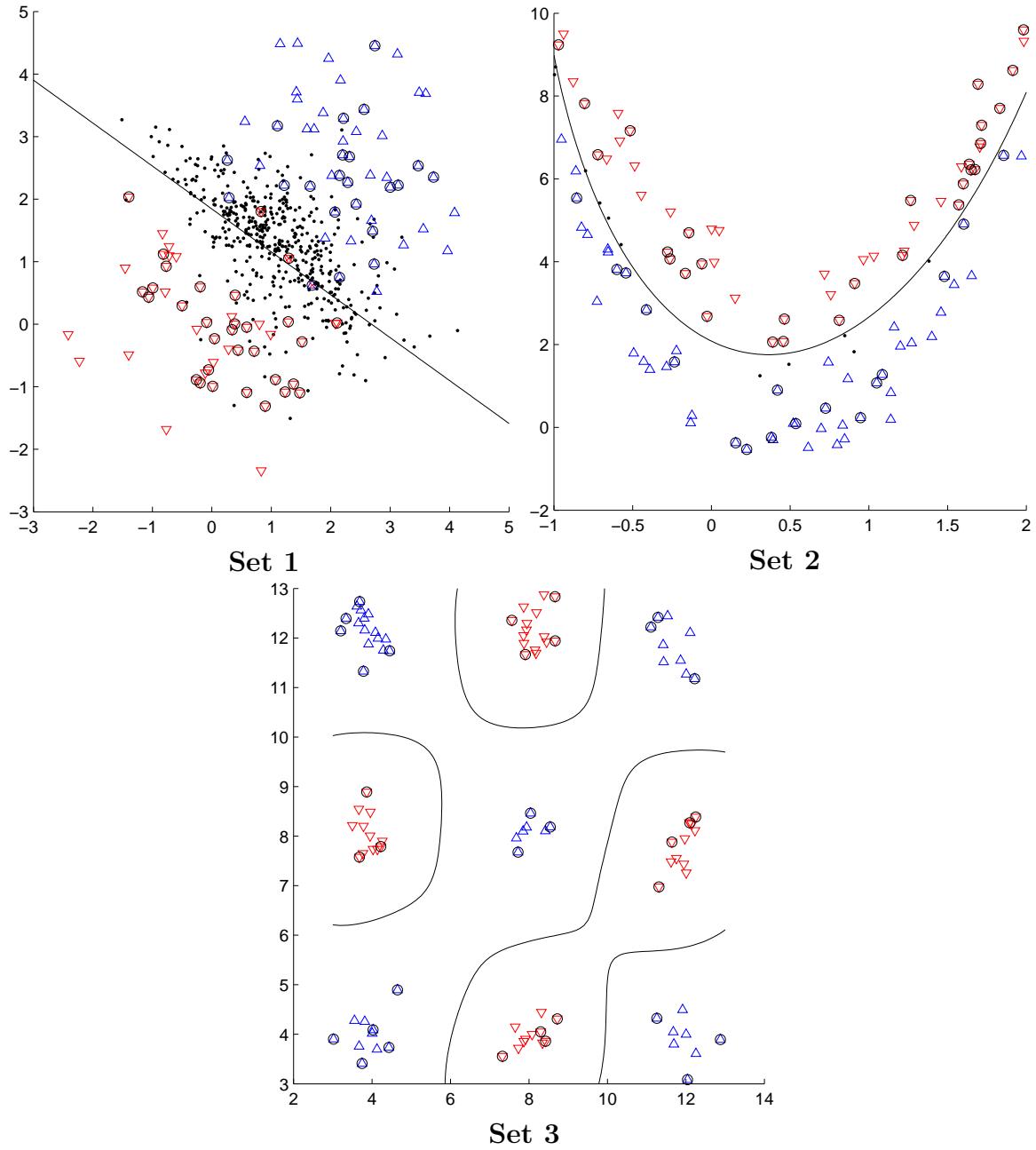


Figure 2: Plots for problem 3 part 1

```

function svm = nusvm_train(data, kernel, param, nu)

y = data.y;
X = data.X;
n = length(y);

% evaluate the kernel matrix
K = feval(kernel,X,X,param); % n x n positive semi-definite matrix
K = (K+K')/2; % should be symmetric. if not, may replace by equiv symm kernel.

%%%%%%%%%%%%%
% For Part 4 of the problem, you must fill in the following section.
% Make sure you understand the parameters to 'quadprog' (doc quadprog)
%%%%%%%%%%%%%
D = diag(y); % diagonal matrix with D(i,i) = y(i)
H = D*K*D; % H(i,j) = y(i)*K(i,j)*y(j)
f = zeros(n,1);
A = -ones(1,n);
b = -nu;
Aeq = y';
beq = 0.0;
LB = zeros(n,1);
UB = 1/n * ones(n,1);
%%%%%%%%%%%%%
X0 = zeros(n,1);

warning off; % suppress 'Warning: Large-scale method ...'
alpha = quadprog(H+1e-10*eye(n),f,A,b,Aeq,beq,LB,UB,X0)
warning on;

% essentially, we have added a (weak) regularization term to
% the dual problem favoring minimum-norm alpha when solution
% is underdetermined. this is also important numerically
% as any round-off error in computation of H could potentially
% cause dual problem to become ill-posed (minimizer at infinity).
% regularization term forces Hessian to be positive definite.
% select support vectors.
S = find(alpha > eps);
NS = length(S);
beta = alpha(S).*y(S);
XS = X(S,:);
% estimate w0 robustly (bias parameter)
dpos = find((y > 0) & (alpha > 0) & (alpha < 1/n));
dneg = find((y < 0) & (alpha > 0) & (alpha < 1/n));
margvecs = [dpos ; dneg];
npos = length(dpos);
nneg = length(dneg);
Mpos = reshape(repmat(reshape(K(S,dpos), [NS npos 1]), [1 1 nneg]), [NS npos * nneg]);
Mneg = reshape(repmat(reshape(K(S,dneg), [NS 1 nneg]), [1 npos 1]), [NS npos * nneg]);
rho = mean(0.5*beta'*(Mpos - Mneg));
w0 = median(rho*y(margvecs) - sum(diag(beta)*K(S,margvecs))');
% store the results
svm.kernel = kernel;
svm.NS = NS;
svm.w0 = w0;
svm.beta = beta;
svm.XS = XS;
svm.rho = rho;
svm.param = param;

```

Figure 3: Solution to problem 3 part 3

# 10702/36702 Statistical Machine Learning, Spring 2008: Homework 4 Solutions

May 8, 2008

## 1 [20 points], (Jingrui)

Let  $X = (X_1, \dots, X_d)^T$  where each  $X_j \in \{0, 1\}$ . Consider the loglinear model

$$\log p(x) = \beta_0 + \sum_{j=1}^d \beta_j X_j + \sum_{j < k} \beta_{jk} X_j X_k + \dots + \sum_{j < k < l} \beta_{jkl} X_j X_k X_l + \dots +$$

Suppose that  $\beta_A = 0$  whenever  $\{1, 2\} \subset A$ . Show that

$$X_1 \perp\!\!\!\perp X_2 | X_3, \dots, X_d.$$

★ SOLUTION: Since  $\beta_A = 0$  whenever  $\{1, 2\} \subset A$ , the loglinear model can be written as follows.

$$\log p(x) = \sum_{1 \in A, 2 \notin A} \beta_A \prod_{j \in A} X_j + \sum_{2 \in A, 1 \notin A} \beta_A \prod_{j \in A} X_j + \sum_{1 \notin A, 2 \notin A} \beta_A \prod_{j \in A} X_j$$

Therefore,

$$p(x) = f_1(x_1, x_3, \dots, x_d) f_2(x_2, x_3, \dots, x_d)$$

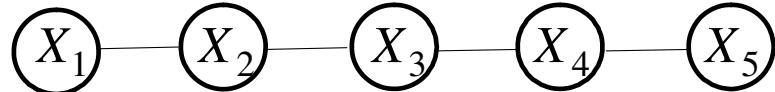
and

$$p(x_1, x_2 | x_3, \dots, x_d) = f'_1(x_1, x_3, \dots, x_d) f'_2(x_2, x_3, \dots, x_d)$$

So  $X_1 \perp\!\!\!\perp X_2 | X_3, \dots, X_d$ .

## 2 [20 points], (Jingrui)

Consider the graph: Assume all variables are binary. One loglinear model that is consistent with this graph



is

$$\log p(x) = \beta_0 + 5(X_1 X_2 + X_2 X_3 + X_3 X_4 + X_4 X_5)$$

Simulate  $n = 100$  random vectors from this distribution. Fit the model

$$\log p(x) = \beta_0 + \sum_j \beta_j X_j + \sum_{j < k} \beta_{jk} X_j X_k$$

using maximum likelihood. Report your estimators. Use forward/backward model selection with BIC to choose a submodel. Compare the selected model to the true model.

Intercept	$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_1 \times X_2$	$X_1 \times X_3$
-24.04	-17.19	-17.19	-17.19	-17.19	-17.19	11.46	11.46
$X_1 \times X_4$	$X_1 \times X_5$	$X_2 \times X_3$	$X_2 \times X_4$	$X_2 \times X_5$	$X_3 \times X_4$	$X_3 \times X_5$	$X_4 \times X_5$
11.46	11.46	11.46	11.46	11.46	11.46	11.46	11.46

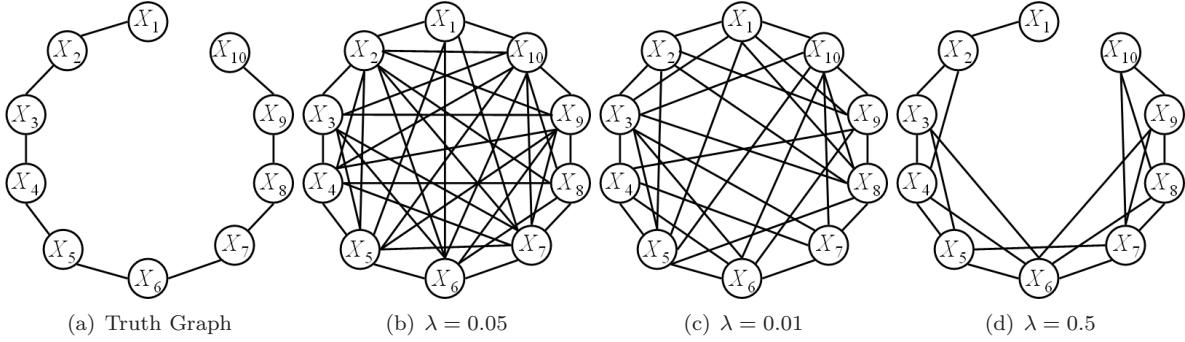
★ **SOLUTION:** Fitting the full model using maximum likelihood, we get the following coefficients:

Using backward model selection with BIC, the selected submodel is  $\log p(x) = -126.63 + 26.25X_1 + 26.25X_2 + 26.25X_3 + 26.25X_4 + 26.25X_5$ . Based on the true model, we can see that the probability of  $X_i = 1$ ,  $i = 1, 2, 3, 4, 5$  is 0.9864352. And the generated vectors are all 1s, which explains why both the full model and the selected model are far from the true model. This problem can be alleviated by sampling more vectors from the true model.

### 3 [20 points], (Jingrui)

Let  $X \sim N(0, \Sigma)$  where  $X = (X_1, \dots, X_p)^T$ ,  $p = 10$ . Let  $\Theta = \Sigma^{-1}$  and suppose that  $\Theta(i, i) = 1$ ,  $\Theta(i, i-1) = .5$ ,  $\Theta(i-1, i) = .5$  and  $\Theta(i, j) = 0$  otherwise. Simulate 50 random vectors and use the glasso to estimate the covariance matrix. Compare your estimated graph to the true graph.

★ **SOLUTION:** The true graph and the estimated graphs with different values of  $\lambda$  are shown as follows.



From these figures, we can see that the larger  $\lambda$  is, the more penalization on the L1 norm of the edges, and the fewer edges we have in the estimated graphs.

### 4 [20 points], (Robin)

Generate  $n = 100$  observations from the model:

$$Y = m(X) + \epsilon$$

where  $\epsilon \sim N(0, \sigma^2)$ ,  $\sigma = 0.1$ ,

$$X \sim \text{Uniform}([0, 1]^{10})$$

and

$$m(x) = \cos(5\pi x_1) + 5x_2^2.$$

Note that  $x$  is 10-dimensional but  $m(x)$  only depends on  $x_1$  and  $x_2$ . Estimate  $m$  using (i) multivariate kernel regression, (ii) additive model, (iii) regression tree. For each estimator, report

$$\frac{1}{n} \sum_{i=1}^n (\hat{m}(X_i) - m(X_i))^2.$$

★ SOLUTION: This code is provided by Maxim Makatchev

```
#generate data
n=100
p=10
x = matrix(runif(p*n), nrow=n, ncol=p)
mx = cos(5*pi*x[,1]) + 5*x[,2]^2
e=rnorm(n = n, mean=0, sd=0.1)
y=mx+e

xtest = matrix(runif(p*n), nrow=n, ncol=p)
mxtest = cos(5*pi*x[,1]) + 5*x[,2]^2
e=rnorm(n = n, mean=0, sd=0.1)
ytest=mxtest+e

#training bias
H = seq(0.1, 3, length=50)
out = Kernreg(y, x, H, x)
#training bias
sum((out\$f-mx)^2/n)
#[1] 0.2056952
out$h
#[1] 0.3367347

#test bias
ftest = kernreg(y, x, out$h, xtest)
#test bias
sum((ftest-mxtest)^2/n)
#[1] 3.850887

##### 4b additive model
training.data = data.frame(x=x, y=y)
names(training.data) = c(paste("x", 1:10, sep=""), "y")
test.data = data.frame(x=xtest, y=ytest)
names(test.data) = c(paste("x", 1:10, sep=""), "y")

library(gam)

out = gam(y ~ lo(x1) + lo(x2) + lo(x3) + lo(x4) + lo(x5) +
           lo(x6) + lo(x7) + lo(x8) + lo(x9) + lo(x10), data=training.data)
#out1=step(gam(y ~ lo(x1) + lo(x2) + lo(x3) + lo(x4) + lo(x5) +
#               lo(x6) + lo(x7) + lo(x8) + lo(x9) + lo(x10), data=training.data),
#               data = training.data, k = log(n))
print(summary(out))

#returned model
#(Intercept) 1.0
#lo(x1)      1.0      2.4 12.3288 9.211e-06 ***
#lo(x2)      1.0      2.3 20.7006 3.446e-08 ***
#lo(x3)      1.0      2.3  0.9850 0.3873147
#lo(x4)      1.0      2.3  8.3140 0.0003579 ***
#lo(x5)      1.0      2.3  0.6713 0.5350998
```

```

#lo(x6)      1.0      2.2  0.4991  0.6280062
#lo(x7)      1.0      2.7  1.6557  0.1899930
#lo(x8)      1.0      2.4  1.0052  0.3839098
#lo(x9)      1.0      2.4  0.6429  0.5593328
#lo(x10)     1.0      2.6  0.5483  0.6242972

y_pred_train = predict(out)
##train bias
sum((y_pred_train - mx)^2/n)
#[1] 0.2096737
y_pred_test = predict(out,newdata=test.data)
#test bias
sum((y_pred_test - mxtest)^2/n)
#[1] 4.113105

##### 4c regression tree
library(tree)
out = tree(y ~ x,data=training.data)
print(summary(out))
plot(out,type="u",lwd=3)
text(out)

#training bias
sum((out$y - mx)^2)/n
#[1] 0.009276874

postscript('prob4_cv.ps')
#Regression tree:
out=tree(formula = y ~ x, data = training.data)

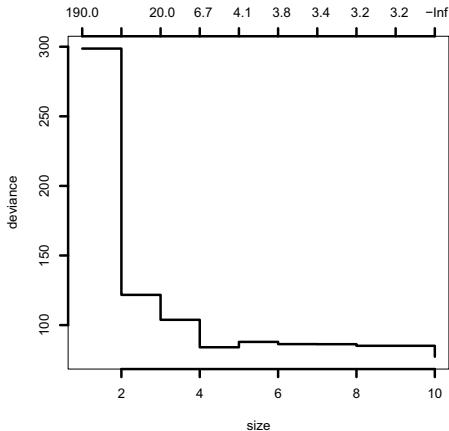
#Variables actually used in tree construction:
#[1] "x.2" "x.4"
#Number of terminal nodes: 6
#Residual mean deviance: 0.3916 = 36.81 / 94

cv=cv.tree(out)
plot(cv,lwd=3)
dev.off()

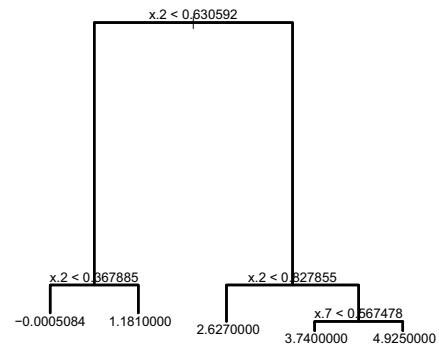
postscript('prob4_pruned.ps')
newtree=prune.tree(out, best=5)
print(summary(newtree))
#Number of terminal nodes: 5
#Residual mean deviance: 0.4219 = 40.08 / 95
plot(newtree,lwd=3)
text(newtree,cex=1)
dev.off()

y_pred_train=predict.tree(newtree)
##train bias
sum((y_pred_train - mx)^2/n)
#[1] 0.3941942

```



(e) CV score



(f) Pruned Tree

## 5 [20 points], (Robin)

You're goal is to compare several classifiers, namely: (i) logistic regression, (ii) additive model, (iii) k-nearest neighbors and (iv) classification trees. The data are the "iris data" which is a famous dataset. These data are already in R. Type:

```
data(iris)
names(iris)
print(iris)
pairs(iris)
boxplot(iris)
```

There are three species and the goal is to predict species from the four features. We will use the first 100 observations only so there are only two species. Construct the classifiers and compare the training error rates. For the additive model, plot the estimated functions.

★ SOLUTION: This code is provided by Yu-Hsiang Chi

```
Logistic Regression: Training Error: 0
Additive Regression: Training Error: 0
K-NN Training Error: 0
Classification Tree Training Error: 0
```

```
hat = function(p){
  n = length(p)
  y = rep(1,n)
  y[p < .5] = 0
  return(y)
}

# data from Iris dataset
n <- 100
x <- matrix(rep(0,n*4),n,4)
```

```

for(i in 1:4){
  x[,i] <- iris[1:n,i]
}

y <- matrix(rep(0,n),n,1)
for(i in 1:n){
  if(iris[i,5]=="setosa"){
    y[i,1] <- 1
  }
}
training.data <- data.frame(x=x,y=y)
names(training.data) = c(paste("x",1:4,sep=""), "y")

### logistic regression
out_lr = glm(y ~ .,family="binomial",data=training.data)
p = predict(out_lr,type="response")
yhat_lr = hat(p)
error_lr = mean(y != yhat_lr)
cat("Logistic Regression: Training Error: ",error_lr,"\\n")

### additive model
library(gam)

out_add = gam(y ~ lo(x1) + lo(x2) + lo(x3) + lo(x4), data = training.data)
print(summary(out_add))

yhat_add = hat(predict(out_add))
error_add = mean(y != yhat_add)
cat("Additive Regression: Training Error: ",error_add,"\\n")

postscript("prob5_fig1.ps");
par(mfrow=c(2,2))
plot(out_add,lwd=3,xlab="",ylab="")
dev.off();

### knn
library(class)
kmax = 50
error2 = rep(0,kmax)
index = rep(0,kmax)
for(i in 1:kmax){
  y_cv = knn.cv(x, y, k = (2*i-1))
  error2[i] = mean(y_cv != y)
  index[i] = 2*i-1
}

postscript("prob5_fig2.ps");
par(mfrow=c(1,1))
plot(index,error2,type="l",lwd=3,xlab="k",ylab="K-NN Training Error")
dev.off();

### Classification tree
library(tree)

```

```

out_tree = tree(as.factor(y) ~ x,data=training.data)
print(summary(out_tree))

postscript("prob5_fig3.ps");
par(mfrow=c(2,2))
plot(out_tree,type="u",lwd=3)
text(out_tree)

cv = cv.tree(out_tree,method="misclass")
plot(cv,lwd=3)

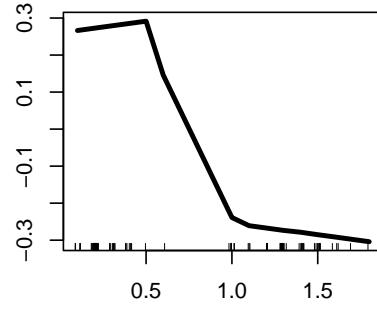
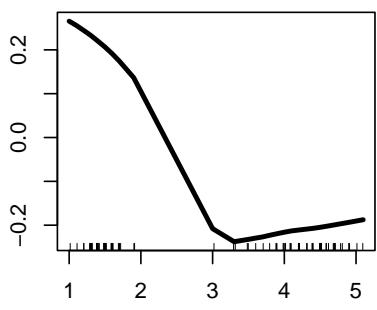
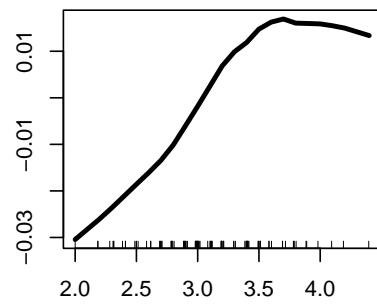
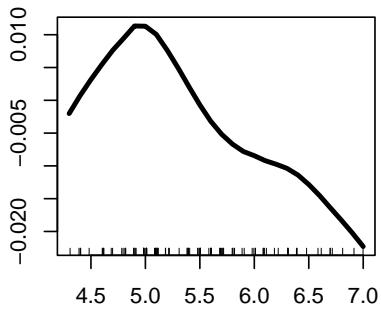
newtree = prune.tree(out_tree,best=cv$size[which.min(cv$dev)],method="misclass")

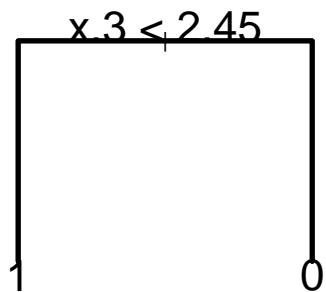
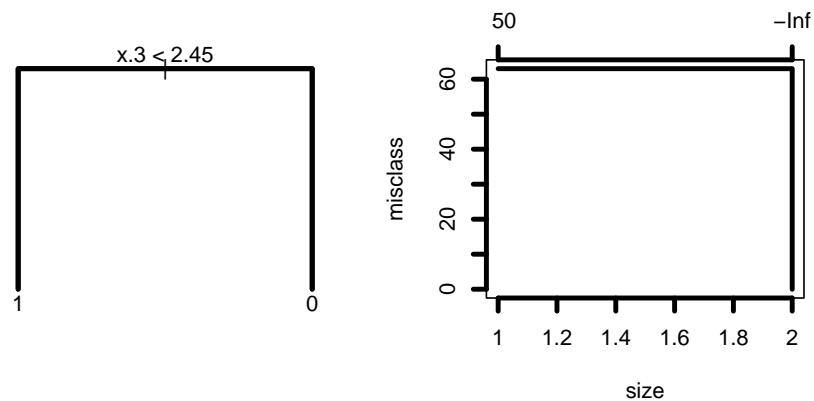
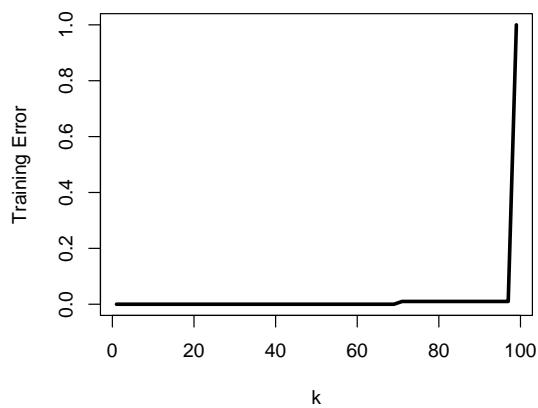
print(summary(newtree))

# Misclassification error rate: 0 = 0 / 100

plot(newtree,lwd=3)
text(newtree,cex=2)
dev.off();

```





# 6.867 Machine Learning

## Problem Set 4 Solutions

November 9, 2004

### Problem 1: Feature Selection

1. Based on the provided data we obtain the following entropies (all logarithms are to the base 2):

$H(Y)$	0.9321
$H(Y x = 0)$	0
$H(Y x = 1)$	0.65
$H(Y x = 2)$	1
$H(Y X)$	0.7739

Suppose now we add a new edge. Let  $H_1(Y)$  be the entropy at the root, and  $H_2(Y)$  be the entropy of the label distribution at the added node. Depending on which edge we add, the entropies take the following values:

	$H_1(Y)$	$H_2(Y)$	root count	leaf count	$H(Y X)$
edge = 0	0.9024	0	22	1	0.8614
edge = 1	0.9940	0.65	11	12	0.8145
edge = 2	0.7793	1	13	10	0.8753

Note that the entropy at the root changed from  $H(Y)$  to  $H_1(Y)$  because some of the samples at the root are now taken to the added node.

Because originally the conditional entropy was  $H(Y) = 0.9321$ , the reduction in conditional entropy due to the addition of an edge is:

edge = 0	0.0707
edge = 1	0.1176
edge = 2	0.0568

Therefore the best edge to add is 1.

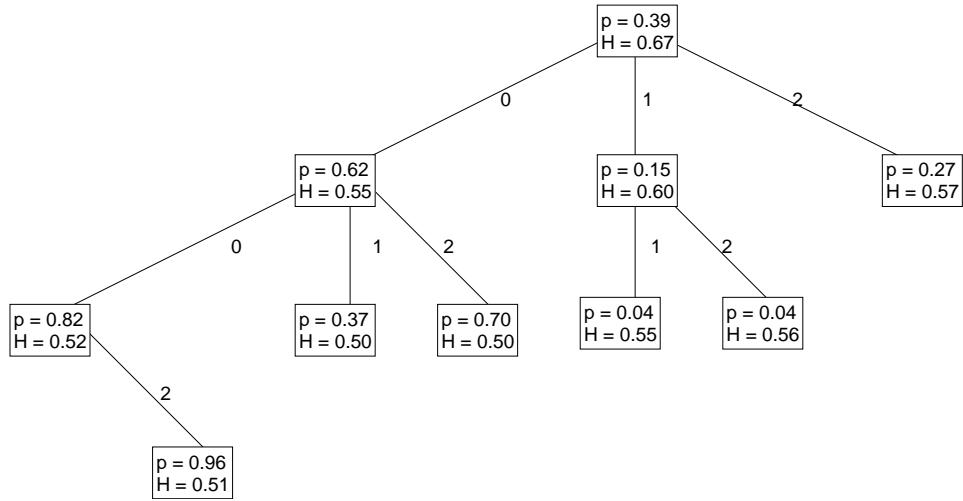


Figure 1: PST tree for Problem 1 Part 2

2. The error rates in the order in which the nodes are added are the following:

0.3846, 0.3846, 0.3846, 0.3846, 0.3846, 0.2912, 0.2912, 0.2912, 0.2912, 0.2474

The actual PST tree is show in Figure 1.

3.  $y = 1$  is most probable when we follow the path  $0 \rightarrow 0 \rightarrow 2$ . The corresponding rule is:

If a word starts with two consonants the next letter is very likely to be a vowel.

The path that makes  $y = 1$  least likely is either  $1 \rightarrow 1$  or  $1 \rightarrow 2$  (you can choose one). The corresponding rules are:

It is very unlikely for a vowel to follow two vowels.

or

It is very unlikely for a word that begins with a vowel to continue with a vowel.

These rules are consistent with common intuition about English.

4. No, we cannot extend a node past testing for “feature = 2”.

Once we the test for “feature = 2” succeeds, all following features will also be equal to 2, because we are past the beginning of the word. Thus all datapoints under this node have the exact feature representation, and we have no basis on which to split them into two groups to further reduce the conditional entropy.

## Problem 2

1. **Solution:**

$$\epsilon_m = \frac{1}{2} - \frac{1}{2} \sum_i \tilde{W}_i^{(m-1)} y_i h(x_i; \hat{\theta}_m) \quad (1)$$

$$= \frac{1}{2} \left[ \sum_i \tilde{W}_i^{m-1} - \left( \sum_{i: y_i = h(x_i; \hat{\theta}_m)} \tilde{W}_i^{(m-1)} - \sum_{i: y_i \neq h(x_i; \hat{\theta}_m)} \tilde{W}_i^{(m-1)} \right) \right] \quad (2)$$

$$= \frac{1}{2} \left[ \tilde{W}_+^{(m-1)} + \tilde{W}_-^{(m-1)} - \left( \tilde{W}_+^{(m-1)} - \tilde{W}_-^{(m-1)} \right) \right] \quad (3)$$

$$= \tilde{W}_-^{(m-1)} \quad (4)$$

■

2. **Solution:** As stated in the problem,

$$Z_m(\alpha_m) = \tilde{W}_+^{(m-1)} \exp(-\alpha_m) + \tilde{W}_-^{(m-1)} \exp(\alpha_m) \quad (5)$$

Differentiating w.r.t.  $\alpha_m$ ,

$$\frac{\partial Z_m(\alpha_m)}{\partial \alpha_m} = -\alpha_m \left( \tilde{W}_+^{(m-1)} \exp(-\alpha_m) \right) + \alpha_m \left( \tilde{W}_-^{(m-1)} \exp(\alpha_m) \right) \quad (6)$$

$$= 0 \quad (7)$$

Therefore,

$$\alpha_m = \frac{1}{2} \log \left( \frac{\tilde{W}_+^{(m-1)}}{\tilde{W}_-^{(m-1)}} \right) \quad (8)$$

Using the result from the previous section,

$$\alpha_m = \frac{1}{2} \log \left( \frac{1 - \epsilon_m}{\epsilon_m} \right) \quad (9)$$

■

3. Minimum value of  $Z_m(\alpha_m)$  is:

$$\begin{aligned} Z_{m,\min} &= \tilde{W}_+^{(m-1)} \exp \left( \frac{1}{2} \log \left( \frac{\tilde{W}_+^{(m-1)}}{\tilde{W}_-^{(m-1)}} \right) \right) + \tilde{W}_-^{(m-1)} \exp \left( \frac{1}{2} \log \left( \frac{\tilde{W}_+^{(m-1)}}{\tilde{W}_-^{(m-1)}} \right) \right) \\ &= \tilde{W}_+^{(m-1)} \sqrt{\frac{\tilde{W}_-^{(m-1)}}{\tilde{W}_+^{(m-1)}}} + \tilde{W}_-^{(m-1)} \sqrt{\frac{\tilde{W}_+^{(m-1)}}{\tilde{W}_-^{(m-1)}}} \\ &= 2 \sqrt{\tilde{W}_+^{(m-1)} \tilde{W}_-^{(m-1)}} \\ &= 2 \sqrt{(1 - \epsilon_m) \epsilon_m} \end{aligned}$$

Using induction on  $L(h_m) = L(h_{m-1})Z_m$ , we get

$$L(h_m) = \prod_{k=1}^m Z_k = \prod_{k=1}^m 2\sqrt{(1-\epsilon_k)\epsilon_k} \quad (10)$$

Since we know that the error is  $\leq L(h_m)$ , we get the required result. ■

#### 4. Solution:

- (a) `alpha = 0.5*log((1-stump.werr)/stump.werr);`
- (b) The following code snippet calculates the minimum voting margin among the training examples:

```
for k=1:num_iter
    [hhtrain,summtrain]=eval_boost(model(1:k),data.xtrain(:,1:2));
    votemargintrain(k)=(min(hhtrain.*data.ytrain))/summtrain;
end
figure;
plot(votemargintrain);
xlabel('Number of boosting iterations');
ylabel('Voting margin train');
title('Voting margin as a function of the number of iterations');
```

See Figures 2 and 3.

The test error decreases initially and then remains constant. However, the minimum voting margin on the training examples increases. It crosses zero (at which point all training samples are correctly classified), and converges to a value slightly less than 0.2. With more test samples, further reduction in (percentage) test error might have been observed.

- (c) See Figure 4. ■

- 5. **Solution:** No. We are jointly optimising the votes here (in contrast to the greedy optimisation in AdaBoost). Also, the votes need not be positive here. ■

## Problem 3

- 1. **Solution:** The VC dimension of the given class of classifiers is 3.

Three points in general position (*i.e.* 3 non-collinear points) can clearly be shattered by this set of classifiers. However, no set of 4 points can be shattered. To see this, we consider two cases (ignoring the cases of 3 or more collinear points, which can clearly not be shattered):

- The convex hull of the 4 points is a triangle, with one point lying strictly inside this convex hull: in this case, labelling the points at the vertices of the triangle as +1 and the interior point as -1 is not possible.
- The convex hull of the 4 points is a quadrilateral: in this case, one of the two labellings of non-adjacent vertices—both +1, remaining vertices -1, or both -1, remaining vertices +1—is not possible.

■

## 2. Solution:

- (a) VC dimension is 2. This can be seen as follows.

Given two decision stump classifiers  $h_1(x)$  and  $h_2(x)$ , the classifier obtained as a convex combination is given by  $\text{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x))$ . As stated in the problem set, a single vertical decision stump (and hence also a convex combination of two vertical decision stumps) can shatter 2 points in  $\mathbb{R}^2$ .

However, no set of 3 points can be shattered. For the purpose of labelling points using vertical decision stumps, we need only consider the horizontal coordinates of the points. Let these be  $x_1, x_2$  and  $x_3$ . Further, let  $\text{sign}(0)$  be equal to +1. In this case, the labelling  $x_1 = -1, x_2 = +1, x_3 = -1$  is not possible. Changing the definition of  $\text{sign}(0)$  does not help, as then the case  $x_1 = +1, x_2 = -1, x_3 = +1$  is not possible.

- (b) VC dimension is 3.

Consider 3 points that form an equilateral triangle, one of whose sides is parallel to the horizontal axis. Any required labelling of these 3 points can be obtained by using either a single horizontal or a single vertical decision stump. Thus, the set of convex combinations of a horizontal and a vertical decision stump (in particular, the subset where one of the two weights in the combination is unity and the other zero) can shatter 3 points.

No set of 4 points can be shattered by the given set of classifiers. Let us assume that  $\text{sign}(0)$  is equal to +1; it is easy to show that the opposite assumption leads to equivalent results. Then the possible decision regions include axis-aligned half spaces (with either label) or axis-aligned quadrants (with label -1). The latter case arises when the two stumps have equal weights in the convex combination.

Consider two cases (ignoring the cases of 3 or more collinear points, which can clearly not be shattered):

- \* The convex hull of the 4 points is a triangle, with one point lying strictly inside this convex hull: in this case, labelling the points at the vertices of the triangle as -1 and the interior point as +1 is not possible. This is because the interior point cannot lie in a half-space that does not contain any of the other three points.

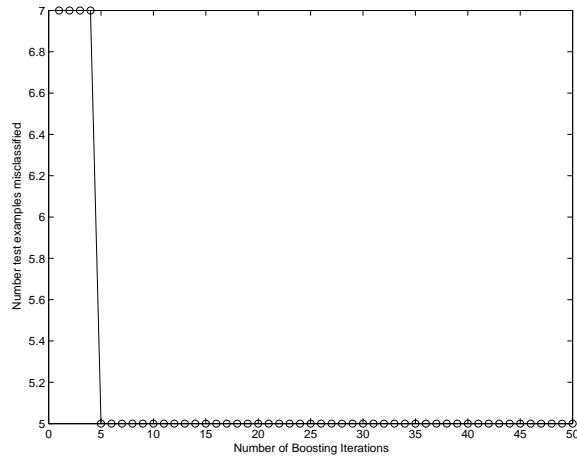


Figure 2: Test error for AdaBoost classifier

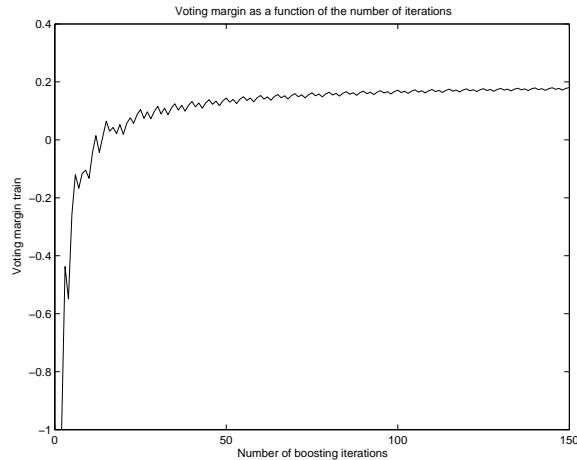


Figure 3: Minimum voting margin over training examples for AdaBoost classifier

- \* The convex hull of the 4 points is a quadrilateral: in this case, one of the two labellings of non-adjacent vertices—both +1, remaining vertices -1, or both -1, remaining vertices +1—is not possible.

■

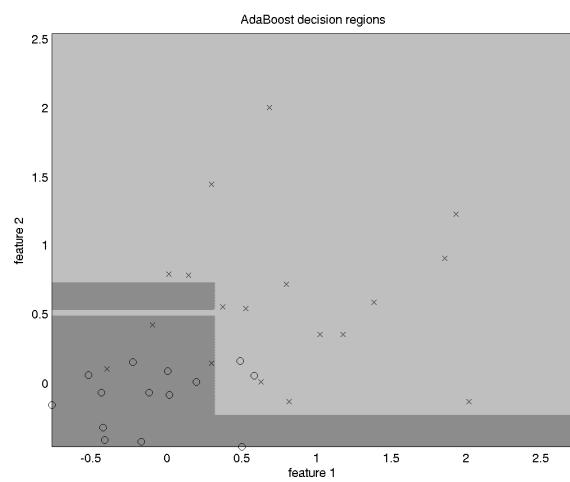


Figure 4: Decision regions for AdaBoost classifier

# Problem Set 4

## 10-601 Fall 2012

Due: Friday Nov. 9, at 4 pm

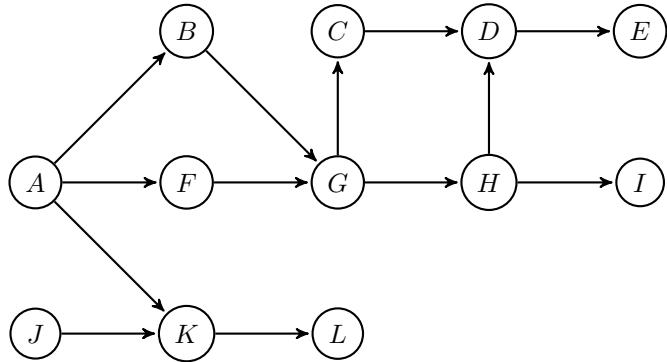
TA: Daegun Won (daegunw@cs.cmu.edu)

### Due Date

This is due at **Friday Nov. 9, at 4 pm**. Hand in a hard copy to Sharon Cavlovich, GHC 8215.

## 1 Bayesian Network

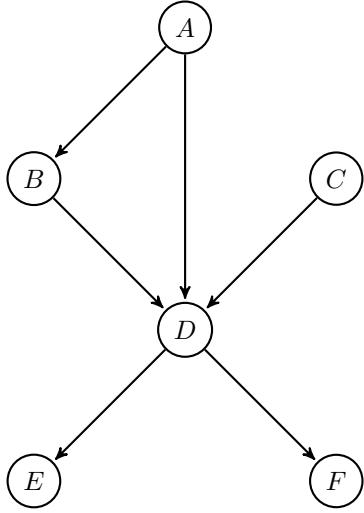
### 1.a d-separation



Which of the following statements are true given the Bayesian network below? For false statements, show one active trail.

1.  $P(H, J) = P(H)P(J)$  [ *Solution: true* ]
2.  $P(H, J|L) = P(H|L)P(J|L)$  [ *Solution: false*. There's an active trail JKAFGH ]
3.  $P(C, I|F) = P(C|F)P(I|F)$  [ *Solution: false*. CGHI is an active trail. ]
4.  $P(C, I|G, E) = P(C|G, E)P(I|G, E)$  [ *Solution: false*. CDHI ]
5.  $P(A, D|B) = P(A|B)P(D|B)$  [ *Solution: false*. AFGHD ]
6.  $P(B, F) = P(B)P(F)$  [ *Solution: false* BAF ]
7.  $P(C, K|B, F) = P(C|B, F)P(K|B, F)$  [ *Solution: true* ]
8.  $P(E, K|L) = P(E|L)P(K|L)$  [ *Solution: false*. KAFGHDE ]

## 1.b Variable Elimination



$$P(A = T) = 0.6, P(C = T) = 0.8$$

$$P(B = T|A = T) = 0.5, P(B = T|A = F) = 0.1$$

$$P(D = T|A = T, B = T, C = T) = 0.6, P(D = T|A = F, B = T, C = T) = 0.3$$

$$P(D = T|A = T, B = T, C = F) = 0.9, P(D = T|A = F, B = T, C = F) = 0.5$$

$$P(D = T|A = T, B = F, C = T) = 0.1, P(D = T|A = F, B = F, C = T) = 0.7$$

$$P(D = T|A = T, B = F, C = F) = 0.1, P(D = T|A = F, B = F, C = F) = 0.6$$

$$P(E = T|D = T) = 0.5, P(E = T|D = F) = 0.6$$

$$P(F = T|D = T) = 0.9, P(F = T|D = F) = 0.8$$

1. Using variable elimination, compute  $P(A = T, B = T, C = T, E = T, F = T)$ . Show your work. [\[ Solution: 0.11088\]](#)
2. From your work above, compute  $P(E = T, F = T)$  by removing  $B$ ,  $A$ , and  $C$  in order. Show your work. [\[ Solution: 0.465408\]](#)
3. Compute  $P(E = T, F = T)$  again but using elimination order of  $A, B, C$  and then  $D$ . [\[ Solution: 0.465408\]](#)
4. Would you say the order of variables matter in terms of final result? How about in terms of computational efficiency? [\[ Solution: It shouldn't affect the final value, but it affects the computational efficiency.\]](#)

### 1.c Constructing a Network

Let  $X, Y, Z$  be binary variables. After observing many instances of  $X, Y, Z$ , you summarized the data with the following joint distribution.

$X$	$Y$	$Z$	$P(X, Y, Z)$
0	0	0	0.042
0	0	1	0.378
0	1	0	0.054
0	1	1	0.126
1	0	0	0.140
1	0	1	0.140
1	1	0	0.096
1	1	1	0.024

Draw a Bayes net that can represent the above distribution with as few edges as possible. How many such networks are there? Show your work.

*[ Solution: If you look for marginal dependencies, only  $X$  and  $Y$  are marginally dependent. Thus the graph has to be in a form such as  $X-Z-Y$ . There are 4 possible BNs with two edges, but only  $X \rightarrow Z \leftarrow Y$  preserves the marginal dependence between  $X$  and  $Y$ . ]*

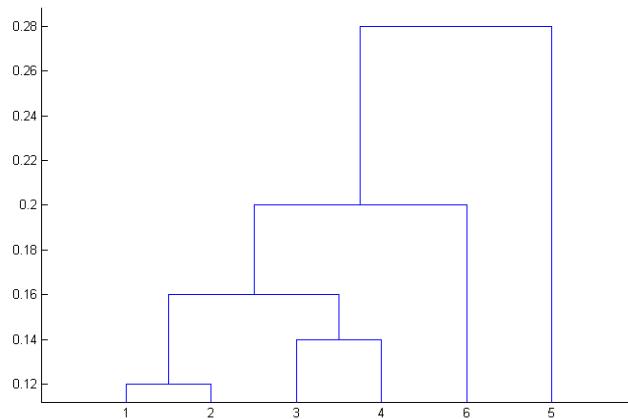
## 2 Clustering

The table below is a distance matrix for 6 objects.

	A	B	C	D	E	F
A	0					
B	0.12	0				
C	0.51	0.25	0			
D	0.84	0.16	0.14	0		
E	0.28	0.77	0.70	0.45	0	
F	0.34	0.61	0.93	0.20	0.67	0

### 2.a Hierarchical clustering

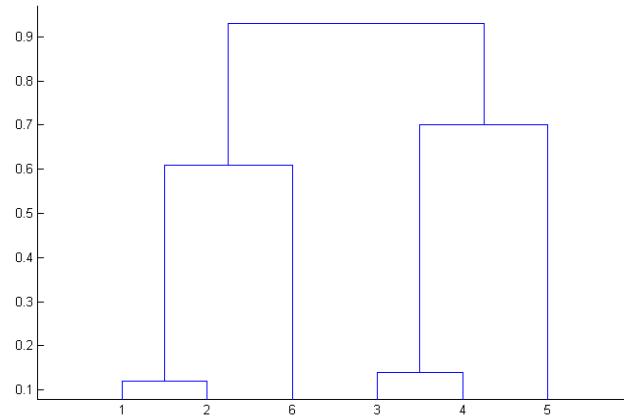
1. Show the final result of hierarchical clustering with single link by drawing a dendrogram.



[ Solution:

]

2. Show the final result of hierarchical clustering with complete link by drawing a dendrogram.



[ Solution:

]

3. Change **two** values from the matrix so that your answer to the last two question would be same.

[ Solution: There is more than one way possible, but one way would be the following:

The first step that the complete link clustering differs from the single link clustering is where AB and F are grouped together by  $\text{dist}(AB,F)=\text{dist}(B,F)=0.61$ . We'd want  $\text{dist}(AB, CD)=\text{dist}(A,D)$  to be smaller than this value, such as 0.53.

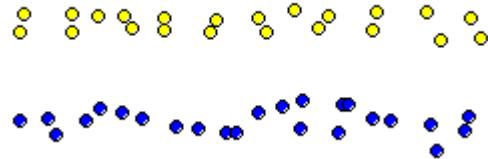
Then we want  $\text{dist}(\text{ABCD}, \text{F}) = \text{dist}(\text{C}, \text{F}) = 0.93$  to be the smallest so that ABCD and F are grouped together. We set this value to 0.63. After these changes both dendograms become identical. ]

## 2.b Which clustering method should we use?

Which clustering method(s) is most likely to produce the following results at  $k = 2$ ? Choose the most likely method(s) and briefly explain why it/they will work better where others will not in **at most 3 sentences**. Here are the five clustering methods you can choose from:

- Hierarchical clustering with single link
- Hierarchical clustering with complete link
- Hierarchical clustering with average link
- K-means
- GMM (with no assumption on the covariance matrices)

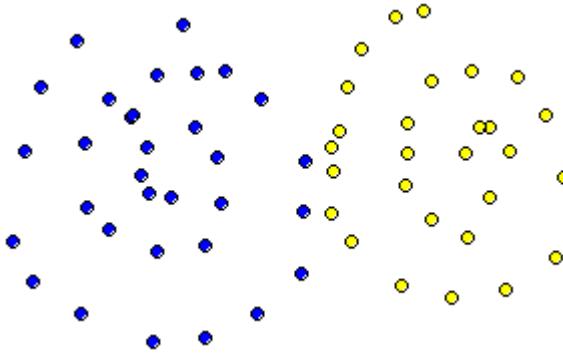
1.



[ *Solution:* Hierarchical clustering with single link is most likely to work well. GMM can also produce a decision boundary that can produce such clustering result, but depending on initialization it might converge to a different set of clusters (left half vs. right half). ]

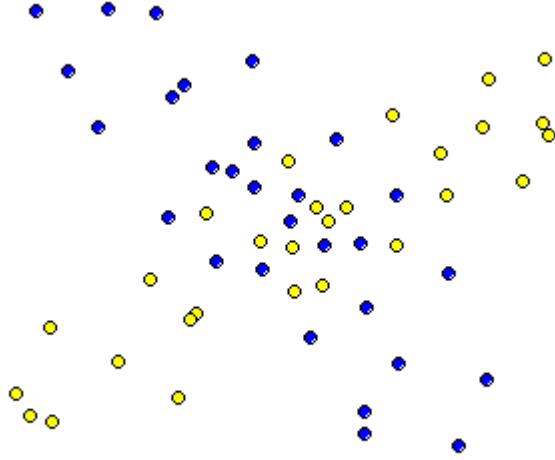
Other hierarchical clusterings won't really work well because at some point, two intermediate clusters from different true cluster will have shorter cluster distance than two from the same true cluster. ]

2.



[ *Solution:* K-means or GMM is most likely. Hierarchical clustering wouldn't work since the early few steps will group instances near the decision boundary (note some of them are very close). ]

3.



[ *Solution:* Among the five methods, only GMM has the capability of handling overlapping clusters. So GMM is the only method that would result in such clusters. ]

### 3 Semi-supervised learning

Let  $H$  be the set of all polynomials. Consider the following function  $d(h_1, h_2) : H \times H \rightarrow \mathbb{R}$ :

$$d(h_1, h_2) = \int |h_1(x) - h_2(x)|p(x)dx$$

#### 3.a

1. Show that  $d(h_1, h_2)$  is a distance metric.

[ *Solution:*

-Non-negativity: Both the absolute value and the pdf  $p(x)$  is nonnegative, thus the integrated value has to be nonnegative

-Symmetry:  $|h_1(x) - h_2(x)| = |h_2(x) - h_1(x)|$ , thus the value being integrated will be the same for  $d(h_1, h_2)$  and  $d(h_2, h_1)$

-Triangle inequality: We know that  $|a + b| \leq |a| + |b|$ , so

$$\begin{aligned} d(h_1, h_2) &= \int |h_1(x) - h_2(x)|p(x)dx \\ &= \int |h_1(x) - h_3(x) + h_3(x) - h_2(x)|p(x)dx \\ &\leq \int (|h_1(x) - h_3(x)| + |h_3(x) - h_2(x)|)p(x)dx \\ &= \int |h_1(x) - h_3(x)|p(x)dx + \int |h_3(x) - h_2(x)|p(x)dx \\ &= d(h_1, h_3) + d(h_3, h_2) \end{aligned}$$

]

2. Let  $L$  be a set of labeled instances,  $U$  be a set of unlabeled instances, and  $f$  be the true classifier. How would you estimate  $d(h_1, f)$  and  $d(h_1, h_2)$ ?

[ *Solution:*

$$d(h_1, h_2) = \frac{1}{|U|} \sum_{x \in U} |h_1(x) - h_2(x)|$$

$$d(h_1, f) = \frac{1}{|L|} \sum_{x \in L} |h_1(x) - f(x)|$$

]

### 3.b

Suppose you made the following observations from  $[0, 1] \times \mathbb{R}$ :

x	0.1	0.2	0.4	0.5	0.6	0.8	0.9	1.0
y	7.72	8.13	6.39	3.35	3.09	12.26	17.73	0.80

n	$h_n$	$\hat{d}(h_n, h_{n-1})$
1	$2.158x + 6.220$	
2	$6.498x^2 - 5.013x + 7.598$	0.451
3	$-175.6x^3 + 293.7x^2 - 133.9x + 20.98$	3.244
4	$-864.1x^4 + 1769x^3 - 1170x^2 + 278.5x - 11.14$	4.553
5	$-2297x^5 + 5417x^4 - 4477x^3 + 1570x^2 - 230.5x + 19.1$	3.315
6	$-2812x^6 + 6920x^5 - 6289x^4 + 2763x^3 - 671.8x^2 + 87.49x + 3.477$	1.171

- Let  $H$  be the set of all polynomials and  $h_n$  be your hypothesis of degree  $n$  minimizing the squared error (i.e.  $\sum_{(x,y)} (h_n(x) - y)^2$ ). Which  $n$  would you choose? Show your work. You may want to write a short Matlab program to do this part (you do not need to submit the code)

[ *Solution:*

Using the answer to 3.a.2,

n	$\hat{d}(h_n, h_{n-1})$	$\hat{d}(h_n, f) + \hat{d}(h_{n-1}, f)$
t2	0.451	8.0934
3	3.244	7.5618
4	4.553	5.5267
5	3.315	2.3702
6	1.171	

At  $n = 5$ , the triangle inequality does not hold, which indicates overfitting. So  $n = 4$ . ]

## 4 Programming (K-means)

In this problem we will implement K-means clustering. The data provided is a Matlab file of image data of 5000 handwritten digits. Each digit is a greyscale image of  $10 \times 10$  pixels and is represented as a row vector of length 100. The variable  $X$  contains all the images in a  $5000 \times 100$  matrix, and the vector  $Y$  contains the true label of each image.

- Implement K-means algorithm. For initial cluster centers, use random points. Repeat the random start 10 times for each clustering run. After getting the K-means result with 10 different initializations, how can you determine the best starting point? For the following questions, use the best initialization for your final result.

[ *Solution:* The best starting point is one whose final clustering result has the smallest objective value.]

2. We define the objective function of K-means as the sum of the squared distances of each point to its cluster centers,  $\sum_{k=1}^K \sum_{i=1}^{n_k} (x_{ki} - \mu_k)^2$ . Run your program with  $K = 10$  and plot the values of objective function against iterations. Is it monotonically decreasing?

[ *Solution: It should be monotonically decreasing* ]

3. Try running it with  $K = 16$  and plot the objective function again. How is the behavior of the objective function different from when  $K = 10$ ?

[ *Solution: The objective function converges at a lower value. Also takes more iterations to converge* ]

4. Clustering performance is hard to evaluate. However, since we have the true labels, we can use the following heuristics. For each cluster  $C$ , we find the most frequent (true) label  $Y_C$  and label the instances in that cluster with the majority label  $Y_C$ . Report your precision (number of correctly labeled instances / number of all instances) and final value of the objective function for  $K = 1, 5, 10, 16, 20$ .

[ *Solution:* ]

K	Precision	Obj. Func
1	0.1140	1.2762e+09
5	0.4482	9.5501e+08
10	0.5912	8.1849e+08
16	0.6842	7.3375e+08
20	0.7170	6.9961e+08

[ ]

5. Among the five values you tried above, what would you choose to be the optimal number of clusters and why?

[ *Solution: Using knee/elbow finding, it looks like 10 (or 5) should be the optimal number. Some said that there is no clear knee/elbow, that's also acceptable given the graph.* ]

# 10702/36702 Statistical Machine Learning, Spring 2008: Homework 5 Solutions

May 9, 2008

## 1 [20 points], (Robin)

### ★ SOLUTION:

(a) Take any set of  $N$  atoms  $J_N$  from the total set  $\mathcal{D}$ . The distance between  $f$  and  $g$  is then:

$$\begin{aligned}
 \|f - g\| &= \|f - f_{J_N}\| \\
 &= \left\| \sum_{\mathcal{D}} \beta_j \psi_j - \sum_{J_N} \beta_j \psi_j \right\| \\
 &= \left\| \sum_{\mathcal{D} - J_N} \beta_j \psi_j \right\| \\
 &= \sqrt{\sum_{\mathcal{D} - J_N} \beta_j^2 \langle \psi_j, \psi_j \rangle} \\
 &= \sqrt{\sum_{\mathcal{D} - J_N} \beta_j^2}
 \end{aligned}$$

which is minimized when  $J_N$  is over the  $N$  largest values of  $|\beta_j|$ .

(b) In OGA, at each step  $r_{N-1} = f - f_{J_{N-1}} = \sum_{\mathcal{D} - J_{N-1}} \beta_j \psi_j$  where  $J_{N-1}$  is a set of functions selected so far. To choose the next function, compute

$$\begin{aligned}
 cp_{N,i} &= |\langle r_{N-1}, \psi_i \rangle| \\
 &= \left| \sum_{\mathcal{D} - J_{N-1}} \beta_j \langle \psi_j, \psi_i \rangle \right| \\
 &= |\beta_i \langle \psi_i, \psi_i \rangle + \sum_{\mathcal{D} - J_{N-1} - i} \beta_j \langle \psi_j, \psi_i \rangle| \\
 &= |\beta_i|
 \end{aligned}$$

The maximum is achieved at  $\text{argmax}_i |\beta_i|$ . Hence OGA recovers  $f_N$  exactly.

(c)

$$\begin{aligned}
\sigma_N(f) &= ||f - f_N|| = \sqrt{\sum_{\mathcal{D} - J_N} \beta_j^2} \\
&< \sqrt{\sum_{N+1}^{|\mathcal{D}|} \frac{C^2}{j^{2/p}}} < \sqrt{\sum_{N+1}^{|\mathcal{D}|} \frac{C^2}{(N+1)^{2/p}}} \\
&< \sqrt{\frac{|\mathcal{D}|C^2}{(N+1)^{2/p}}} < \sqrt{\frac{(N+1)C^2}{(N+1)^{2/p}}} \\
&= \sqrt{\frac{C^2}{(N+1)^{\frac{2}{p-1}}}} = O\left(\frac{1}{N^{1/p-1/2}}\right) \\
&= O\left(\frac{1}{N^s}\right)
\end{aligned}$$

## 2 [20 points], (Robin)

★ SOLUTION: Start with Bernstein's inequality

$$P(|\bar{X}_n| > t) \leq 2e^{-\frac{nt^2}{2\sigma^2 + \frac{2ct}{3}}}$$

Then substitute  $t = \sigma\sqrt{\frac{2\delta}{n}} + \frac{2c\delta}{3n}$  and simplify to get

$$P(|\bar{X}_n| > \sigma\sqrt{\frac{2\delta}{n}} + \frac{2c\delta}{3n}) \leq 2e^{-\delta}$$

## 3 [20 points], (Robin)

★ SOLUTION:

- (a) When  $r = 1$  we have  $u(x) - l(x) \leq \epsilon$ , i.e. the bracket is in the ball of  $\epsilon/2$  around  $(u + l)/2$ , the center of the bracket. This is also the  $\epsilon/2$  cover. Since the bracket is contained in within this ball, we need more  $\epsilon$ -brackets than the  $\epsilon/2$  cover to cover the function. Hence,

$$N_1(\epsilon/2, \mathcal{F}) \leq N_{[]}(\epsilon, \mathcal{F}, L_1(P))$$

From Theorem 1.46 in notes we have

$$\begin{aligned}
P(\sup_{f \in \mathcal{F}} |P_n(f) - P(f)| > \epsilon) &\leq 8N_1(\epsilon/8, \mathcal{F})e^{-n\epsilon^2/128B^2} \\
&\leq 8N_{[]}(\epsilon/4, \mathcal{F}, L_1(P))e^{-n\epsilon^2/32B^2} \\
&\rightarrow 0 \quad \text{as } n \rightarrow \infty
\end{aligned}$$

Hence proved.

- (b) we can create the  $\epsilon$ -brackets as follows. Let  $-\infty = t_0 < t_1 < \dots < t_k = \infty$  where  $t \in \mathbb{R}$ . We can choose the bracketing functions themselves to be indicator functions. Let  $I_{t_i} = I_{(-\infty, t_i]}$ . For  $\epsilon$ -bracket, we have  $\int_{-\infty}^{\infty} (u(x) - l(x))p(x)dx \leq \epsilon$ . Hence,

$$\int_{-\infty}^{\infty} (I_{t_i}(x) - I_{t_{i-1}}(x))p(x)dx \leq \epsilon \quad \forall i = 0, 1, \dots, k$$

But if  $x < t_{i-1}$  then  $I_{t_i}(x) = I_{t_{i-1}}(x)$  and if  $x \geq t_{i-1}$  then  $I_{t_i}(x) - I_{t_{i-1}}(x) = 1$ . So,  $\int_{t_{i-1}}^{t_i} p(x) \leq \epsilon$ . Hence, each bracket consumes a probability mass of atmost  $\epsilon$  and since the total probability mass is 1, there are  $1/\epsilon$  such brackets consuming the entire mass of 1 which is bounded by  $2/\epsilon$  (if  $1/\epsilon$  is not a whole number). Hence proved.

## 4 [20 points], (Robin)

★ SOLUTION: The VC dimension of axis-aligned rectangles in  $R^d$  is  $2d$ .

- (1) Show that the  $VC - dim \geq 2d$ .

Consider a set of  $2d$  points where each point only has one of the  $d$  dimensions set to either 1 or  $-1$  and 0 for all other dimensions. It is easy to see that any subset of these points can be shattered by an axis-aligned rectangle. Hence the VC-dim is atleast  $2d$ .

- (2) Show that the  $VC - dim < 2d + 1$ . Consider a set of  $2d+1$  points. Consider finding the minimum and maximum of value in each dimension for these set of points and then building a  $R^d$  rectangle with these bounds. Since there are  $2d+1$  points, atleast one point must lie inside this rectangle. If we label this interior point as negative then there is no rectangle that can separate this labeling. This proves that  $VC - dim < 2d + 1$ .

Combining (1) and (2) we get that the  $VC - dim = 2d$ . Intuitively, there are  $2d$  free parameters (lower and upper bound in each dimension) of the rectangle and hence the VC-dimension is  $2d$ .

## 5 [20 points], (Robin)

★ SOLUTION:

- (a) You can assume  $k=6$  since we need to compare the predicted and true clusters. Also, you need to define a good comparison metric.

One such metric is to penalize a pair of points that belongs to the same true cluster but is present in different predicted clusters and vice versa. You can then report the probability of error over all pairs of points in the data.

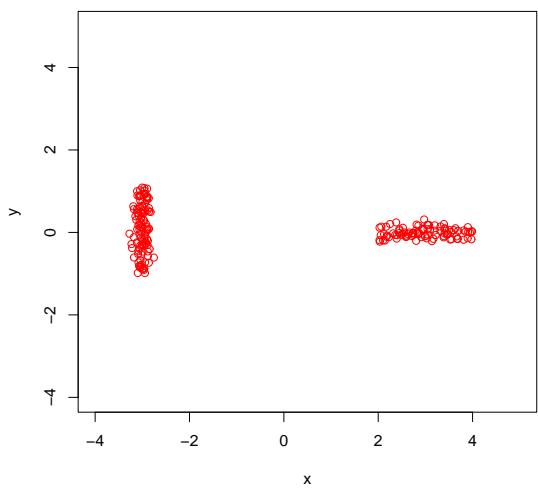
Under this scoring metric, K-Means error is 0.13 and Hierarchical clustering error is 0.16. This shows that K-Means performs better on the data.

- (b) Note that here we need to select a subset of original features and not use any kind of projection or transformation of data. Some possibilities for feature selection are: (1) apply kmeans using each feature and select the set of features greedily with minimum distortion in predicted clusters (2) assume that the clusters must have similar number of points ad hence use entropy measure to select the features.

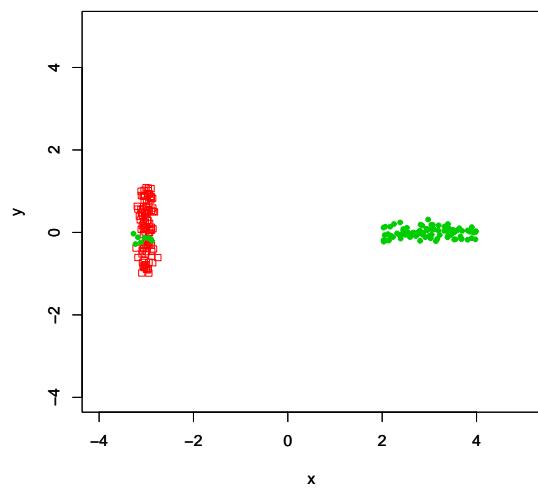
Using the first technique, K-Means error with top 15 features is 0.15 which is only 2% more than using all features, i.e. with a only quarter of original features.

## 6 [20 points], (Robin)

★ SOLUTION:



(a) Original data



(b) Clustered data

# 6.867 Machine Learning

## Problem Set 5 Solutions

Due date: Monday November 22

### Problem 1: Model Selection

1.

$$\begin{aligned} P(y_1, \dots, y_n | \mathbf{x}_1, \dots, \mathbf{x}_n, \text{PST}) &= \int_{[0,1]^K} \left[ \prod_{i=1}^K p_i^{n_i^+} (1-p_i)^{n_i^-} \right] dp_1 dp_2 \dots dp_K = \\ &= \prod_{i=1}^K \left[ \int_0^1 p_i^{n_i^+} (1-p_i)^{n_i^-} dp_i \right] = \prod_{i=1}^K \frac{n_i^+! n_i^-!}{(n_i + 1)!} \end{aligned}$$

2.

$$\sum_{i=1}^K \log \hat{p}_i^{n_i^+} (1 - \hat{p}_i)^{n_i^-} - \frac{d}{2} \log n = \sum_{i=1}^K (n_i^+ \log n_i^+ + n_i^- \log n_i^- - n_i \log n_i) - \frac{d}{2} \log n$$

3. Possible implementation of `log_bayesian_model_score`:

```
% You can assume n_plus and n_minus are never 0 together
%
function score = log_bayesian_model_score(n_plus, n_minus)

% total counts for each node
n = n_plus + n_minus;

%
% SCORE COMPUTATION
%
score = sum(log_fact(n_plus) + log_fact(n_minus) - log_fact(n+1));

% -----
function [logfac] = log_fact(n)

logfac = gammaln(n+1);
```

Possible implementation of `bic_model_score`:

```
% You can assume n_plus and n_minus are never 0 together
%
function score = bic_model_score(n_plus, n_minus)

K = length(n_plus);
n = n_plus + n_minus;

% maximum likelihood probabilities
Pml = n_plus./n;
Pml(n_plus == 0) = eps;
Pml(n_minus == 0) = 1 - eps;

%
% SCORE COMPUTATION
%
score = sum(n_plus .* log(Pml) + n_minus .* log(1 - Pml)) - 0.5*K*log(sum(n));
```

The plots of the resulting PST's are shown in Figure 1.

4. The generated plot is shown in Figure 2. We can interpret the properties of the plot in the following manner:
  - The curves are negative for small sample sizes because the simpler models will always be preferred if data is not sufficient.
  - The curve of the Bayesian score becomes positive for smaller sample size because in this task the BIC score tends to prefer simpler models than the BIC score. In general this may not always be the case, because Bayesian selection depends on the prior, while BIC does not.
  - At large sample sizes the two curves converge to each other because the BIC score is a large-sample approximation of the log-Bayesian score (up to an additive constant that cancels out when taking the difference of scores).
  - At large sample size the log-likelihood of the data at the ML parameters becomes linear in the number of samples, because the ML parameters converge to their asymptotic value. Since the BIC complexity penalty is sublinear, it follows that at large sample size the BIC score becomes linear. Thus the difference in BIC scores for two models will also be asymptotically linear. The slope of the difference is positive or negative depending on which of the two models is more complex (in this case positive).

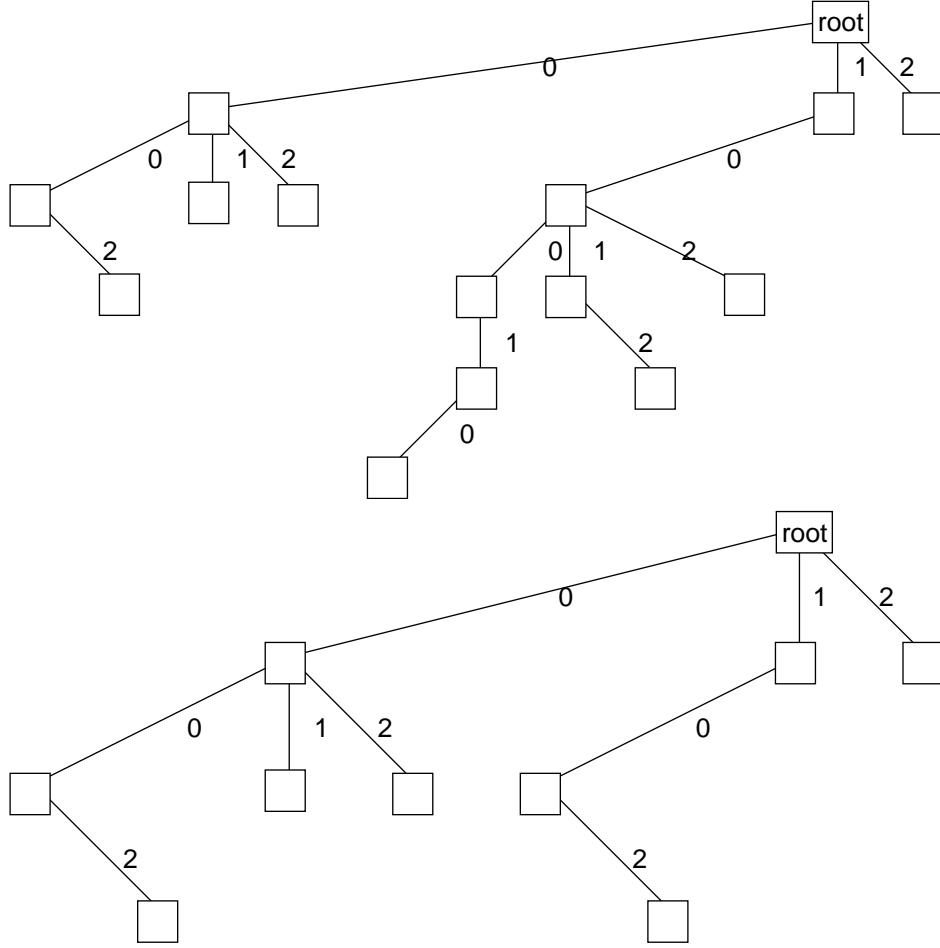


Figure 1: Problem 1 Part 3: PST's trained with Bayesian model selection (above) and the BIC score (below). Note that the second tree is a subset of the first tree.

## Problem 2

- Solution:** The hidden variables for a training point  $x_i$  are  $j_i$  and  $k_i$ .

$$p(j_i, k_i | x_i, \theta^c) = \frac{p(x_i | j_i, k_i, \theta^c) p(j_i, k_i | \theta^c)}{p(x_i, \theta^c)} \quad (1)$$

$$= \frac{p_{j_i}^c q_{k_i}^c N(x_i; \mu_{j_i}^c, \sigma_{k_i}^{c2})}{\sum_{j=1}^m \sum_{k=1}^l p_j^c q_k^c N(x_i; \mu_j^c, \sigma_k^{c2})} \quad (2)$$

■

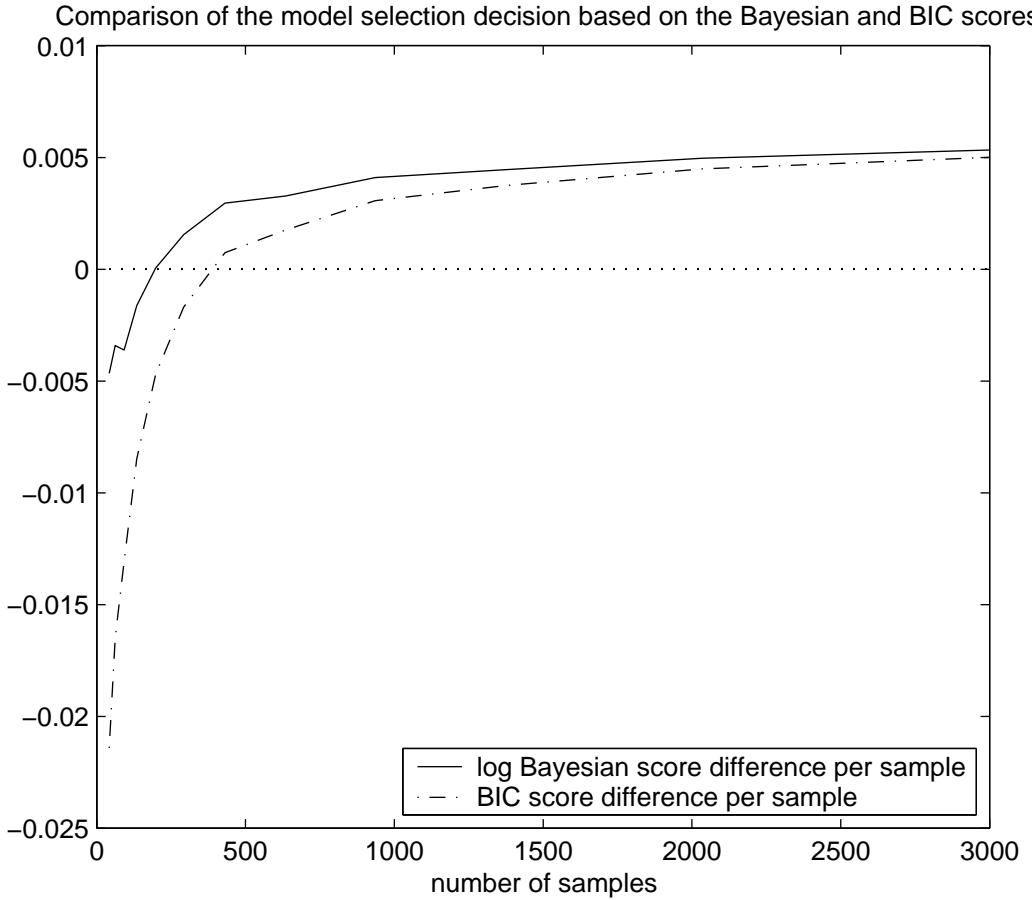


Figure 2: Problem 1 Part 4: Comparison of the log Bayesian and BIC scores on two models

2. **Solution:** The expression for the expected complete log-likelihood (ECLL) is:

$$\text{ECLL} = \sum_{i=1}^n E \left[ \log(p_{j_i} q_{k_i} N(x_i; \mu_{j_i}, \sigma_{k_i}^2)) | x_i, \theta^c \right] \quad (3)$$

$$= \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^l p(j, k | x_i, \theta^c) \log(p_j q_k N(x_i; \mu_j, \sigma_k^2)) \quad (4)$$

The terms in the ECLL depending on the mean parameters are therefore:

$$\text{ECLL}(\mu) = \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^l p(j, k | x_i, \theta^c) \left( \frac{1}{2\sigma_k^2} (x_i - \mu_j)^2 \right) \quad (5)$$

■

3. **Solution:** Differentiating w.r.t.  $\mu_j$ ,

$$\sum_{i=1}^n \sum_{k=1}^l p(j, k|x_i, \theta^c) \frac{(x_i - \mu_j)^2}{2\sigma_k^2} = 0 \quad (6)$$

$$\Rightarrow \mu_j = \frac{\sum_{i=1}^n \sum_{k=1}^l p(j, k|x_i, \theta^c) (x_i / (\sigma_k^2))}{\sum_{i=1}^n \sum_{k=1}^l p(j, k|x_i, \theta^c) / (\sigma_k^2)} \quad (7)$$

■

4. **Solution:** We plotted the maximum likelihood solutions after 4 runs of the EM algorithm for each setting of  $m$  and  $l$ . See Figure 3 for the plots.

A model where components can have two different variance parameters cannot be sufficiently well approximated by a model consisting of a small number of components where all components have a single variance parameter. In other words, the number of components with a single common variance required to approximate the underlying two-variance distribution accurately (using Parzen density estimation, for example) is much larger than 4. ■

## Problem 3

1. **Solution:** For the data in `X1.mat`, the typical  $k$ -means solution is shown in Figure 4. This clearly does not correspond to the intuitive notion of clustering, where the clusters should be the inner and outer rings. The reason for this is that the clusters found by  $k$ -means are a partition (tessellation) of space into polygonal regions separated by straight line-segments. For  $k = 2$ , the two clusters must be separated by a straight line.

For `X2.mat`, the typical  $k$ -means solution is shown in Figure 5. However this is not the only solution possible, since the initialisation of the means is random. Other solutions, such as those shown in Figure 6 occur about once in seven trials.

The typical solution here does correspond to the intuitive notion of clusters: groups of closely spaced points, where distance between groups are typically larger than distances within the group. The two point in the top-left part of the space would be considered outliers, since there are many more points in the ‘clusters’ in the bottom-right part. The  $k$ -means algorithm assigns these two points to one or the other cluster seemingly randomly. ■

2. **Solution:** See Figures 7 and 8 for clustering results with data in `X1.mat`, and Figures 9 and 10 for corresponding results with `X2.mat`.

The spectral clustering algorithm discussed in class has two parameters: the number of nearest neighbours to which each point is connected,  $r$ , and the exponential decay parameter,  $\beta$ .

First let us consider the data in `X1.mat`. When  $r = 5$ , the corresponding graph is not connected, but consists of two subgraphs (each of which is connected). In this case, the largest eigenvalue calculated in the spectral clustering solution is not unique, but has multiplicity two. Thus, the first and second eigenvectors are equivalent, and both of them appear in the expression for the asymptotic transition probability matrix,  $P^\infty$ . Further, neither of these vectors has all elements equal, nor does either correspond to the first order correction term to  $P^\infty$ . So looking at the sign of the elements of the second eigenvector is practically meaningless. In particular, the first two eigenvectors have zero elements corresponding to points in one of the two clusters whenever the graph is disconnected. This can be seen from Figure 11 in the present case. Thus, the resulting clusters depend on how Matlab chooses to interpret the zero elements, as a consequence of finite precision arithmetic. For instance, running the same code with  $r = 4$  gives different results, with the inner ring containing both red and blue points (see Figure 12). With  $r = 2$ , all the points—both inner and outer rings—are considered to be belonging to the same cluster.

However, there is a second way of interpreting the eigenvectors of the normalised affinity matrix. When the affinities (or distances) within each cluster are all constant, and the distances between points belonging to different clusters are also (some other) constant, the two eigenvectors are piecewise constant (i.e. the elements of the eigenvectors are constant for all indices corresponding to a distinct cluster). Hence, clustering the elements of these two eigenvectors (using k-means clustering) provides us the indices of the points belonging to clusters in the original space.

Even if the affinity matrix is not exactly block constant, the eigenvector elements are still approximately piecewise constant as long as the cluster structure is clearly present in the data. This can be seen from both the first and second eigenvectors in the present case. For instance, the elements of the latter tend to cluster around 0 and  $-0.16$ , and can easily be separated by k-means clustering.

When  $r = 20$ , the graph becomes connected. Thus, the spectral clustering solution is a valid one. However, for the default value of the exponential decay parameter ( $\beta = 1$ ), we find that the clustering does not separate the rings. This is because there are now many connections between the inner and outer rings, and the solution found is essentially the same as that using k-means clustering on the data.

The ‘correct’ solution—the one separating the inner and outer rings—can be obtained by increasing  $\beta$  to 5, which makes longer distance transitions in the random walk more unlikely (see Figure 13).

Next, consider the data in `X2.mat`. Here, when  $r = 5$ , the corresponding graph is connected, by virtue of the two outlying points. However, even though connections within each cluster are strong, there are no direct connections between points in the

two clusters. Thus, the spectral clustering solution is the one we expect best explains the underlying data.

We have plotted the second eigenvector for this spectral clustering solution in Figure 14. As explained above, there are two clearly identifiable clusters of eigenvector elements (around 0.1 and  $-0.1$ ), due to tight clustering evident from the data. However, note the two eigenvector elements with values near 0. These indicate the possible presence of a third cluster, and correspond to the two outlying points.

When  $r = 20$ , we find new connections between points in distinct clusters. Now, the random walk can start at a point in one cluster and end at a point in the other without having to transition through the outlying points. In fact, since the outlying points lie far away from the two clusters, they are connected to the latter by edges with very small weights. In contrast, the weights associated with the edges directly connecting the two underlying clusters are very large. Thus, spectral clustering effectively groups the two clusters into one, and labels the two outlying points as the other cluster. ■

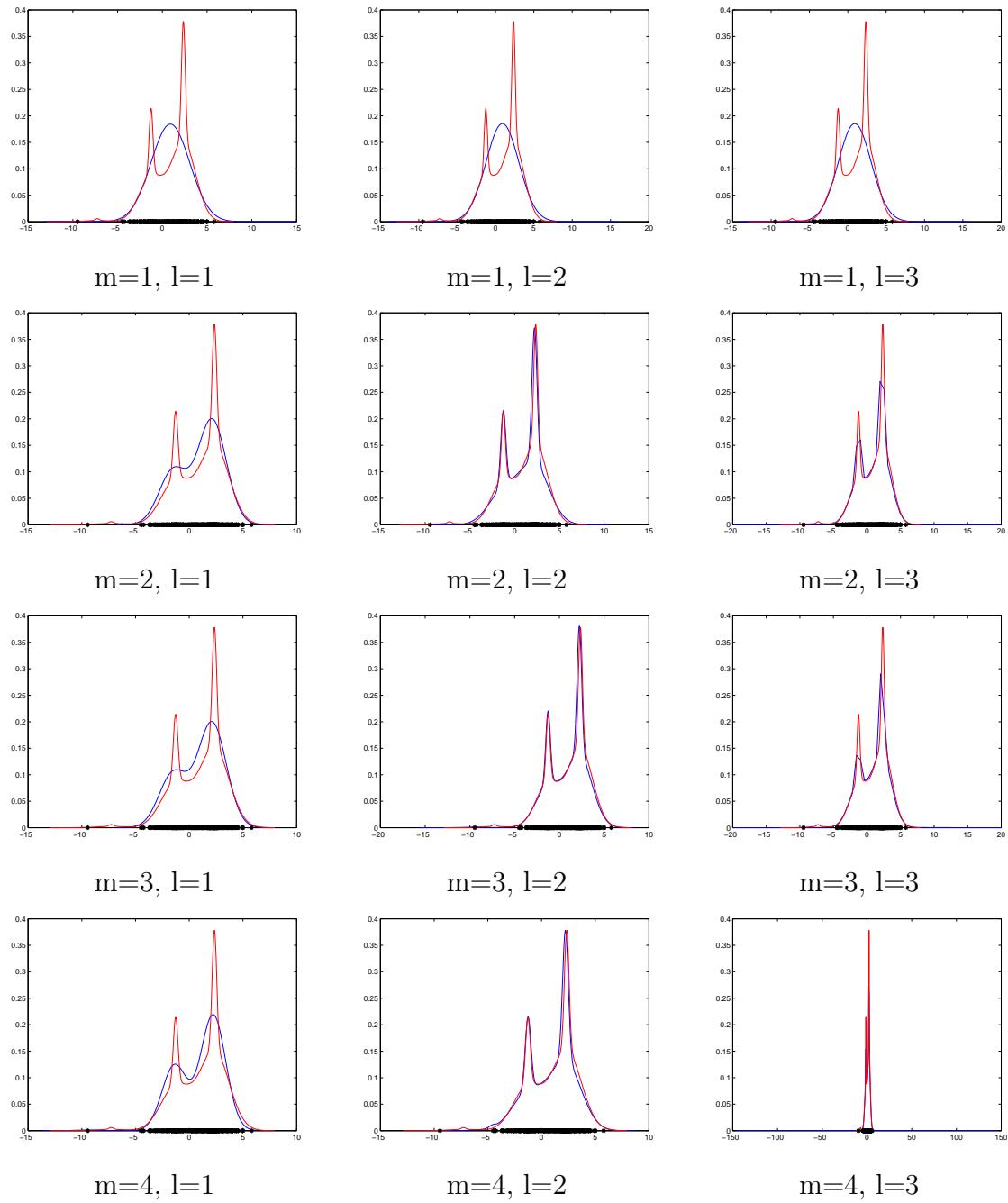


Figure 3: Samples, estimated densities and true densities, for different values of  $l$  and  $m$

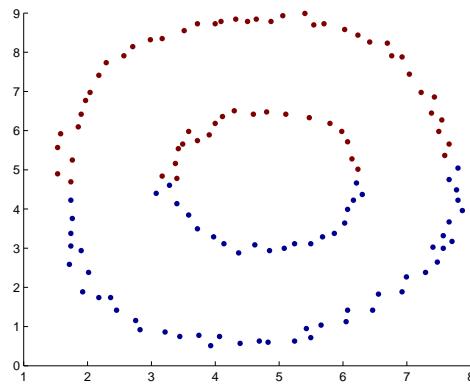


Figure 4: Typical clusters found by  $k$ -means ( $k = 2$ ) for data in `X1.mat`

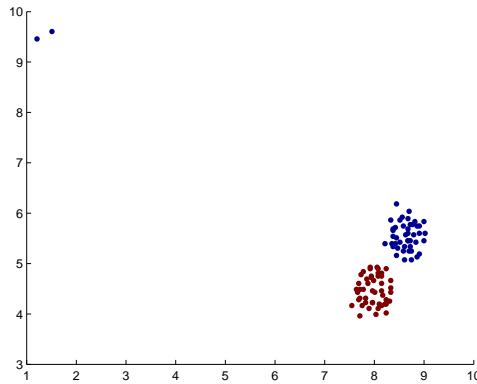


Figure 5: Typical clusters found by  $k$ -means ( $k = 2$ ) for data in `X2.mat`

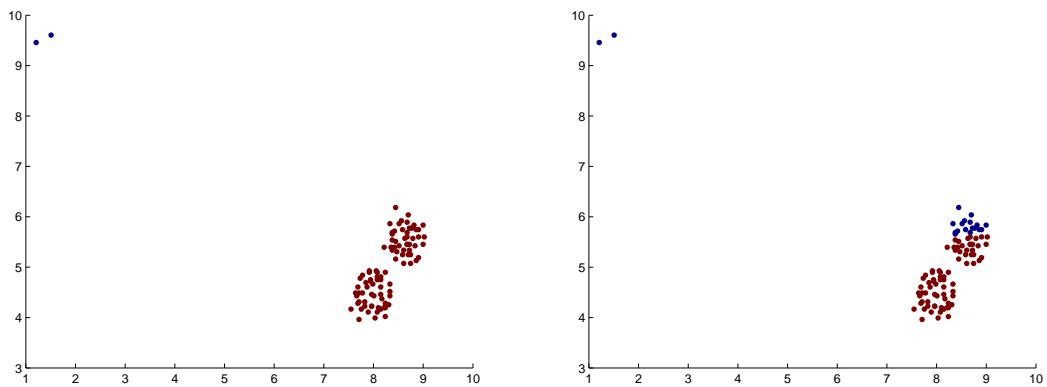


Figure 6: Other (less typical) clustering solutions found by  $k$ -means ( $k = 2$ ) for `X2.mat`

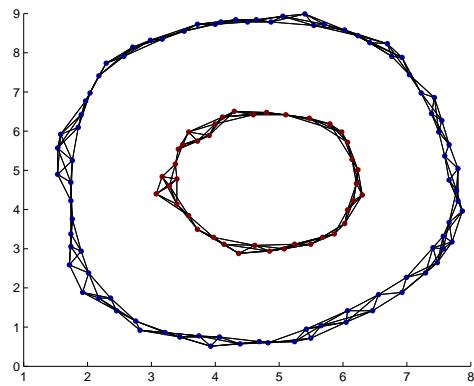


Figure 7: Neighbourhood graph and clusters for data in `X1.mat`, with  $r = 5$

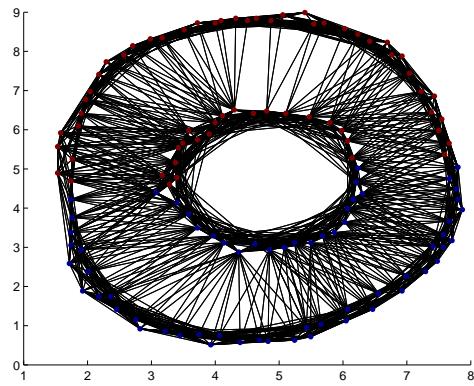


Figure 8: Neighbourhood graph and clusters for data in `X1.mat`, with  $r = 20$

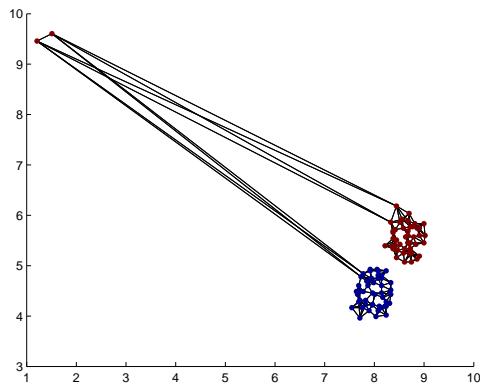


Figure 9: Neighbourhood graph and clusters for `X2.mat`, with  $r = 5$

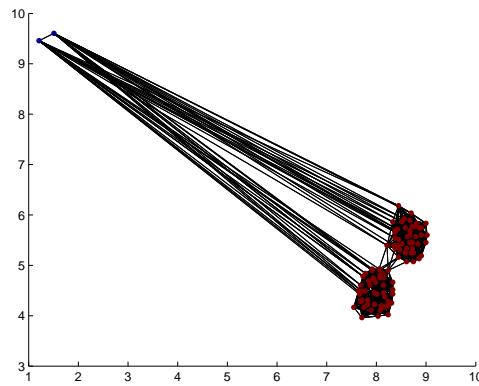


Figure 10: Neighbourhood graph and clusters for `X2.mat`, with  $r = 20$

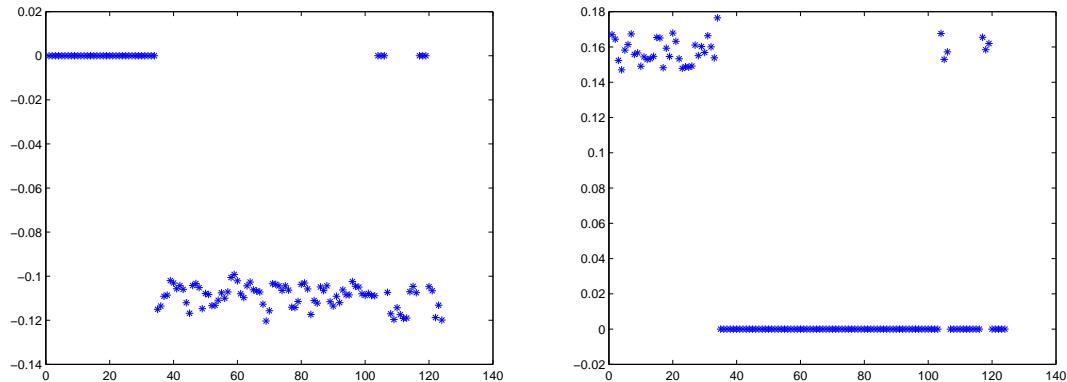


Figure 11: First two eigenvectors (plotted elementwise), with  $r = 5$  for data in `X1.mat`

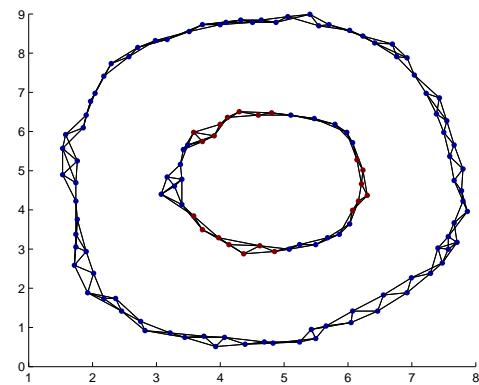


Figure 12: Neighbourhood graph and clusters for `X1.mat`, with  $r = 4$

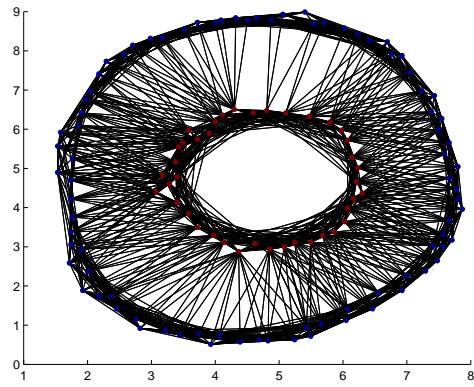


Figure 13: Neighbourhood graph and clusters for `X1.mat`, with  $r = 20$  and  $\beta = 5$ . Notice that the ‘correct’ clustering is now obtained.

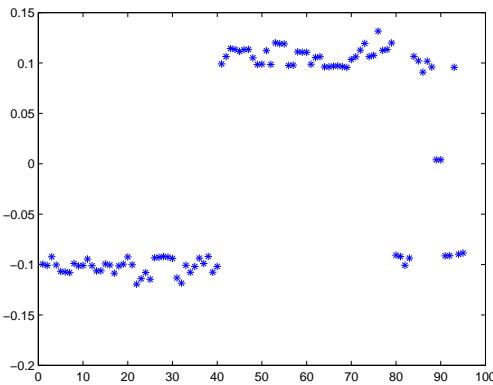


Figure 14: Second eigenvector (plotted elementwise), with  $r = 5$  for data in `X2.mat`

# Problem Set 5

10-601 Fall 2012

Due: Friday Nov 30th, by 4pm

Please write both your Andrew ID and name on the assignment.

TAs: Selen and Brendan

## 1 Bayes Net questions [Brendan]

### 1.a Legal Bayes Nets [10 points]

Prove that in every Bayesian network there is at least one node with **no** incoming edge.

*[Solution:*

By definition, a Bayes Net is a directed acyclic graph. If every node had an incoming edge, we could construct a cycle: start at one node, proceed to one of its parents, and repeat. If all nodes have incoming edges, this process could be repeated infinitely; but if it is repeated more times than the number of nodes, it is necessary to visit some node twice, thus a cycle. Therefore, in a BN it cannot be the case that every node has at least one incoming edge.

*End solution]*

### 1.b Stochastic inference [10 points]

Prove that when performing stochastic inference, if not all nodes have been sampled then there is always at least one **unsampled node** that either

- Does not have any parent, or
- All its parents have been already sampled

*[Solution:*

Simplest solution: Consider the set of unsampled nodes. They and all edges connecting them must form a DAG, since it is a subgraph of the overall BN which is a DAG. By the previous question, it must contain a node without a parent (in the subgraph). This node must have had all its parent been sampled, by definition of the subgraph. (Or this condition vacuously holds because it has no parents in the original DAG.)

Another view: here is pseudo-code for the sampling algorithm.

Input: a query, which is a set of nodes (random variables). Goal is to create a single sample: values for those query nodes.

Maintain a data structure  $S$ : a to-be-sampled set of nodes. (And also a set of already-been-sampled nodes)

Procedure:

1. Initialize  $S :=$  all roots (i.e. nodes without parents)
2. Sample values for all variables in  $S$ .
3. Let  $S := \{ \text{all nodes, that haven't been sampled yet, for whom all their parents have been sampled} \}$
4. Go to step (2). Terminate when all nodes involved in the query have been sampled.

So when  $S$  is non-empty there are several possible cases. First, it could be one of the root variables (that have no parent). Otherwise, a variable is added to  $S$  only when all its parents have already been sampled. (Because you can't sample a variable until its parents have been sampled). Those are the two possibilities the question asked us to verify.

Another popular way to solve this problem was proof by contradiction.

*End solution]*

## 2 Hidden Markov Models [30 points] [Selen]

Hidden Markov Models (HMMs) are probabilistic models that are used in a wide variety of sequence analysis problems. We define an HMM for  $K$  classes of hidden states and  $T$  data points. Let the data set be  $\mathbf{X} = \{x_1, \dots, x_T\}$ , where each  $x_i$  a discrete observed variable. Hidden state variables are  $\mathbf{Z} = \{z_1, \dots, z_T\}$ , where each hidden state is  $z_t \in \{1..K\}$ .

The transition probabilities are given by a  $K \times K$  matrix  $\mathbf{A}$ , where  $a_{kj} = P(z_t = k | z_{t-1} = j)$ . The initial state variable  $z_1$  is special since it does not have a parent node. Its distribution can be represented by a vector of probabilities  $\pi$  where  $P(z_1) = \pi_{z_1}$ . Finally, the emission distribution for a hidden state class  $k$  is parametrized by  $\vec{\phi}_{.k}$ , where  $\phi_{xk} = P(x_i = x | z_i = k)$ . Let  $\Theta = \{\mathbf{A}, \pi, \phi\}$ .

### 2.a The full likelihood of a data set

If we have a data set  $\mathbf{X} = \{x_1, \dots, x_T\}$ , write the following expressions in terms of the parameters.

1. [2 points] Write down the the full likelihood of observed and latent variables,  $P(\mathbf{X}, \mathbf{Z} | \Theta)$ .

*[Solution:]*

$$\begin{aligned}
 P(\mathbf{X}, \mathbf{Z} | \theta) &= P(\mathbf{X} | \mathbf{Z}, \theta)P(\mathbf{Z} | \theta) \\
 &= \prod_{t=1}^T P(x_t | z_t)P(z_1) \prod_{t=2}^T P(z_t | z_{t-1}) \\
 &= \prod_{t=1}^T \phi_{x_{tk}}^t \pi_{z_1} \prod_{t=2}^T a_{kj}^t \\
 &= \prod_{k=1}^K \pi_{z_1}^{\delta(z_{1k})} \prod_{t=1}^T \prod_{k=1}^K \prod_{j=1}^K a_{jk}^{\delta(z_{tk} z_{t-1j})} \prod_{t=1}^T \prod_{k=1}^K \phi_{x_{tk}}^{\delta(z_{tk})}
 \end{aligned}$$

*End solution]*

2. [2 points] Write down the the likelihood of the data set,  $P(\mathbf{X} | \Theta)$ .

*[Solution:]*

$$P(\mathbf{X} | \theta) = \sum_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z} | \theta)$$

*End solution]*

### 2.b Expectation-Maximization (EM) for Maximum Likelihood Learning

Our goal is to estimate  $\mathbf{A}$  and  $\phi$  that maximizes the likelihood of the data set  $P(\mathbf{X} | \Theta)$ .

1. [3 points] We can use the EM algorithm to compute  $P(\mathbf{X} | \Theta)$ :

- In the E step, we use the current parameters and compute the posterior distribution of the latent variables  $P(\mathbf{Z}|\mathbf{X}, \Theta^{\text{old}})$ .
- In the M step, we find the new parameter values by solving an optimization problem:

$$\Theta^{\text{new}} = \operatorname{argmax}_{\Theta} Q(\Theta, \Theta^{\text{old}}) \quad (1)$$

where

$$Q(\Theta, \Theta^{\text{old}}) = \sum_{\mathbf{Z}} P(\mathbf{Z}|\mathbf{X}, \Theta^{\text{old}}) \ln P(\mathbf{X}, \mathbf{Z}|\Theta) \quad (2)$$

Assume that we can compute  $P(\mathbf{X}, \mathbf{Z}|\Theta)$  in  $O(1)$ . What is the time complexity of  $P(\mathbf{X}|\Theta)$  if we use the above procedure?

*[Solution:]*

The runtime of the E step is  $O(K^2T)$  and the run time of the M step is  $O(K^T)$ . Since  $O(K^T)$  dominates  $O(K^2T)$ , the run time of the algorithm will be  $O(K^T)$ .

*End solution]*

2. [8 points] In class, we learned how to compute:

$$\alpha(z_t) = P(x_1, \dots, x_t, z_t) \quad (3)$$

$$\beta(z_t) = P(x_{t+1}, \dots, x_T | z_t) \quad (4)$$

Show that

$$\xi(z_{t-1}, z_t) = P(z_{t-1}, z_t | \mathbf{X}) \quad (5)$$

$$= \frac{\alpha(z_{t-1}) P(x_t | z_t) P(z_t | z_{t-1}) \beta(z_t)}{p(\mathbf{X})} \quad (6)$$

How can you use one of the  $\alpha$  or  $\beta$  definitions to compute  $P(\mathbf{X})$ ?

*[Solution:]*

$$\begin{aligned} \xi(z_{t-1}, z_t) &= P(z_{t-1}, z_t | \mathbf{X}) \\ &= \frac{P(\mathbf{X} | z_{t-1}, z_t) P(z_{t-1}, z_t)}{P(\mathbf{X})} \text{ (Bayes rule)} \\ &= \frac{P(x_1, \dots, x_{t-1} | z_{t-1}) P(x_t | z_t) P(x_{t+1}, \dots, x_T | z_t) P(z_t | z_{t-1}) P(z_{t-1})}{P(\mathbf{X})} \text{ from d-seperation} \\ &= \frac{\alpha(z_{t-1}) P(x_t | z_t) P(z_t | z_{t-1}) \beta(z_t)}{P(\mathbf{X})} \end{aligned}$$

$$P(\mathbf{X}) = \sum_{z_T} \alpha(z_T)$$

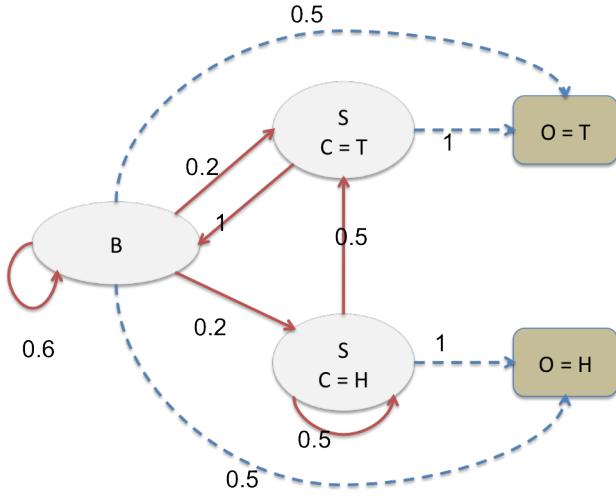
*End solution]*

3. [5 points] Can we say that if any elements of the parameters  $\pi$  or  $\mathbf{A}$  for a hidden Markov model are initially set to 0, then they will remain zero in all subsequent updates of the EM algorithm? If yes, show your steps. If no, explain.

*[Solution:]*

$\pi$  and  $\mathbf{A}$  updated based on the following update rules

$$\begin{aligned} \pi_k &= \frac{\gamma(z_{1k})}{\sum_{j=1}^K \gamma(z_{1j})} \\ A_{jk} &= \frac{\sum_{t=1}^T \xi(z_{t-1,j} z_{tk})}{\sum_{m=1}^K \sum_{t=2}^T \xi(z_{t-1,j} z_{tm})} \end{aligned}$$



We found in the previous question that  $\xi(z_{t-1}, z_t) = \frac{\alpha(z_{t-1})P(x_t|z_t)P(z_t|z_{t-1})\beta(z_t)}{P(\mathbf{X})}$ . If  $A_{jk}$  is 0, then  $\xi(z_{t-1}, z_t)$  is also 0, which makes the subsequent updates in the EM algorithm 0.

*End solution]*

## 2.c A coin game [10 points]

TA's Brendan and Selen play a coin toss game to illustrate how we can use HMMs for sequence analysis problems. Brendan starts tossing first, and they take turns. The game finishes when "THT" appears, and the winner is the one who last flips the coin. At each timestep, they can flip the coin many times, and the stopping rules are as follows:

a. At his turn, each time Brendan flips the coin, he also flips an extra biased coin ( $P(H) = 0.4$ .) He stops only if the extra coin lands H, otherwise he keeps flipping the fair and extra coins. The flips of the extra biased coin are not recorded.

b. At her turn, Selen flips the (fair) coin until T appears (all of her flips are recorded).

You are given a sequence of recorded coin flips, you would like to infer the winner and the flips of each player.

1. [5 points] Describe an HMM to model this game.
2. [5 points] How would you use this HMM model to infer the (most probable) winner and the (most probable) flips of each player?

*[Solution:*

Please see the Figure below.

*End solution]*

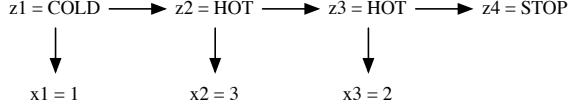
## 3 HMM Programming [Brendan]

Anthropologists in the future are trying to study global warming in our present day, but have lost all temperature records. However, they have my diary of how many ice cream cones I've eaten every day, and want to reconstruct the temperatures from that.<sup>1</sup> (Note that, compared to Problem 2, we now include an explicit STOP state to make the stochastic process well-defined.)

We model this as a Hidden Markov Model, with three latent states {COLD, HOT, STOP} and three discrete observation types {ONEIC, TWOIC, THREEIC}. (I only eat 1, 2, or 3 cones.) Cold days tend to follow cold days, and hot days tend to follow hot days; and we eat more ice cream on hot days.

<sup>1</sup>This example is adapted from Jason Eisner: <http://cs.jhu.edu/~jason/papers/#eisner-2002-tlsp>

Generation proceeds until it reaches a STOP state. Let  $T$  be the number of observations (same thing as the number of state variables, not including the STOP state). For example, one possible generated chain, three timesteps long, consists of latent states  $(z_1 \dots z_4) = (\text{COLD}, \text{HOT}, \text{HOT}, \text{STOP})$  and observations  $(x_1 \dots x_3) = (\text{ONEIC}, \text{THREEIC}, \text{TWOIC})$ . Note the output space is discrete: the numbers don't matter; they could just as well be meaningless symbols.



Given a dataset of  $x_1 \dots x_T$ , and known transition and emission parameters, we are interested in inferring the most likely path  $z_1 \dots z_T$ , which we'll explicitly write out for clarity:

$$\arg \max_{z_1 \dots z_T} P(x_1 \dots x_T, z_1 \dots z_T, z_{T+1} = \text{STOP}) \quad (7)$$

$$= P(z_1) P(x_1 | z_1) \left( \prod_{t=2}^T p(z_t | z_{t-1}) p(x_t | z_t) \right) p(z_{T+1} = \text{STOP} | z_T) \quad (8)$$

The transition and emission parameters  $A$  and  $\phi$  are provided in the starter code on the website. See below for submission details.

1. [5 points] Implement an exhaustive best-path algorithm: enumerate every possible path, compute its log-probability, and choose the highest-scoring one.

[Debugging hint: check the examples  $x = \{1, 2, 1\}$  and  $x = \{3, 2, 3\}$ . The second datapoint is equally likely under either state, but the most likely path states are different due to the HMM chain dependencies: the HMM gives you temporal smoothing, where you consider both past and future for a better estimate of the present.]

Report the most-likely path for this example dataset (*smallX* in the starter code):

$$\vec{x} = [1, 1, 3, 1, 2, 3, 3]$$

*[Solution:*

Actually, there are three paths tied at equal probability, log-prob  $-11.639$ ,

$$\begin{aligned} & [1, 1, 1, 1, 1, 2, 2] \quad (A) \\ & [1, 1, 1, 1, 2, 2, 2] \quad (B) \\ & [1, 1, 2, 2, 2, 2, 2] \quad (C) \end{aligned}$$

This is because of the symmetries in the transition and emission distributions. It's simple why (A) and (B) are tied. They only disagree on the 5th timestep, which is a two-icecream-day; two-icecream days are equally likely under either hot or cold days. While the model does care about the transition from cold to hot, it doesn't have an opinion whether it happened on the 5th or 6th timestep.

(C) is more complex. The third and fourth timesteps have highly differing amounts of ice cream (1 then 3 scoops), which by the emission distribution alone would imply a cold then hot day. But transition stickiness disfavors this sort of flip-flopping, so it wants to pick a run. Thinking they're both cold isn't a great explanation because it's bad for the three-icecream day, but thinking they're both hot isn't good either for the one-icecream day. In fact, they're both equally bad because of the symmetries in the emissions matrix; thus (B) and (C) are tied.

*End solution]*

2. [2 points] Find and report a dataset for which the most-likely path is all HOT's, that differs from *smallX* to the minimal extent possible.

[*Solution*:

There are many possibilities, but here is one answer that differs by *smallX* on only one day: flips the first day to 3.

$$\vec{x} = [3, 1, 3, 1, 2, 3, 3], \log p(\vec{x}, \vec{y}) = -12.8921$$

This illustrates the joint smoothing inferences done by the HMM. Contrast to the (A-C) solution, which has a string of colds then transitions in the middle to hot. Using a '3' on the first day flips all the initial cold states to hot. Since the HMM likes to be sticky, sufficiently changing the evidence on just one day persuades the model that it must have been hot that day, and that it's better to use the stickiness assumption (that temperature shifts are unlikely) to explain all the middle days as hot, rather than assuming hot days in the middle bookended by cold ones.

[*End solution*]

3. [5 points] Report the log joint probability (natural logarithm, please!) that these two data-path pairs attain; i.e. for each, where  $\hat{z}$  is the max-likely path, report:

$$\log p(x_1 \dots x_T, \hat{z}_1 \dots \hat{z}_T, z_{T+1} = \text{STOP})$$

4. [20 points] Implement the Viterbi algorithm to compute the best-path. Test it on small examples; it should always agree with the exhaustive solution. (Corner case: for certain inputs, there may be multiple paths that tie for the highest probability; in that case, they may give different solutions, but the solutions should both have the same probability.)

[Hint: *simdata.m* will simulate new examples for more thorough testing, if you like. You will not of course always recover the correct path, but should be reasonably close, especially if *A* is sharper. In any case, both Viterbi and the exhaustive algorithm should agree. We will test your implementations with similar simulated data.]

Report the Viterbi algorithm's solution to the the dataset *bigX*. Is it feasible to run the exhaustive algorithm on this dataset?

[*Solution*:

No, it is not feasible: exhaustive takes exponential time in the sequence length! By contrast, Viterbi is linear.

Exhaustive runtime:  $O(K^N)$

Viterbi runtime:  $O(K^2 N)$

[*End solution*]

### 3.a Submitting your code

Write your solution in either Matlab or Python, and fill out the stubs given (we provide starter code for both languages). Submit both hardcopy and electronically:

- Print out your implementation. **Your code must be 2 or fewer pages long, in 10-point font.** It should be fewer than 100 lines, and certainly less than 200.
- Copy your code to Andrew AFS, e.g. by using *scp* or an SFTP program to *unix.andrew.cmu.edu*, to the directory:

`/afs/andrew.cmu.edu/usr10/brendano/10601/ps5_submit/YOURANDREWID/`

Only copy the files you need: if you write in Matlab, don't copy the Python files, and vice versa. Please copy all the individual files so that it is directly runnable.

Try copying a file into the directory before the deadline to make sure everything is working. You should be able to delete the file as well.

Make sure your implementations have filled out the three given functions *logjointprob*, *exhaustive\_bestpath*, and *viterbi\_bestpath*. We will use automated scripts to call those functions.

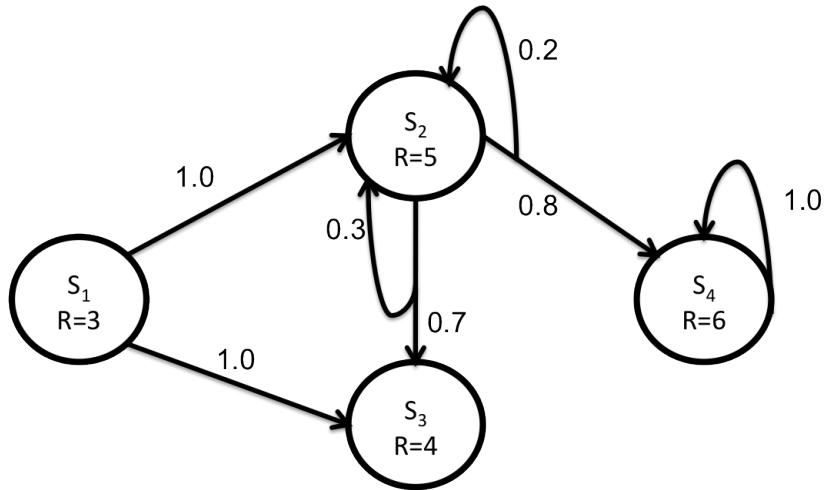
Your code must be runnable on *unix.andrew.cmu.edu*, and is not allowed to use any external libraries (so we can run it reliably).

## 4 Markov Decision Processes [20 points] [Selen]

1. [10 points] A standard (first-order) MDP is described by a set of states  $S$ , a set of actions  $A$ , a transition function  $T$ , and a reward function  $R$  where  $T(s; a; s')$  gives the probability of transitioning to  $s'$  after taking action  $a$  in state  $s$ , and  $R(s)$  gives the immediate reward of being in state  $s$ . In a  $k$ -order MDP, probability of transitioning into a state  $s'$  given that an action  $a$  was taken in state  $s$  depends on the previous  $k - 1$  states. Formally, the transition function  $T$  is described as  $T(s_{k-1}, \dots, s_1, s, a, s') = P(s', a, s, s_1, \dots, s_{k-1})$  where  $P(s', a, s, s_1, \dots, s_{k-1})$  is the probability of transitioning to state  $s'$  given that action  $a$  was taken in state  $s$ , and the previous  $k - 1$  states are  $(s_{k-1}, \dots, s_1)$ .

Given a  $k$ -order MDP  $M = (S; A; T; R)$  describe how to construct a standard (first-order) MDP  $M' = (S'; A'; T'; R')$  that is equivalent to  $M$ , meaning that a solution to  $M'$  can be easily converted into a solution to  $M$ . Describe  $S'$ ,  $A'$ ,  $T'$ ,  $R'$  and give a brief justification for your construction.

2. [10 points] Consider the MDP given in the figure below.  $R$  denotes rewards, and the numbers next to arrows denote probabilities of outcomes. The discount factor is  $\gamma = 0.8$ .



[Solution:

A  $k$ -order MDP  $M$  depends on the current state  $s$ , and the previous  $k-1$  states. Denoting the current and previous states as a  $k$ -tuple, i.e.  $(s, s_1, s_2, \dots, s_{k-1})$  we can construct  $S^k$  states in the new MDP  $M'$ , each state assigned to a  $k$ -tuple. Therefore, each state in  $M'$  corresponds to a state and its history in  $M$ . Note that  $|S'| = |S|^k$ , the number of states in  $M'$  is exponential in  $k$ .

Actions are the same in both MDPs, i.e.  $A' = A$ . This is given in the description.

Reward function depends on the current state only, so  $R'((s, s_1, s_2, \dots, s_k)) = R(s)$ . Note the tuple notation in  $R'$ .

Transitions in  $M'$  is defined as follows:

$$T'((s_{k-1}, \dots, s_1, s), a, s^*) = P(s', a, s, s_1, \dots, s_{k-1}) \quad \text{if } s^* = (s', s, s_1, s_2, \dots, s_{k-2}) \\ = 0$$

This enforces that the history is maintained correctly after each state transitions: a transition to a state that does not update the history correctly has 0 probability. The correct transition to the new state  $s'$  has the same probability given by  $M$ , and this transition involves shifting the history in the current state  $s$  by one step.

*End solution]*

**a. [5 points]** Write down the numerical value of  $J(S_2)$  after the first and second iterations of Value Iteration (in other words compute  $J^1(S_2)$  and  $J^2(S_2)$ ).

Initial value functions are  $J^0(S_1) = 0, J^0(S_2) = 0, J^0(S_3) = 0, J^0(S_4) = 0$ .

*[Solution:*

$$J^1(S_2) = 5 \text{ immediate reward}$$

$$J^2(S_2) = \max(5 + \gamma(0.7J^1(S_3) + 0.3J^1(S_2)), 5 + \gamma(0.8J^1(S_4) + 0.2J^1(S_2)))$$

$$J^2(S_2) = \max(5 + 0.8(0.7 * 4 + 0.3 * 5), 5 + 0.8(0.8 * 6 + 0.2 * 5))$$

$$J^2(S_2) = \max(8.44, 9.64)$$

$$J^2(S_2) = 9.64$$

*End solution]*

**b. [5 points]** Write down  $J^*(S_2)$ , the optimal value of state  $S_2$ .

*[Solution:*

Optimal policy from  $S_2$  will try to transition to  $S_4$ , since it is the state with the largest reward. Lets first compute the optimal value of  $S_4$

$$J^*(S_4) = 6 + 0.8J^*(S_4)$$

$$J^*(S_4) = 30$$

Using  $J^*(S_4)$  we can compute  $J^*(S_2)$ :

$$J^*(S_2) = 5 + 0.8(0.8J^*(S_4) + 0.2J^*(S_2))$$

$$J^*(S_2) = 5 + 0.8(24 + 0.2J^*(S_2))$$

$$J^*(S_2) = \frac{24.2}{0.84}$$

$$J^*(S_2) = 28.8$$

*End solution]*

Solution Sketches  
Midterm Exam  
COSC 6342 *Machine Learning*  
March 20, 2013

Your Name:

Your student id:

Problem 1 [5+?]: Hypothesis Classes

Problem 2 [8]: Losses and Risks

Problem 3 [11]: Model Generation

Problem 4 [8]: Parametric Model Generation/Mahalanobis

Problem 5 [8]: EM (and K-means)

Problem 6 [8]: DBSCAN (and K-means)

Problem 7 [4]: Non-Parametric Prediction Approaches

Problem 8 [8]: General Questions

$\Sigma$  [60]:

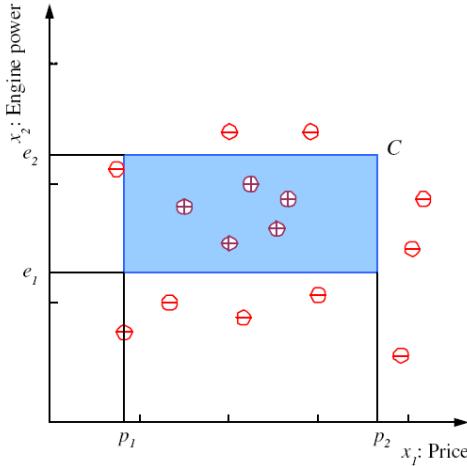
**Grade:**



The exam is “open books and notes” and you have 75 minutes to complete the exam. The exam will count approx. 28% towards the course grade.

## 1) Hypothesis Classes [5]

Assume the hypothesis class for family car problem, discussed in Chapter 2 of the textbook, is a triangle. What are the parameters of the hypothesis class? Describe an approach that calculates the parameters of the triangle from a set of examples—more sophisticated approaches will receive extra credit up to 3 points [5+[3]]



**Parameters: 3 points (6 numbers in 2D) as a triangle is defined by 3 points.  
No algorithm given!**

## 2) Losses and Risks [8]

Determine the optimal decision making strategy with reject option for the following cancer diagnosis problem [6]!

C1=has cancer

C2=has not cancer

$\lambda_{12}=0.2, \lambda_{21}=1$  (as “always” we assume:  $\lambda_{11}=0, \lambda_{22}=0$ )

$\lambda_{\text{reject},2}=0.1$

$\lambda_{\text{reject},1}=0.5$

Inputs:  $P(C1|x), P(C2|x)$

Decision Making Strategy: ...

*The risks associated with the 3 options are as follows:*

$R(a1|x) = 0.2P(C2|x)$

$R(a2|x) = P(C1|x)$

$R(\text{reject}|x) = 0.5P(C1|x)+0.1P(C2|x)$

Equating all 3 pairs of risks<sup>1</sup>, we find out that all three risks are equal if  $P(C1|x)=1/6$ . After analyzing which risk is higher if  $P(C1|x)$  is greater/smaller 1/6 we obtain the following decision rule:

---

<sup>1</sup> For example, setting  $P(C1|x)=0.5P(C1|x)+0.1(1-P(C1|x))$  we obtain  $0.6P(C1|x)=0.1$  therefore  $P(C1|x)=1/6$ ; equating the other two pairs of risk functions leads to the same result!

If  $P(C1|x) > 1/6$ , choose class 1  
If  $P(C1|x) < 1/6$ , choose class 2  
If  $P(C1|x) = 1/6$ , choose class1 or class2 or reject<sup>2</sup>.

Summarize when the decision  $C1 = \text{"Has Cancer"}$  will be taken in your strategy. [2]  
Hence tell the patient she/he has cancer if  $P(C1|x) > 1/6$

### 3) Model Generation [11]

a) Assume you have a model with a high bias and a low variance. What are the characteristics of such a model? [2]

One answer is suggesting underfitting!

Another answer: the model is simple (therefore high bias, as the model is too simple to obtain a good match with the distribution in the dataset) and can be learnt (due its simplicity easily) just using a few training example and is not very sensitive to noise (therefore low variance)

b) Assume you have a small dataset from which a model has to be generated. Would you prefer to learn a complex model or a simple model in this case? Give reasons for your answer! [3]

The simple model should be preferred. Reason: it will not be possible to learn the good model due to the small number of examples—particularly the model's variance will be high leading to a high generalization error.

c) What is overfitting? Limit your answer to 2-3 sentences! [3]

Definition:

+ Let  $D$  is the training set,  $D'$  is the testing set. We say a hypothesis  $h$  is overfitting in a dataset  $D$  if there is another hypothesis  $h'$  in the hypothesis space where  $h$  has better classification accuracy than  $h'$  on  $D$  but worse classification accuracy than  $h'$  on  $D'$ .

Or:

+ Overfitting is the phenomenon in which: when the number of parameters increases (the model gets more complex), the training error decreases but the testing error increases.

- Low bias and high variance  $\Leftrightarrow$  overfitting.

d) What objective function does maximum likelihood estimation minimize in parametric density estimation? How does the maximum likelihood approach differ from the maximum a posteriori (MAP) approach? [3]

- MLE maximizes the  $\ln P(D|\theta)$  likelihood of model  $(\theta)$  with respect to data  $D$ .

- MAP maximizes the  $\ln P(\theta | D)$ , likelihood of data  $D$  with respect to model  $\theta$ .

Moreover, MAP additionally uses priors.

---

<sup>2</sup> All three risks are the same in this case!

#### 4) Parametric Model Generation / Mahalanobis Distance [8]

a) Assume we have a dataset with 3 attributes with the following covariance matrix:

$$\begin{matrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{matrix}$$

What does this co-variance matrix tell you about the relationship of the three attributes?

[2]

- Three attributes have the same variance.
- Attributes 1 and 2 are independent/there is no linear relationship between the attributes.
- Attributes 3 and 2 are independent/there is no linear relationship between the attributes.
- Attributes 1 and 3 have negative correlation of -1 (attribute1=-λ \*attribute2 )

b) What are the advantages of using Mahalanobis distance over Euclidean distance? [2]

- Malanobis distance normalizes attributes based on variance; this makes all independent attributes equally important, and
- it alleviates problem caused by using different scales, and
- downplays the contribution of correlated attributes in distance computations

c) For what problems does parametric model estimation **not** work well? [4]

Parametric model estimation does not work well in problems which have:

- The dataset does not match well the assumption of the employed parametric model estimation technique.
- Feature space has high number of dimensions.
- If there is no global model and only regional/local models exist.
- Parametric models are kind of simplistic and therefore might not capture the characteristic of the data.

#### 5) EM and K-means [8]

a) EM uses a mixture of  $k$  Gaussian for clustering; what purpose do the  $k$  Gaussian serve? [2]

$k$  Gaussians serve as a model for the  $k$  clusters—one Gaussian per cluster---each Gaussian is used to compute the probability that an object belongs to a particular cluster.

b) What is the task of the E-step of the EM-algorithm? Give a verbal description (and not (just) formulas) how EM accomplishes the task of the E-step! [4]

E-step is Expectation step. Compute the posterior distribution

$$Q(\theta', \theta^{(t-1)}) = E_{T|Z, \theta^{(t-1)}}[l_0(\theta'; T)]$$

The E-step computes for each object  $o$  the probability that it belongs to each of the  $k$  clusters $1, \dots, k$ . [2.5] The step is done by dividing the density of  $P(C_i|o)$  for  $i=1, \dots, k$  by the sum of the densities of  $o$  with respect to the  $k$  Gaussians.

c) Characterize what the E-step of K-means does! [2]

It forms clusters by assigning each object  $o$  in the dataset to the cluster with the closest centroid to  $o$ .

## 6) DBSCAN and K-means [8]

a) What is a core point in DBSCAN? What role do core points play in forming clusters? [3]

A point is a core point if it has more than a specified number of points (MinPts) within Eps

DBSCAN takes an unprocessed core point  $p$  and using this core-point it forms a cluster that contains all core- and border points that are density-reachable from  $p$ ; this process continues until all core-points have been assigned to clusters. In summary, forms clusters by recursively computing points in the radius of core points.

b) Compare DBSCAN and K-means; what are the main differences between the 2 clustering approaches? [5]

- DBSCAN has the potential to find arbitrary shape clusters, whereas kmeans is limited to clusters that take the shape of convex polygons [2]
- DBSCAN has outlier detection, but not kmeans [1]
- K-Means performs an iterative maximization procedure, whereas DBSCAN forms clusters in a single iteration [1]
- K-means results depend on initialization and different clusters are usually obtained for different initializations; DBSCAN is more or less deterministic (the only exception is the assignment of borderpoints that lie in the radius of multiple core points)[1]
- K-means is basically  $O(n)$ , DBSCAN is  $O(n \cdot \log(n))$ / $O(n^{**}2)$  [1]

**At most 5 points, even if all 5 answers are given!**

## 7) Non-parametric Prediction and Traditional Regression [4]

Compare traditional regression with non-parametric prediction approaches, such as regressograms/kernel smoothers? What are the main differences between the two approaches?

1. Traditional regression approaches generate a **single** model[1] that is computed **using all examples** of the training set[0.5].
2. Non-parametric approaches employ **multiple (local) models** [0.5] that are derived from a **subset of the training examples in the neighborhood of the query point**<sup>3</sup>. [1]
3. Non-parametric approaches are lazy, as they create the model on the fly, and not beforehand as the traditional regression approach does. [1]

---

<sup>3</sup> or by using a weighted sampling approach which assigns higher weights to points that are closer to the query point.

## 8) General Questions [8]

- a) What is the goal of PCA? Limit your answer to 3-4 sentences! [3]

**Your answer should mention:**

“PCA seeks for linear transformations to a lower dimensional space, and it tries to capture most of the variance of data...”

- b) Most decision tree learning algorithms, such as C4.5, construct decision trees top-down using a greedy algorithm—why is this approach so popular? [3]

As learning the “optimal” decision trees is NP-hard (very time consuming), it is not realistic to find the optimal decision tree.[1.5] Greedy algorithms, as they reach solutions quickly without any backtracking, make it feasible to create a “decent”, but not optimal decision tree relatively quickly.[1.5]

- c) What objective function do regression trees minimize? [2]

It computes the variance of the values of the dependent variable (output variable) of the objects that are associated with the node of a tree; tests that lead to a lower overall variance are preferred by the regression tree induction algorithm.

Name: \_\_\_\_\_ Andrew ID: \_\_\_\_\_

## Instructions

- Anything on paper is OK in arbitrary shape size, and quantity.
- Electronic devices are not acceptable. This includes iPods, iPads, Android tablets, Blackberries, Nokias, Windows phones, Microsoft Surface, laptops, Chromebooks, MP3 players, digital cameras, Google Glass, Android Wear, or anything else that requires electricity.<sup>1</sup>
- **If we catch you cheating or using any such device, the exam is over for you. Your results will not count and you will have failed this exam automatically. There might be more serious consequences, too. No exceptions. Switch off your phones before the exam.**
- There are more questions in the exam than what you are likely to be able to solve in 90 minutes time. Choose wisely and answer those first that you believe you can solve easily. This is an opportunity.
- None of the questions should require more than 1 page to answer. Its OK to be concise if you address the key points of the derivation (it should be human-readable, though).

## Point Distribution

Problem	Points
1	/10
2	/5
3	/5
4	/10
5	/5
6	/10
7	/5
8	/10
9	/15
10	/10
Total	/85

---

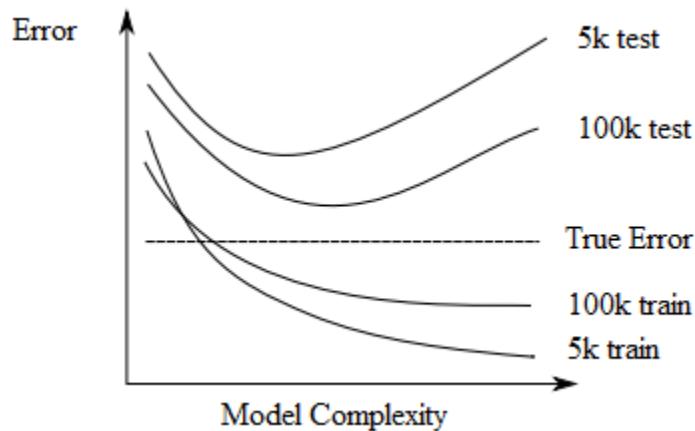
<sup>1</sup>Obvious exceptions for pacemakers and hearing aids.

## 1 Quick Questions (10 points)

### 1.1 Model Selection

Assume two data sets sampled from the same distribution where the number of observations for each data is 5,000 and 100,000 respectively. Randomly construct the train and test set by dividing the data 90:10.

- Draw two curves for training error and test error for each data set with y-axis denoting the error and x-axis denoting the model complexity.
- You should have total of 4 curves: one training error and one test error curve for each dataset.
- Draw all 4 of them in the same diagram.
- Clearly *mark* all your curves.



**Solution:**

- The training error decreases when increasing the model complexity, while the test error decreases first but then increases due to overfitting.
- Given the same model complexity, the model has larger training samples is less likely to overfit than the one with less samples. So the two curves representing 100k samples are nearer the dash line than the other two curves.

### 1.2 $k$ Nearest Neighbor Classification

Suppose we have a large training set. Name a drawback when using a  $k$  Nearest Neighbor during testing.

**Solution:**

k-NN is slow in testing phase, since the time complexity for finding  $k$  nearest neighbors is  $O(knd)$ .  $n$  is number of training data points.  $d$  is number of dimensions.

### 1.3 Density Estimation

- List two drawbacks of bin counting.

**Solution:**

Not continuous, curse of dimensionality, zero density if no training points in the bin, etc.

- What is the advantage of Parzen Windows compared to bin counting?

**Solution:**

Parzen Window Method gives smoother pdf than bin counting.

- Suggest a method to handle low density regions in a Watson-Nadaraya estimator.

**Solution:**

Use average distance from k nearest neighbors. Non-uniform bandwidth for smoother.

## 2 Probabilities (5 points)

### 2.1 Conditional Independence

Construct an example of three random variables  $X, Y$  and  $Z$ , such that  $X \perp Y$  but  $X \not\perp Y|Z$ .

**Solution:**

Let  $X$  and  $Y$  be independent Bernoulli(0.5) random variables. Now define  $Z = X \oplus Y$ . Then, we can see that  $X \perp Y$  but  $X \not\perp Y|Z$ .

### 2.2 Tough Machine Class

The *Tough Machine Learning Course* 10-801 is attended by students majoring in ML and some students that don't major in ML. In it only 50% of the ML students and 30% of the non-ML students pass the midterm exam. Unfortunately 60% of the entire class are non-ML students. What is the percentage of ML students among those that actually pass the exam.

**Solution:**

Let  $S$  denote that a person pass the midterm. Let  $M$  denote that a person is a ML major, and  $N$  denotes the otherwise.

$$P(M) = 0.4, P(N) = 0.6, P(S|M) = 0.5, P(S|N) = 0.3$$

$$P(M|S) = \frac{P(S|M)P(M)}{P(S)} = \frac{P(S|M)P(M)}{P(S|M)P(M) + P(S|N)P(N)}$$

$$= \frac{0.5 * 0.4}{0.5 * 0.4 + 0.3 * 0.6} = \frac{10}{19}$$

### 2.3 Gaussians for free

Denote by  $X_i$  random variables drawn from the uniform distribution  $U[0, 1]$ . Design a random variable

$$Z_n := f(X_1, \dots, X_n)$$

such that  $Z_n$  converges to the normal distribution for  $n \rightarrow \infty$ .

**Solution:**

Many possibilities. One can be  $f(X_1, \dots, X_n) = \frac{1}{n} \sum_{i=1}^n X_i$ .

### 3 Naive Bayes Classifier (5 points)

Annabelle Antique is a collector of old paintings. She is sick of getting e-mails offering her fake artwork and wants to train her own Naive Bayes classifier such that she doesn't have to read all the spam any longer. One of the words she knows that corresponds to a fake is the occurrence of the word *replica*. The Naive Bayes classifier doesn't know this yet. All it can do is compute probabilities. Your job is to help it by generating suitable messages:

#### 3.1 Positive and Negative Examples

Generate two messages corresponding to ham and spam respectively, which will lead to the classifier correctly recognizing *replica* as spam.

**Solution:**

Ham 1: Van Gogh's Starry night in canvas  
Ham 2: The original copy of Da Vinci's Vitruvian Man  
Spam 1: Van Gogh's Starry night oil painting replica  
Spam 2: Da Vinci's Vitruvian Man adapted to oil painting

#### 3.2 Misclassification

Generate a message that would be incorrectly classified as ham based on the four messages generated above. Explain why.

**Solution:**

An exquisite copy of Van Gogh's Starry night

#### 3.3 Breaking Naive Bayes

Generate a message that would lead to an undefined probability estimate. Explain why. Suggest how to fix the Naive Bayes classifier.

## 4 Perceptron Algorithm (10 points)

Assume that you are given observations  $(x, y) \in \mathbb{R}^2 \times \{\pm 1\}$  in the following order:

Instance	1	2	3	4	5	6	7	8
Label $y$	+1	-1	+1	-1	+1	-1	+1	+1
Data $(x_1, x_2)$	(10,10)	(0,0)	(8,4)	(3,3)	(4,8)	(0.5,0.5)	(4,3)	(2,5)

Show the action of the perceptron algorithm for the above sequence of observations. We start with an initial set of weights  $w = (1, 1)$  and bias  $b = 0$ .

Solution:

Let us define  $g(y^{(i)}, w, x^{(i)}, b) = y^{(i)}(\langle w, x^{(i)} \rangle + b)$  and simplify notation by  $g_i = g(y^{(i)}, w, x^{(i)}, b)$ . Rewriting the perceptron algorithm, there is update  $w^{(j)} \leftarrow w^{(j-1)} + y^{(i)}x^{(i)}$ ,  $b^{(j)} \leftarrow b^{(j-1)} + y^{(i)}$  if  $g_i \leq 0$  and no update otherwise. Following this algorithm instance by instance with the starting point  $w^{(0)} = (1, 1)$ ,  $b^{(0)} = 0$ ,

1.  $g_1 = +1 \cdot (10 + 10) > 0 \rightarrow$  no update:

$$\begin{aligned} w^{(1)} &= (1, 1) \\ b^{(1)} &= 0. \end{aligned}$$

2.  $g_2 = -1 \cdot (0 + 0) \leq 0 \rightarrow$  update:

$$\begin{aligned} w^{(2)} &= (1, 1) + (0, 0) = (1, 1) \\ b^{(2)} &= 0 - 1 = -1. \end{aligned}$$

3.  $g_3 = +1 \cdot (12 - 1) > 0 \rightarrow$  no update:

$$\begin{aligned} w^{(3)} &= (1, 1) \\ b^{(3)} &= -1. \end{aligned}$$

4.  $g_4 = -1 \cdot (6 - 1) < 0 \rightarrow$  update:

$$\begin{aligned} w^{(4)} &= (1, 1) - (3, 3) = (-2, -2) \\ b^{(4)} &= -1 - 1 = -2. \end{aligned}$$

5.  $g_5 = +1 \cdot (-8 - 16 - 2) < 0 \rightarrow$  update:

$$\begin{aligned} w^{(5)} &= (-2, -2) + (4, 8) = (2, 6) \\ b^{(5)} &= -2 + 1 = -1. \end{aligned}$$

6.  $g_6 = -1 \cdot (1 + 3 - 1) < 0 \rightarrow$  update:

$$\begin{aligned} w^{(6)} &= (2, 6) - (0.5, 0.5) = (1.5, 5.5) \\ b^{(6)} &= -1 - 1 = -2. \end{aligned}$$

7.  $g_7 = +1 \cdot (6 + 16.5 - 2) > 0 \rightarrow$  no update:

$$\begin{aligned} w^{(7)} &= (1.5, 5.5) \\ b^{(7)} &= -2. \end{aligned}$$

8.  $g_8 = +1 \cdot (3 + 27.5 - 2) > 0 \rightarrow$  no update:

$$\begin{aligned} w^{(8)} &= (1.5, 5.5) \\ b^{(8)} &= -2. \end{aligned}$$

## 5 Representer Theorem (5 points)

Denote by  $\ell : \mathbb{R} \rightarrow \mathbb{R}$  a nonnegative convex differentiable function, i.e. a loss function. Moreover, denote by  $\|\cdot\|_2$  the Euclidean norm. Finally, let  $y_i \in \mathbb{R}$  be scalars and  $x_i, w \in \mathbb{R}^d$  be vectors (i.e.  $d \in \mathbb{N}$ ). Prove that the solution of the optimization problem

$$\underset{w}{\text{minimize}} \sum_{i=1}^n \ell(y_i - \langle x_i, w \rangle) + \|w\|_2$$

can be written as

$$w^* = \sum_i \alpha_i x_i \text{ for some } \alpha_i \in \mathbb{R}.$$

**Solution:**

You can express  $w$  in terms of linear combination of  $x_i$  and  $v$ , i.e.  $w = \alpha_0 v + \sum_{k=1}^n \alpha_k x_k$ , where  $v \in \text{null}\{x_1, \dots, x_n\}$  (in other expression  $v \perp \text{span}\{x_1, \dots, x_n\}$ ). Then, plugging in the  $w = \alpha_0 v + \sum_{k=1}^n \alpha_k x_k$  expression in to the objective function,

$$\begin{aligned} \sum_i^n \ell(y_i - \langle x_i, w \rangle) + \|w\|_2 &= \sum_i^n \ell(y_i - \langle x_i, \alpha_0 v + \sum_{k=1}^n \alpha_k x_k \rangle) + \|\alpha_0 v + \sum_{k=1}^n \alpha_k x_k\|_2 \\ &= \sum_i^n \ell(y_i - \sum_{i=1}^n \alpha'_i x_i) + \sqrt{\|\alpha_0 v\|_2^2 + \|\sum_{k=1}^n \alpha_k x_k\|_2^2} \quad \text{using } v \perp \text{span}\{x_1, \dots, x_n\} \\ &\geq \sum_i^n \ell(y_i - \sum_{i=1}^n \alpha'_i x_i) + \|\sum_{k=1}^n \alpha_k x_k\|_2 \end{aligned}$$

We just proved that the objective function is smaller when the weight of  $v$ ,  $w_0$  is 0, and therefore we conclude that  $\hat{w} = \sum_i^n \alpha_i x_i$ .

## 6 Bundle Methods (10 points)

Let us revisit the bundle method problem we discussed in class. In each iterative step, we need to solve the following optimization problem:

$$\underset{w}{\text{minimize}} \max_{i \in \{1, \dots, k\}} \langle a_i, w \rangle + b_i + \frac{1}{2} \|w\|_2^2$$

where  $w, a_i \in \mathbb{R}^d$  and  $b_i \in \mathbb{R}$  for  $1 \leq i \leq k$ .

### 6.1 Constrained Optimization Problem

Rewrite the optimization problem as the quadratic programming problem. Hint: substitute the maximum with an auxiliary slack variable  $\xi$ .

**Solution:**

Given  $A = [a_1, a_2, \dots, a_k]$ ,  $z \in \mathbb{R}$ , and  $\mathbf{1} \in \mathbb{R}^n$ , we can rewrite the primal problem as

$$\begin{aligned} \min_{w, \xi} \quad & \xi + \frac{1}{2} \|w\|_2^2 \\ \text{s.t.} \quad & A^T w + b \leq \xi \mathbf{1} \end{aligned} \tag{1}$$

*Caution* In order to have the same minimization problem, you have to minimize with  $\xi$  as well on the final expression. -1 if this point is missed. In order to remove  $\max_i$ , you need  $b_i$  incorporated with the constraint related to the  $\xi$ . -1 if this point is missed and hence still has the term  $b_i$  in the final expression.

### 6.2 Dual Problem

Derive the dual optimization problem.

**Solution:**

Writing the Lagrangian function with slack variable  $u \geq 0$ ,  $u \in \mathbb{R}^n$ .

$$L(w, \xi, u) = \xi + \frac{1}{2} \|w\|_2^2 + u^T (A^T w + b - \xi \mathbf{1}),$$

the dual problem of (6.2) is defined as

$$\max_{u \geq 0} \min_{w, \xi} L(w, \xi, u) = \max_{u \geq 0} \min_{w, \xi} \xi + \frac{1}{2} \|w\|_2^2 + u^T (A^T w + b - \xi \mathbf{1}).$$

Let's solve over  $\min_{w, \xi}$ , first minimizing over  $w$ , we obtain the following

$$\frac{\partial L(w, \xi, u)}{\partial w} = 0 \iff w = -Au.$$

Now, plugging  $uA$  into the optimization problem becomes

$$\max_{u \geq 0} \min_{\xi} L(\xi, u) = \max_u \min_{\xi} \xi \mathbf{1} - \frac{1}{2} u^T A^T A u + u^T b - \xi u^T \mathbf{1}.$$

Solving over the  $\min_{\xi}$ ,

$$\frac{\partial L(\xi, u)}{\partial \xi} = 0 \iff u^T \mathbf{1} = 1 \Leftrightarrow \|u\|_1 = 1 \quad (\because u \geq 0)$$

Then, we can remove the terms related to the  $\xi$  and finally obtain the quadratic programming problem.

$$\begin{aligned} \max_u -\frac{1}{2} u^T A^T A u + u^T b \quad & \text{or rewriting as} \quad -\min_u \frac{1}{2} u^T A^T A u - u^T b \\ \text{s.t. } u \geq 0, \|u\|_1 = 1, \quad & \text{s.t. } u \geq 0, \|u\|_1 = 1. \end{aligned}$$

## 7 Entropy of Exponential Family (5 points)

The entropy of a random variable  $X$  distributed according to a probability density function  $p(\cdot)$  is given by

$$H[X] = - \int p(x) \log p(x) dx$$

What is the entropy if the probability density function belonged to exponential family, i.e.

$$p(x|\theta) = \exp(\langle \phi(x), \theta \rangle - g(\theta))$$

Express your answer in terms of  $\theta$ ,  $g(\theta)$  and  $\nabla g(\theta)$ , i.e. the gradient of  $g(\theta)$ .

**Solution:**

Multiple ways to solve it. One is as follows: Let us first pre-compute  $\nabla g(\theta) = \int p(x|\theta) \phi(x) dx$ . Now evaluating the entropy using the definition:

$$\begin{aligned}
 H[X] &= - \int p(x|\theta) \log p(x|\theta) dx \\
 &= - \int p(x|\theta) (\langle \phi(x), \theta \rangle - g(\theta)) dx \\
 &= - \int p(x|\theta) \langle \phi(x), \theta \rangle dx + \int p(x|\theta) g(\theta) dx \\
 &= - \left\langle \int p(x|\theta) \phi(x) dx, \theta \right\rangle + g(\theta) \int p(x|\theta) dx \\
 &= - \langle \nabla g(\theta), \theta \rangle + g(\theta)
 \end{aligned} \tag{2}$$

## 8 Kernels (10 points)

Let  $k(\cdot, \cdot)$  and  $l(\cdot, \cdot)$  with  $k, l : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$  be Mercer kernels. That is,  $k, l$  are symmetric and any Gram matrix constructed based on them is positive semidefinite.

### 8.1 Linear Combinations

Prove that  $\alpha k(x, x') + l(x, x')$  is a valid kernel for any  $\alpha \geq 0$ .

**Solution:**

One proof strategy is based on explicit feature map construction by concatenation. Specifically, let  $k(x, x') = \langle \phi(x), \phi(x') \rangle$  and  $l(x, x') = \langle \psi(x), \psi(x') \rangle$ . Then the feature map for the linear combination is  $\begin{pmatrix} \sqrt{\alpha}\phi \\ \psi \end{pmatrix}$ , i.e.

$$\alpha k(x, x') + l(x, x') = \langle \begin{pmatrix} \sqrt{\alpha}\phi \\ \psi \end{pmatrix}(x), \begin{pmatrix} \sqrt{\alpha}\phi \\ \psi \end{pmatrix}(x') \rangle$$

### 8.2 Upper Bound

Prove the inequality  $k^2(x, x') \leq k(x, x)k(x', x')$ .

**Solution:**

The proof follows from Cauchy-Schwartz inequality in the feature space. Specifically, let  $k(x, x') = \langle \phi(x), \phi(x') \rangle$ . Then  $k(x, x) = \|\phi(x)\|^2$  and  $k(x', x') = \|\phi(x')\|^2$ . By Cauchy-Schwartz, we have  $\|\phi(x)\|\|\phi(x')\| \geq \langle \phi(x), \phi(x') \rangle$  from which the result follows immediately.

### 8.3 Kernel Product

Prove that  $k(x, x')l(x, x')$  is a kernel. Hint: design an explicit feature map for  $k(x, x')l(x, x')$ .

**Solution:**

One proof strategy is to show that elementwise product of any two positive semidefinite matrices is always a positive semidefinite matrix.

Another proof strategy is based on explicit feature map construction as kronecker product. Specifically, let  $k(x, x') = \langle \phi(x), \phi(x') \rangle$  and  $l(x, x') = \langle \psi(x), \psi(x') \rangle$ . Then the feature map for the kernel product is  $\phi \otimes \psi$ , i.e.  $k(x, x')l(x, x') = \langle (\phi \otimes \psi)(x), (\phi \otimes \psi)(x') \rangle$

## 9 Load Balancing (15 points)

In general, load balancing is a problem to distribute tasks among multiple resources. This has useful application across computer science and in particular large scale distributed machine learning algorithms.

Assume that we have  $M$  machines and  $N$  independent tasks that require the same work to execute. Ideally we want to assign the tasks to machines so that every machine gets approximately the same amount of work. For this we use the following load balancing algorithm

$$m(t) := \operatorname{argmin}_{m' \in \{1, \dots, M\}} h(t, m').$$

Here  $t \in \{1, \dots, N\}$  is the task and  $h$  is an ideal hash function. That is, for all practical purposes consider the value associated with  $(t, m)$  as a random variable, drawn independently from any other pair  $(t', m')$  over all integers  $\{1, 2^{64} - 1\}$  (you do not need to worry about collisions).

### 9.1 Expected Load

Show that the *expected fraction* of work per machine is  $1/M$  regardless of the total number of tasks.

**Solution:**

Just need to identify uniform distribution.

### 9.2 Maximum Load for a Machine

Bound the probability that the load for a machine, e.g. machine #42, will exceed  $1/M$  by  $\epsilon$ . Hint: use Hoeffding's theorem.

**Solution:**

Let  $X_m$  denote the load for machine  $m$ . Applying this Hoeffding's inequality to the problem, we have:

$$\Pr \left( X_m > \frac{1}{M} + \epsilon \right) \leq \exp(-2N\epsilon^2) \quad (3)$$

### 9.3 Worst Case for the Cluster

Give a lower bound on the probability that *none of the machines* in the cluster will need to perform more than a fraction of  $1/M + \epsilon$  work. Hint: why can you treat the machines as if they were independent.

**Solution:**

$$\begin{aligned} \Pr \left( \forall m : X_m < \frac{1}{M} + \epsilon \right) &= 1 - \Pr \left( \exists m : X_m > \frac{1}{M} + \epsilon \right) \\ \text{By union bound} &\geq 1 - \sum_m \Pr \left( X_m > \frac{1}{M} + \epsilon \right) \\ &\geq 1 - M \exp(-2N\epsilon^2) \end{aligned} \tag{4}$$

Compute a bound on the number of independent work packages  $N$  in terms of  $\epsilon$  and the confidence  $1 - \delta$ .

**Solution:**

Being conservative, we need to have:

$$\Pr \left( \forall m : X_m < \frac{1}{M} + \epsilon \right) \geq 1 - \delta \tag{5}$$

But we can only compute the lower bound for the probability. So we would like to have the lower bounder to be higher than the minimum confidence required, i.e.

$$\begin{aligned} 1 - M \exp(-2N\epsilon^2) &\geq 1 - \delta \\ M \exp(-2N\epsilon^2) &\leq \delta \\ N &\geq \frac{1}{2\epsilon^2} \log \frac{M}{\delta} \end{aligned} \tag{6}$$

### 9.4 Not enough Tasks

Now assume that  $M = N$ . What is the probability that a particular machine is empty? What is the expected fraction of machines sitting idle for  $M = N \rightarrow \infty$ .

**Solution:**

Probability a particular machine being empty =  $(1 - \frac{1}{N})^N$

As  $N \rightarrow \infty$ , we have  $(1 - \frac{1}{N})^N \rightarrow \frac{1}{e}$

Therefore approximately  $\frac{N}{e}$  computers are sitting idle.

## 10 Shoe Distribution (10 points)

Assume that each student in class has on average 3 pairs, i.e. 6 shoes. Moreover, assume that the standard deviation in the number of shoes is 1.5. Consider the event that a randomly selected student has at least 9 shoes.

### 10.1 Markov's Inequality [2 pts]

Use Markov's inequality to compute an upper bound on the probability of this event.

**Solution:**

$$P(X \geq 9) \leq \frac{E[X]}{9} = \frac{2}{3}$$

### 10.2 Chebyshev's Inequality [3 pts]

Use Chebyshev's inequality to compute an upper bound on the probability of this event.

**Solution:**

Similar to the previous one, two versions are acceptable.

$$P(X \geq 9) \leq P(|X - E[X]| \geq 3) \leq \frac{Var[X]}{3^2} = \frac{1.5^2}{3^2} = \frac{1}{4}$$

### 10.3 A tighter fit [2 pts]

Explain how you could tighten the upper bound computed above (hint — shoes come in pairs).

**Solution:**

Using  $P(X \geq 10)$  with Chebyshev's Inequality will get you a tighter bound.

$$P(X \geq 10) \leq P(|X - E[X]| \geq 4) \leq \frac{Var[X]}{4^2} = \frac{1.5^2}{4^2} = \frac{9}{64}$$

*(This page was left blank intentionally)*

*(This page was left blank intentionally)*

# 10702/36702 Statistical Machine Learning, Spring 2008: Midterm Solutions

March 17, 2008

## 1 Regression [25 points] (Robin)

Let  $X_1 \in \mathbb{R}$  and  $X_2 \in \mathbb{R}$  and

$$Y = m(X_1, X_2) + \epsilon \quad (1)$$

where  $\mathbb{E}(\epsilon) = 0$ .

- (a) Consider the class of multiplicative predictors of the form  $m(x_1, x_2) = \beta x_1 x_2$ . Let  $\beta_*$  be the best predictor, that is,  $\beta_*$  minimizes  $\mathbb{E}(Y - \beta X_1 X_2)^2$ . Find an expression for  $\beta_*$ .

★ **SOLUTION:**  $R = E(Y - \beta X_1 X_2)^2$   
 $\frac{\partial R}{\partial \beta} = -2E(Y - \beta X_1 X_2)X_1 X_2 = 0$   
 $\Rightarrow \beta_* = \frac{E(Y X_1 X_2)}{E(X_1^2 X_2^2)}$

- (b) Suppose the true regression function is

$$Y = X_1 + X_2 + \epsilon.$$

Also assume that  $\mathbb{E}(X_1) = \mathbb{E}(X_2) = 0$ ,  $\mathbb{E}(X_1^2) = \mathbb{E}(X_2^2) = 1$  and that  $X_1$  and  $X_2$  are independent. Find the predictive risk  $R = \mathbb{E}(Y - \beta_* X_1 X_2)^2$  where  $\beta_*$  was defined in part (a).

★ **SOLUTION:**

$$\begin{aligned} \beta_* &= \frac{E(Y X_1 X_2)}{E(X_1^2)E(X_2^2)} = E(Y X_1 X_2) \\ &= E((X_1 + X_2 + \epsilon)(X_1 X_2)) \\ &= E(X_1^2 X_2 + X_1 X_2^2 + X_1 X_2) \\ &= 0 \\ \text{Hence, } E(Y - \beta X_1 X_2)^2 &= E(Y^2) \\ &= E((X_1 + X_2 + \epsilon)^2) \\ &= E(X_1^2 + X_2^2 + \epsilon^2 + 2X_1 X_2 + 2X_1 \epsilon + 2X_2 \epsilon) \\ &= 2 + E(\epsilon^2) \end{aligned}$$

- (c) We are given  $n$  observations  $(X_1, Y_1), \dots, (X_n, Y_n)$  from (1). Give an estimator  $\hat{\beta}_n$  for  $\beta_*$  and show that it is consistent.

$$\begin{aligned}\star \text{ SOLUTION: } \hat{\beta} &= \frac{\frac{1}{n} \sum Y_i X_{1i} X_{2i}}{\frac{1}{n} \sum X_{1i}^2 X_{2i}^2} \\ \frac{1}{n} \sum Y_i X_{1i} X_{2i} &\xrightarrow{P} E(Y X_1 X_2) \quad \frac{1}{n} \sum X_{1i}^2 X_{2i}^2 \xrightarrow{P} E(X_1^2 X_2^2) \quad \therefore \hat{\beta} \rightarrow \beta\end{aligned}$$

## 2 Bayes and Minimax [25 points] (Jingrui)

Let  $X_1, \dots, X_n \sim f(x; \theta)$  where  $f(x; \theta)$  is a distribution from the family of distributions

$$\mathcal{P} = \{f(x; \theta) : \theta \in \Theta\}.$$

Let the loss function for an estimator  $\hat{\theta}$  be

$$L(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2$$

(a) Define the risk function  $R(\theta, \hat{\theta})$ .

**★ SOLUTION:**

$$R(\theta, \hat{\theta}) = E[L(\theta, \hat{\theta})]$$

(b) Define the minimax estimator.

**★ SOLUTION:**  $\hat{\theta}$  minimizes  $\sup_{\theta} R(\theta, \hat{\theta})$ .

(c) Let  $\pi(\theta)$  denote a prior distribution. Define the Bayes' estimator  $\hat{\theta}_{\pi}$  with respect to  $\pi$ .

**★ SOLUTION:**  $\hat{\theta}_{\pi}$  minimizes  $R_{\pi} = \int R(\theta, \hat{\theta}) \pi(\theta) d\theta$ .

(d) Show that the Bayes estimator is

$$\hat{\theta}_{\pi} = \mathbb{E}(\theta | X_1, \dots, X_n).$$

**★ SOLUTION:**  $R_{\pi} = \int [\int (\theta - \hat{\theta})^2 f(\theta | X_1 = x_1, \dots, X_n = x_n) d\theta] m(x_1, \dots, x_n) dx_1 \dots, dx_n$ . Taking the partial derivative of  $\int (\theta - \hat{\theta})^2 f(\theta | x_1, \dots, x_n) d\theta$  with respect to  $\hat{\theta}$ , we have

$$\frac{\partial}{\partial \hat{\theta}} \int (\theta - \hat{\theta})^2 f(\theta | x_1, \dots, x_n) d\theta = 2 \int (\hat{\theta} - \theta) f(\theta | x_1, \dots, x_n) d\theta$$

Setting it to 0, we get  $\hat{\theta} = \int \theta f(\theta | x_1, \dots, x_n) d\theta = \mathbb{E}(\theta | X_1, \dots, X_n)$ .

(e) Suppose that  $R(\theta, \hat{\theta}_{\pi}) = c$  for some constant  $c$ . Show that  $\hat{\theta}_{\pi}$  is minimax.

**★ SOLUTION:** Let  $\tilde{\theta}$  be any other estimator, then

$$\sup_{\theta} R(\theta, \tilde{\theta}) \geq \int R(\theta, \tilde{\theta}) \pi(\theta) d\theta \geq \int R(\theta, \hat{\theta}_{\pi}) \pi(\theta) d\theta = c = \sup_{\theta} R(\theta, \hat{\theta}_{\pi})$$

Therefore,  $\hat{\theta}_{\pi}$  is minimax.

### 3 Model Selection [25 points] (Robin)

Suppose we have the following data:  $(X_1, Y_1), \dots, (X_n, Y_n)$  where  $Y_i \in \mathbb{R}$  and  $X_i \in \mathbb{R}^p$ . Assume that  $p < n$ . Also assume that

$$Y_i = X_i^T \beta + \epsilon_i$$

where  $\epsilon_i$  has mean 0. Let  $\mathbb{X}$  be the  $n \times p$  design matrix, that is,  $\mathbb{X}(i, j) = X_{ij}$ . Suppose that  $\mathbb{X}^T \mathbb{X} = I$  where  $I$  is the  $p \times p$  identity matrix. (We say that the design matrix is orthogonal.)

(a) Recall that the ridge regression estimator is

$$\hat{\beta} = (\mathbb{X}^T \mathbb{X} + \lambda I)^{-1} \mathbb{X}^T Y$$

where  $Y = (Y_1, \dots, Y_n)^T$ . Find the predictive risk of  $\hat{m}(x) = x^T \hat{\beta}$ . Hint: first find the mean and variance of  $\hat{\beta}$ .

$$\begin{aligned} \star \text{ SOLUTION: } \hat{\beta} &= (X^T X + \lambda I)^{-1} X^T Y = (I + \lambda I)^{-1} X^T Y = \frac{1}{1+\lambda} X^T Y = \frac{1}{1+\lambda} X^T [X \beta + \epsilon] = \frac{\beta}{1+\lambda} + \frac{1}{1+\lambda} X^T \epsilon \\ \bar{\beta} &= E(\hat{\beta}) = \frac{\beta}{1+\lambda} \quad V(\hat{\beta}|X) = \frac{\sigma^2}{(1+\lambda)^2} X^T X = \frac{\sigma^2}{(1+\lambda)^2} I \\ \text{Also, } \beta - \bar{\beta} &= \frac{\lambda}{1+\lambda} \beta \\ S = V(\hat{\beta}|X) &= \left(\frac{\sigma}{(1+\lambda)}\right)^2 I \quad R = E(Y - X^T \hat{\beta})^2 \end{aligned}$$

$$\begin{aligned} E(Y - X^T \hat{\beta})^2 &= E(X\beta + \epsilon - X^T \hat{\beta})^2 \\ &= E[(\hat{\beta} - \beta)^T X X^T (\hat{\beta} - \beta)] + \sigma^2 \\ &= E[(\hat{\beta} - \bar{\beta})^T X X^T (\hat{\beta} - \bar{\beta})] + 2E[(\hat{\beta} - \bar{\beta})^T X X^T (\bar{\beta} - \beta)] + E[(\bar{\beta} - \beta)^T X X^T (\bar{\beta} - \beta)] + \sigma^2 \\ &= \sum_{j=1}^p E(X_j^2) \left[ \left(\frac{\lambda}{1+\lambda}\right)^2 \beta_j^2 + \left(\frac{\sigma}{1+\lambda}\right)^2 \right] + 0 + \left(\frac{\lambda}{1+\lambda}\right)^2 \beta^T E(X X^T) \beta + \sigma^2 \end{aligned}$$

(b) Still assuming that the design matrix is orthogonal, show that it is possible to find the lasso estimator without using iterative algorithms or quadratic programming. Hint: consider the transformed response  $Z = \mathbb{X}^T Y$ .

$$\star \text{ SOLUTION: } Z = X^T Y = X^T (X \beta + \epsilon) = \beta + X^T \epsilon$$

$$Z \sim N(\beta, \sigma^2)$$

Apply soft thresholding to  $Z$

### 4 Convex Duality [25 points] (Jingrui)

Let  $X_i \sim \text{Bernoulli}(\theta)$  be independent, with observations  $\{X_1, X_2, X_3\} = \{0, 1, 0\}$ . Thus,  $\mathbb{P}(X_i = 1) = \theta$  and  $\mathbb{P}(X_i = 0) = 1 - \theta$  where  $0 \leq \theta \leq 1$ . Consider the optimization problem

$$\begin{aligned} \min_{\theta} \quad & f(\theta) \\ \text{such that } \theta \geq & 1/2 \end{aligned}$$

where  $f(\theta)$  is the negative log-likelihood.

(a) What is the solution to this problem?

★ SOLUTION: The likelihood is  $L = \theta(1 - \theta)^2$ . Therefore,  $f(\theta) = -\log \theta - 2 \log(1 - \theta)$ , which is a convex function. Let  $\frac{\partial f(\theta)}{\partial \theta} = 0$ , we get  $\hat{\theta} = 1/3$ . However, this solution does not satisfy the constraint. When  $\theta \geq 1/2$ ,  $f(\theta)$  is a decreasing function. Therefore, the solution to this problem is  $\hat{\theta} = 1/2$ .

(b) Write the Lagrangian.

★ SOLUTION:

$$L(\theta, \lambda) = -\log \theta - 2 \log(1 - \theta) + \lambda(\frac{1}{2} - \theta)$$

(c) Derive the dual problem.

★ SOLUTION:  $\frac{\partial L(\theta, \lambda)}{\partial \theta} = -\frac{1}{\theta} + \frac{2}{1-\theta} - \lambda = 0$ . Therefore,  $\lambda\theta^2 + (3 - \lambda)\theta - 1 = 0$ , and  $\theta^* = \frac{\lambda-3+\sqrt{(\lambda-3)^2+4\lambda}}{2\lambda}$ . The dual function:  $l(\lambda) = -\log \theta^* - 2 \log(1 - \theta^*) + \lambda(\frac{1}{2} - \theta^*)$ .

(d) State the KKT conditions.

★ SOLUTION:

$$\begin{aligned} -\frac{1}{\theta^*} + \frac{2}{1-\theta^*} - \lambda^* &= 0 \\ \frac{1}{2} - \theta^* &\leq 0 \\ \lambda^* &\geq 0 \\ \lambda^*(\frac{1}{2} - \theta^*) &= 0 \end{aligned}$$

## 5 Regularization [25 points] (Robin)

Let  $Y$  be the random variable

$$Y = \mu + \epsilon$$

where  $\epsilon \sim N(0, 1)$  and  $\mu \in \mathbb{R}$  is a constant. The elastic net estimator  $\hat{\mu}$  is defined to be the value of  $\mu$  that minimizes

$$M(\mu) = (Y - \mu)^2 + \lambda|\mu| + \alpha\mu^2$$

where  $\lambda, \alpha > 0$ . Find  $\hat{\mu}$ .

★ SOLUTION:  $\frac{\partial M}{\partial \mu} = -2(Y - \mu) + \lambda z + 2\alpha\mu$

$$\text{where } z = \begin{cases} 1 & \text{if } \mu > 0 \\ -1 & \text{if } \mu < 0 \\ \in [-1, 1] & \text{if } \mu = 0 \end{cases}$$

When  $\mu = 0$ ,  $-2Y + \lambda z = 0 \quad Y = \frac{\lambda z}{2} \quad \therefore \hat{\mu} = 0 \quad \text{if } |Y| \leq \frac{\lambda}{2}$

When  $\mu > 0$ ,  $-2(Y - \mu) + \lambda + 2\alpha\mu = 0 \quad \therefore \hat{\mu} = \frac{2Y - \lambda}{2(1 + \alpha)}$

When  $\mu < 0$ ,  $-2(Y - \mu) - \lambda + 2\alpha\mu = 0 \quad \therefore \hat{\mu} = \frac{2Y + \lambda}{2(1 + \alpha)}$

$$\hat{\mu} = \begin{cases} \frac{2Y - \lambda}{2(1 + \alpha)} & Y > \lambda/2 \\ 0 & |Y| \leq \lambda/2 \\ \frac{2Y + \lambda}{2(1 + \alpha)} & Y < -\lambda/2 \end{cases}$$

## 6 Mixture Models [25 points] (Jingrui)

Let  $(Z_1, Y_1), \dots, (Z_n, Y_n)$  be generated as follows:

$$Z_i \sim \text{Bernoulli}(p)$$

$$Y_i \sim \begin{cases} N(0, 1) & \text{if } Z_i = 0 \\ N(5, 1) & \text{if } Z_i = 1 \end{cases}$$

- (a) Assume we do not observe the  $Z_i$ 's. Write the distribution  $f(y)$  of  $Y$  as a mixture.

★ SOLUTION:

$$f(y) = p\phi(y - 5) + (1 - p)\phi(y)$$

where  $\phi(\cdot)$  is the pdf of a standard normal distribution.

- (b) Write down the likelihood function for  $p$ .

★ SOLUTION:

$$L(p) = \prod_{i=1}^n [p\phi(y_i - 5) + (1 - p)\phi(y_i)]$$

- (c) Write down the complete likelihood function for  $p$  (assuming the  $Z_i$ 's are observed).

★ SOLUTION:

$$L(p) = \prod_{i=1}^n [p^{z_i} (\phi(y_i - 5))^{z_i} (1 - p)^{1 - z_i} (\phi(y_i))^{1 - z_i}]$$

- (d) Find a consistent estimator of  $p$  that avoids using EM.

★ SOLUTION:  $\mathbb{E}(Y) = 5p + 0(1 - p) = 5p$ . Let  $\hat{p} = \frac{\bar{Y}}{5}$ .  $\mathbb{E}(\hat{p}) = p$ . According to Law of Large Numbers,  $\hat{p}$  converges to  $\mathbb{E}(\hat{p})$  in probability. Therefore,  $\hat{p}$  is a consistent estimator of  $p$ .

## 7 Classification [25 points] (Robin)

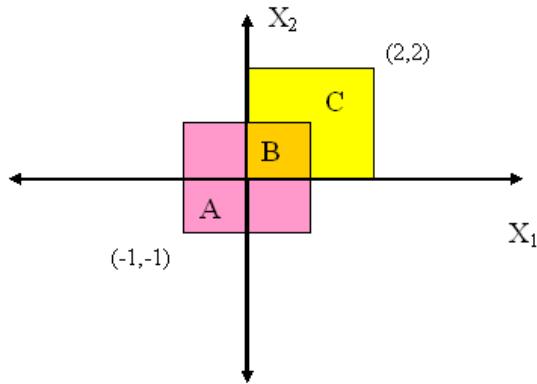
Suppose that  $\mathbb{P}(Y = 1) = \mathbb{P}(Y = 0) = \frac{1}{2}$  and

$$X|Y = 0 \sim \text{Uniform on } S_0$$

$$X|Y = 1 \sim \text{Uniform on } S_1$$

where  $S_0$  is the square in  $\mathbb{R}^2$  with corners  $(1, 1), (1, -1), (-1, 1), (-1, -1)$  and where  $S_1$  is the square in  $\mathbb{R}^2$  with corners  $(0, 0), (2, 0), (2, 2), (0, 2)$ .

- (a) Find an expression for the Bayes classifier and find an expression for the Bayes risk.



★ SOLUTION:

$$A = S_0 - (S_0 \cap S_1)$$

$$B = S_0 \cap S_1$$

$$C = S_1 - (S_0 \cap S_1)$$

$$h_*(x) = \begin{cases} 1 & x \in C \\ 0 & x \in A \\ \text{either} & x \in B \end{cases}$$

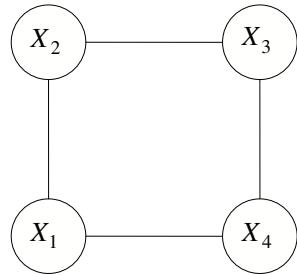
$$\text{Bayes Risk } R = P(Y \neq h_*(x)) = \frac{1}{2}P(B) = \frac{1}{2} \frac{1}{4} = \frac{1}{8}$$

(b) What is the best linear classifier?

Any classifier that preserves A & C. For e.g.,  $X_1 + X_2 = 1$

## 8 Graphical Models [25 points] (Jingrui)

Let  $X = (X_1, X_2, X_3, X_4)$  be a random vector and consider the graph:



(a) List the local Markov properties.

★ SOLUTION:

$$X_1 \perp X_3 | X_2, X_4$$

$$X_2 \perp X_4 | X_1, X_3$$

(b) List the global Markov properties.

★ SOLUTION:

$$\begin{aligned} X_1 &\perp X_3 | X_2, X_4 \\ X_2 &\perp X_4 | X_1, X_3 \end{aligned}$$

(c) Assume that all the variables are binary. Write down a graphical loglinear model for this graph.

★ SOLUTION:

$$\log P = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_{12} x_1 x_2 + \beta_{23} x_2 x_3 + \beta_{34} x_3 x_4 + \beta_{41} x_4 x_1$$

(d) Write down a nongraphical loglinear model for this graph.

★ SOLUTION: Many solutions are OK for this problem. For example,

$$\log P = \beta_0 + \beta_{12} x_1 x_2 + \beta_{23} x_2 x_3 + \beta_{34} x_3 x_4 + \beta_{41} x_4 x_1$$

# 10-601 Machine Learning, Midterm Exam

Instructors: Tom Mitchell, Ziv Bar-Joseph

Monday 22<sup>nd</sup> October, 2012

There are 5 questions, for a total of 100 points.

This exam has 16 pages, make sure you have all pages before you begin.  
This exam is open book, open notes, but *no computers or other electronic devices*.

Good luck!

Name: \_\_\_\_\_

Andrew ID: \_\_\_\_\_

Question	Points	Score
Short Answers	20	
Comparison of ML algorithms	20	
Regression	20	
Bayes Net	20	
Overfitting and PAC Learning	20	
Total:	100	

## Question 1. Short Answers

### True False Questions.

- (a) [1 point] We can get multiple local optimum solutions if we solve a linear regression problem by minimizing the sum of squared errors using gradient descent.

True      False

**Solution:**

False

- (b) [1 point] When a decision tree is grown to full depth, it is more likely to fit the noise in the data.

True      False

**Solution:**

True

- (c) [1 point] When the hypothesis space is richer, over fitting is more likely.

True      False

**Solution:**

True

- (d) [1 point] When the feature space is larger, over fitting is more likely.

True      False

**Solution:**

True

- (e) [1 point] We can use gradient descent to learn a Gaussian Mixture Model.

True      False

**Solution:**

True

### Short Questions.

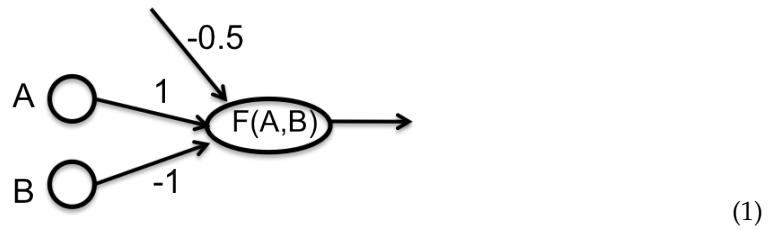
- (f) [3 points] Can you represent the following boolean function with a single logistic threshold unit (i.e., a single unit from a neural network)? If yes, show the weights. If not, explain why not in 1-2 sentences.

A	B	f(A,B)
1	1	0
0	0	0
1	0	1
0	1	0

**Solution:**

Yes, you can represent this function with a single logistic threshold unit, since it is linearly separable. Here is one example.

$$F(A, B) = 1\{A - B - 0.5 > 0\}$$



- (g) [3 points] Suppose we clustered a set of  $N$  data points using two different clustering algorithms: k-means and Gaussian mixtures. In both cases we obtained 5 clusters and in both cases the centers of the clusters are exactly the same. Can 3 points that are assigned to different clusters in the k-means solution be assigned to the same cluster in the Gaussian mixture solution? If no, explain. If so, sketch an example or explain in 1-2 sentences.

**Solution:**

Yes, k-means assigns each data point to a unique cluster based on its distance to the cluster center. Gaussian mixture clustering gives soft (probabilistic) assignment to each data point. Therefore, even if cluster centers are identical in both methods, if Gaussian mixture components have large variances (components are spread around their center), points on the edges between clusters may be given different assignments in the Gaussian mixture solution.

**Circle the correct answer(s).**

- (h) [3 points] As the number of training examples goes to infinity, your model trained on that data will have:
- A. Lower variance
  - B. Higher variance
  - C. Same variance

**Solution:**

Lower variance

- (i) [3 points] As the number of training examples goes to infinity, your model trained on that data will have:
- A. Lower bias
  - B. Higher bias
  - C. Same bias

**Solution:**

Same bias

- (j) [3 points] Suppose you are given an EM algorithm that finds maximum likelihood estimates for a model with latent variables. You are asked to modify the algorithm so that it finds MAP estimates instead. Which step or steps do you need to modify?
- A. Expectation
  - B. Maximization
  - C. No modification necessary
  - D. Both

**Solution:**

Maximization

## Question 2. Comparison of ML algorithms

Assume we have a set of data from patients who have visited UPMC hospital during the year 2011. A set of features (e.g., temperature, height) have been also extracted for each patient. Our goal is to decide whether a new visiting patient has any of diabetes, heart disease, or Alzheimer (a patient can have one or more of these diseases).

- (a) [3 points] We have decided to use a neural network to solve this problem. We have two choices: either to train a *separate* neural network for each of the diseases or to train a single neural network with one output neuron for each disease, but with a shared hidden layer. Which method do you prefer? Justify your answer.

**Solution:**

1- Neural network with a shared hidden layer can capture dependencies between diseases. It can be shown that in some cases, when there is a dependency between the output nodes, having a shared node in the hidden layer can improve the accuracy.  
 2- If there is no dependency between diseases (output neurons), then we would prefer to have a separate neural network for each disease.

- (b) [3 points] Some patient features are expensive to collect (e.g., brain scans) whereas others are not (e.g., temperature). Therefore, we have decided to first ask our classification algorithm to predict whether a patient has a disease, and if the classifier is 80% confident that the patient has a disease, then we will do additional examinations to collect additional patient features. In this case, which classification methods do you recommend: neural networks, decision tree, or naive Bayes? Justify your answer in one or two sentences.

**Solution:**

We expect students to explain how each of these learning techniques can be used to output a confidence value (any of these techniques can be modified to provide a confidence value). In addition, Naive Bayes is preferable to other cases since we can still use it for classification when the value of some of the features are unknown.

We gave partial credits to those who mentioned neural network because of its non-linear decision boundary, or decision tree since it gives us an interpretable answer.

- (c) Assume that we use a logistic regression learning algorithm to train a classifier for each disease. The classifier is trained to obtain MAP estimates for the logistic regression weights  $W$ . Our MAP estimator optimizes the objective

$$W \leftarrow \arg \max_W \ln[P(W) \prod_l P(Y^l | X^l, W)]$$

where  $l$  refers to the  $l$ th training example. We adopt a Gaussian prior with zero mean for the weights  $W = \langle w_1 \dots w_n \rangle$ , making the above objective equivalent to:

$$W \leftarrow \arg \max_W -C \sum_i w_i + \sum_l \ln P(Y^l | X^l, W)$$

Note  $C$  here is a constant, and we re-run our learning algorithm with different values of  $C$ . Please answer each of these true/false questions, and explain/justify your answer in no more than 2 sentences.

- i. [2 points] The average log-probability of the *training data* can never increase as we increase  $C$ .  
 True      False

**Solution:**

True. As we increase  $C$ , we give more weight to constraining the predictor. Thus it makes our predictor less flexible to fit to training data (over constraining the predictor, makes it unable to fit to training data).

- ii. [2 points] If we start with  $C = 0$ , the average log-probability of *test data* will likely decrease as we increase  $C$ .

True      False

**Solution:**

False. As we increase the value of  $C$  (starting from  $C = 0$ ), we avoid our predictor to over fit to training data and thus we expect the accuracy of our predictor to be increased on the test data.

- iii. [2 points] If we start with a very large value of  $C$ , the average log-probability of *test data* can never decrease as we increase  $C$ .

True      False

**Solution:**

False. Similar to the previous parts, if we over constraint the predictor (by choosing very large value of  $C$ ), then it wouldn't be able to fit to training data and thus makes it to perform worst on the test data.

(d) Decision boundary

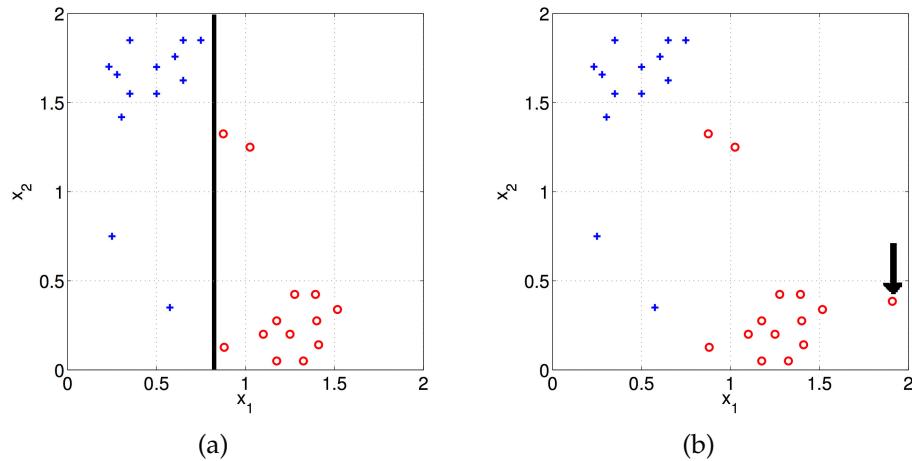


Figure 1: Labeled training set.

- i. [2 points] Figure 1(a) illustrates a subset of our training data when we have only two features:  $X_1$  and  $X_2$ . Draw the decision boundary for the logistic regression that we explained in part (c).

**Solution:**

The decision boundary for logistic regression is linear. One candidate solution which classifies all the data correctly is shown in Figure 1. We will accept other possible solutions since decision boundary depends on the value of  $C$  (it is possible for the trained classifier to miss-classify a few of the training data if we choose a large value of  $C$ ).

- ii. [3 points] Now assume that we add a new data point as it is shown in Figure 1(b). How does it change the decision boundary that you drew in Figure 1(a)? Answer this by drawing both the old and the new boundary.

**Solution:**

We expect the decision boundary to move a little toward the new data point.

- (e) [3 points] Assume that we record information of all the patients who visit UPMC every day. However, for many of these patients we don't know if they have any of the diseases, can we still improve the accuracy of our classifier using these data? If yes, explain how, and if no, justify your answer.

**Solution:**

Yes, by using EM. In the class, we showed how EM can improve the accuracy of our classifier using both labeled and unlabeled data. For more details, please look at [http://www.cs.cmu.edu/~tom/10601\\_fall2012/slides/GrMod3\\_10\\_9\\_2012.pdf](http://www.cs.cmu.edu/~tom/10601_fall2012/slides/GrMod3_10_9_2012.pdf), page 6.

### Question 3. Regression

Consider real-valued variables  $X$  and  $Y$ . The  $Y$  variable is generated, conditional on  $X$ , from the following process:

$$\epsilon \sim N(0, \sigma^2)$$

$$Y = aX + \epsilon$$

where every  $\epsilon$  is an independent variable, called a *noise* term, which is drawn from a Gaussian distribution with mean 0, and standard deviation  $\sigma$ . This is a one-feature linear regression model, where  $a$  is the only weight parameter. The conditional probability of  $Y$  has distribution  $p(Y|X, a) \sim N(aX, \sigma^2)$ , so it can be written as

$$p(Y|X, a) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(Y - aX)^2\right)$$

The following questions are all about this model.

#### MLE estimation

- (a) [3 points] Assume we have a training dataset of  $n$  pairs  $(X_i, Y_i)$  for  $i = 1..n$ , and  $\sigma$  is known.

Which ones of the following equations correctly represent the maximum likelihood problem for estimating  $a$ ? Say yes or no to each one. More than one of them should have the answer "yes."

[Solution: no]  $\arg \max_a \sum_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right)$

[Solution: yes]  $\arg \max_a \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right)$

[Solution: no]  $\arg \max_a \sum_i \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right)$

[Solution: yes]  $\arg \max_a \prod_i \exp\left(-\frac{1}{2\sigma^2}(Y_i - aX_i)^2\right)$

[Solution: no]  $\arg \max_a \frac{1}{2} \sum_i (Y_i - aX_i)^2$

[Solution: yes]  $\arg \min_a \frac{1}{2} \sum_i (Y_i - aX_i)^2$

- (b) [7 points] Derive the maximum likelihood estimate of the parameter  $a$  in terms of the training example  $X_i$ 's and  $Y_i$ 's. We recommend you start with the simplest form of the problem you found above.

**Solution:**

Use  $F(a) = \frac{1}{2} \sum_i (Y_i - aX_i)^2$  and minimize  $F$ . Then

$$0 = \frac{\partial}{\partial a} \left[ \frac{1}{2} \sum_i (Y_i - aX_i)^2 \right] \quad (2)$$

$$= \sum_i (Y_i - aX_i)(-X_i) \quad (3)$$

$$= \sum_i aX_i^2 - X_i Y_i \quad (4)$$

$$a = \frac{\sum_i X_i Y_i}{\sum_i X_i^2} \quad (5)$$

Partial credit: 1 point for writing a correct objective, 1 point for taking the derivative, 1 point for getting the chain rule correct, 1 point for a reasonable attempt at solving for  $a$ . 6 points for correct up to a sign error.

Many people got  $\sum y_i / \sum x_i$  as the answer, by erroneously cancelling  $x_i$  on top and bottom. 4 points for this answer when it is clear this cancelling caused the problem. If they explicitly derived  $\sum x_i y_i / \sum x_i^2$  along the way, 6 points. If it is completely unclear where  $\sum y_i / \sum x_i$  came from, sometimes worth only 3 points (based on the partial credit rules above).

Some people wrote a gradient descent rule. We intended to ask for a closed-form maximum likelihood estimate, not an algorithm to get it. (Yes, it is true that lectures never said there exists a closed-form solution for linear regression MLE. But there is. In fact, there is a closed-form solution even for multiple features, via linear algebra.) But we gave 4 points for getting the rule correct; 3 points for correct with a sign error.

For gradient descent/ascent signs are tricky. If you are using the log-likelihood, thus maximization, you want gradient ascent, and thus add the gradient. If instead you're doing the minimization problem, and using gradient descent, need to subtract the gradient. Either way, it comes out to  $a \leftarrow a + \eta \sum_i (y_i - ax_i)x_i$ . Interpretation:  $\sum_i (y_i - ax_i)x_i$  is the correlation of data against the residual. In the case of positive  $x, y$ , if the data still correlates with the residual, that means predictions are too low, so you want to increase  $a$ .

Here is a lovely book chapter by Tufte (1974) on one-feature linear regression:

<http://www.edwardtufte.com/tufte/dapp/chapter3.html>

## MAP estimation

Let's put a prior on  $a$ . Assume  $a \sim N(0, \lambda^2)$ , so

$$p(a|\lambda) = \frac{1}{\sqrt{2\pi}\lambda} \exp\left(-\frac{1}{2\lambda^2}a^2\right)$$

The posterior probability of  $a$  is

$$p(a | Y_1, \dots, Y_n, X_1, \dots, X_n, \lambda) = \frac{p(Y_1, \dots, Y_n | X_1, \dots, X_n, a)p(a|\lambda)}{\int_{a'} p(Y_1, \dots, Y_n | X_1, \dots, X_n, a')p(a'|\lambda)da'}$$

We can ignore the denominator when doing MAP estimation.

- (c) [3 points] Under the following conditions, how do the prior and conditional likelihood curves change? Do  $a^{MLE}$  and  $a^{MAP}$  become closer together, or further apart?

	$p(a \lambda)$ prior probability: wider, narrower, or same?	$p(Y_1 \dots Y_n   X_1 \dots X_n, a)$ conditional likelihood: wider, narrower, or same?	$ a^{MLE} - a^{MAP} $ increase or decrease?
As $\lambda \rightarrow \infty$	[Solution: wider]	[Solution: same]	[Solution: decrease]
As $\lambda \rightarrow 0$	[Solution: narrower]	[Solution: same]	[Solution: increase]
More data: as $n \rightarrow \infty$ (fixed $\lambda$ )	[Solution: same]	[Solution: narrower]	[Solution: decrease]

- (d) [7 points] Assume  $\sigma = 1$ , and a fixed prior parameter  $\lambda$ . Solve for the MAP estimate of  $a$ ,

$$\arg \max_a [\ln p(Y_1 \dots Y_n | X_1 \dots X_n, a) + \ln p(a|\lambda)]$$

Your solution should be in terms of  $X_i$ 's,  $Y_i$ 's, and  $\lambda$ .

**Solution:**

$$\frac{\partial}{\partial a} [\log p(Y|X, a) + \log p(a|\lambda)] = \frac{\partial \ell}{\partial a} + \frac{\partial \log p(a|\lambda)}{\partial a} \quad (6)$$

To stay sane, let's look at it as maximization, not minimization. (It's easy to get signs wrong by trying to use the squared error minimization form from before.) Since  $\sigma = 1$ , the log-likelihood and its derivative is

$$\ell(a) = \log \left[ \prod_i \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{1}{2\sigma^2} (Y_i - aX_i)^2 \right) \right] \quad (7)$$

$$\ell(a) = -\log Z - \frac{1}{2} \sum_i (Y_i - aX_i)^2 \quad (8)$$

$$\frac{\partial \ell}{\partial a} = - \sum_i (Y_i - aX_i)(-X_i) \quad (9)$$

$$= \sum_i (Y_i - aX_i)X_i \quad (10)$$

$$= \sum_i X_i Y_i - aX_i^2 \quad (11)$$

Next get the partial derivative for the log-prior.

$$\frac{\partial \log p(a)}{\partial a} = \frac{\partial}{\partial a} \left[ -\log(\sqrt{2\pi}\lambda) - \frac{1}{2\lambda^2} a^2 \right] \quad (12)$$

$$= -\frac{a}{\lambda^2} \quad (13)$$

The full partial is the sum of that and the log-likelihood which we did before.

$$0 = \frac{\partial \ell}{\partial a} + \frac{\partial \log p(a)}{\partial a} \quad (14)$$

$$0 = \left( \sum_i X_i Y_i - a X_i^2 \right) - \frac{a}{\lambda^2} \quad (15)$$

$$a = \frac{\sum_i X_i Y_i}{(\sum_i X_i^2) + 1/\lambda^2} \quad (16)$$

Partial credit: 1 point for writing out the log posterior, and/or doing some derivative. 1 point for getting the derivative correct.

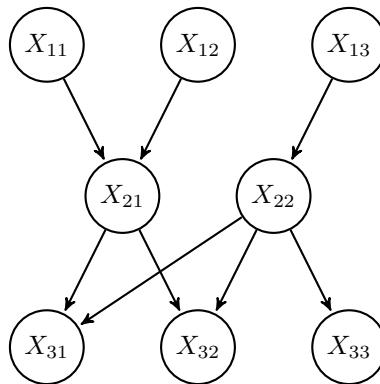
For full solution: deduct a point for a sign error. (There are many potential places for flipping signs). Deduct a point for having  $n/\lambda^2$ : this results from wrapping a sum around the log-prior. (Only the log-likelihood as a  $\sum_i$  around it since it's the probability of drawing each data point. The parameter  $a$  is drawn only once.)

Some people didn't set  $\sigma = 1$  and kept  $\sigma$  to the end. We simply gave credit if substituting  $\sigma = 1$  gave the right answer; a few people may have derived the wrong answer but we didn't carefully check all these cases.

People who did gradient descent rules were graded similarly as before: 4 points if correct, deduct one for sign error.

## Question 4. Bayes Net

Consider a Bayesian network  $B$  with boolean variables.



- (a) [2 points] From the rule we covered in lecture, is there any variable(s) conditionally independent of  $X_{33}$  given  $X_{11}$  and  $X_{12}$ ? If so, list all.

**Solution:**

$X_{21}$

- (b) [2 points] From the rule we covered in lecture, is there any variable(s) conditionally independent of  $X_{33}$  given  $X_{22}$ ? If so, list all.

**Solution:**

Everything but  $X_{22}, X_{33}$ .

- (c) [3 points] Write the joint probability  $P(X_{11}, X_{12}, X_{13}, X_{21}, X_{22}, X_{31}, X_{32}, X_{33})$  factored according to the Bayes net. How many parameters are necessary to define the conditional probability distributions for this Bayesian network?

**Solution:**

$$\begin{aligned}
 & P(X_{11}, X_{12}, X_{13}, X_{21}, X_{22}, X_{31}, X_{32}, X_{33}) \\
 & = P(X_{11})P(X_{12})P(X_{13})P(X_{21}|X_{11}, X_{12})P(X_{22}|X_{13})P(X_{31}|X_{21}X_{22})P(X_{32}|X_{21}X_{22})P(X_{33}|X_{22})
 \end{aligned}$$

9 parameters are necessary.

- (d) [2 points] Write an expression for  $P(X_{13} = 0, X_{22} = 1, X_{33} = 0)$  in terms of the conditional probability distributions given in your answer to part (c). Show your work.

**Solution:**

$$P(X_{13} = 0)P(X_{22} = 1|X_{13} = 0)P(X_{33} = 0|X_{22} = 1)$$

- (e) [3 points] From your answer to (d), can you say  $X_{13}$  and  $X_{33}$  are independent? Why?

**Solution:**

No. Conditional independence doesn't imply marginal independence.

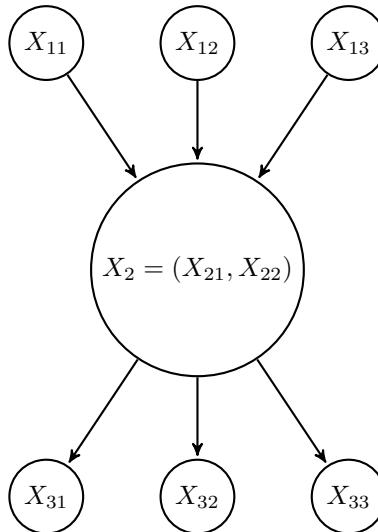
- (f) [3 points] Can you say the same thing when  $X_{22} = 1$ ? In other words, can you say  $X_{13}$  and  $X_{33}$  are independent given  $X_{22} = 1$ ? Why?

**Solution:**

Yes.  $X_{22}$  is the only parent of  $X_{33}$  and  $X_{13}$  is a nondescendant of  $X_{33}$ , so by the rule in the lecture we can say they are independent given  $X_{22} = 1$

- (g) [2 points] Replace  $X_{21}$  and  $X_{22}$  by a single new variable  $X_2$  whose value is a pair of boolean values, defined as:  $X_2 = \langle X_{21}, X_{22} \rangle$ . Draw the new Bayes net  $B'$  after the change.

**Solution:**



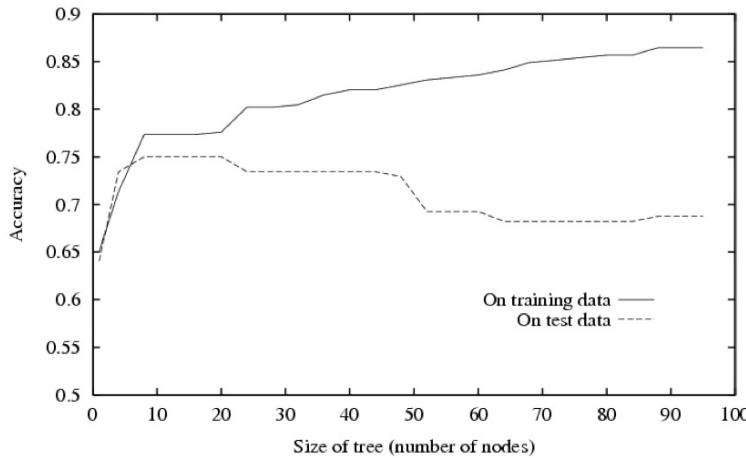
- (h) [3 points] Do all the conditional independences in  $B$  hold in the new network  $B'$ ? If not, write one that is true in  $B$  but not in  $B'$ . Consider only the variables present in both  $B$  and  $B'$ .

**Solution:**

No. For instance,  $X_{32}$  is not conditionally independent of  $X_{33}$  given  $X_{22}$  anymore.

\* Note: We noticed the problem description was a bit ambiguous, so we also accepted yes as a correct answer

## Question 5. Overfitting and PAC Learning



- (a) Consider the training set accuracy and test set accuracy curves plotted above, during decision tree learning, as the number of nodes in the decision tree grows. This decision tree is being used to learn a function  $f : X \rightarrow Y$ , where training and test set examples are drawn independently at random from an underlying distribution  $P(X)$ , after which the trainer provides a noise-free label  $Y$ . Note error = 1 - accuracy. Please answer each of these true/false questions, and explain/justify your answer in 1 or 2 sentences.

- i. [2 points] T or F: Training error at each point on this curve provides an unbiased estimate of true error.

**Solution:**

False. Training error is an optimistically biased estimate of true error, because the hypothesis was chosen based on its fit to the training data.

- ii. [1 point] T or F: Test error at each point on this curve provides an unbiased estimate of true error.

**Solution:**

True. The expected value of test error (taken over different draws of random test sets) is equal to true error.

- iii. [1 point] T or F: Training accuracy minus test accuracy provides an unbiased estimate of the degree of overfitting.

**Solution:**

True. We defined overfitting as test error minus training error, which is equal to training accuracy minus test accuracy.

- iv. [1 point] T or F: Each time we draw a different test set from  $P(X)$  the test accuracy curve may vary from what we see here.

**Solution:**

True. Of course each random draw from  $P(X)$  may vary from another draw.

- v. [1 point] T or F: The variance in test accuracy will increase as we increase the number of test examples.

**Solution:**

False. The variance in test accuracy will *decrease* as we increase the size of the test set.

(b) Short answers.

- i. [2 points] Given the above plot of training and test accuracy, which size decision tree would you choose to use to classify future examples? Give a one-sentence justification.

**Solution:**

The tree with 10 nodes. This has the highest test accuracy of any of the trees, and hence the highest expected true accuracy.

- ii. [2 points] What is the amount of overfitting in the tree you selected?

**Solution:**

overfitting = training accuracy minus test accuracy =  $0.77 - 0.74 = 0.03$

Let us consider the above plot of training and test error from the perspective of agnostic PAC bounds. Consider the agnostic PAC bound we discussed in class:

$$m \geq \frac{1}{2\epsilon^2}(\ln |H| + \ln(1/\delta))$$

where  $\epsilon$  is defined to be the difference between  $error_{true}(h)$  and  $error_{train}(h)$  for any hypothesis  $h$  output by the learner.

- iii. [2 points] State in one carefully worded sentence what the above PAC bound guarantees about the two curves in our decision tree plot above.

**Solution:**

If we train on  $m$  examples drawn at random from  $P(X)$ , then with probability  $(1 - \delta)$  the overfitting (difference between training and true accuracy) for each hypothesis in the plot will be less than or equal to  $\epsilon$ . Note the the true accuracy is the expected value of the test accuracy, taken over different randomly drawn test sets.

- iv. [2 points] Assume we used 200 training examples to produce the above decision tree plot. If we wish to reduce the overfitting to half of what we observe there, how many training examples would you suggest we use? Justify your answer in terms of the agnostic PAC bound, in *no more than two sentences*.

**Solution:**

The bound shows that  $m$  grows as  $\frac{1}{2\epsilon^2}$ . Therefore if we wish to halve  $\epsilon$ , it will suffice to increase  $m$  by a factor of 4. We should use  $200 \times 4 = 800$  training examples.

- v. [2 points] Give a one sentence explanation of why you are not certain that your recommended number of training examples will reduce overfitting by exactly one half.

**Solution:**

There are several reasons, including the following. 1. Our PAC theory result gives a bound, not an equality, so 800 examples might decrease overfitting by more than half. 2. The "observed" overfitting is actually the test set accuracy, which is only an estimate of true accuracy, so it may vary from true accuracy and our "observed" overfitting will vary accordingly.

- (c) You decide to estimate of the probability  $\theta$  that a particular coin will turn up heads, by flipping it 10 times. You notice that if repeat this experiment, each time obtaining as new set of 10 coin flips, you get different resulting estimates. You repeat the experiment  $N = 20$  times, obtaining estimates  $\hat{\theta}^1, \hat{\theta}^2 \dots \hat{\theta}^{20}$ . You calculate the variance in these estimates as

$$var = \frac{1}{N} \sum_{i=1}^{i=N} (\hat{\theta}^i - \theta^{mean})^2$$

where  $\theta^{mean}$  is the mean of your estimates  $\hat{\theta}^1, \hat{\theta}^2 \dots \hat{\theta}^{20}$ .

- i. [4 points] Which do you expect to produce a smaller value for  $var$ : a Maximum likelihood estimator (MLE), or a Maximum a posteriori (MAP) estimator that uses a Beta prior? Assume both estimators are given the same data. Justify your answer in one sentence.

**Solution:**

We should expect the MAP estimate to produce a smaller value for  $var$ , because using the Beta prior is equivalent to adding in a fixed set of "hallucinated" training examples that will *not* vary from experiment to experiment.

# 1 Comparison of Machine Learning Algorithms [Jayant, 20 points]

In this problem, you will review the important aspects of the algorithms we have learned about in class. For every algorithm listed in the two tables on the next pages, fill out the entries under each column according to the following guidelines. Turn in your completed table with your problem set. [ $\approx \frac{1}{2}$  point per entry]

## Guidelines:

1. **Generative or Discriminative** – Choose either “generative” or “discriminative”; you may write “G” and “D” respectively to save some writing.
2. **Loss Function** – Write either the name or the form of the loss function optimized by the algorithm (e.g., “exponential loss”).
3. **Decision Boundary / Regression Function Shape** – Describe the shape of the decision surface or regression function, e.g., “linear”. If necessary, enumerate conditions under which the decision boundary has different forms.
4. **Parameter Estimation Algorithm / Prediction Algorithm** – Name or concisely describe an algorithm for estimating the parameters or predicting the value of a new instance. Your answer should fit in the provided box.
5. **Model Complexity Reduction** – Name a technique for limiting model complexity and preventing overfitting.

**Solution:** Completed tables are on the following pages.

Learning Method	Generative or Discriminative?	Loss Function	Decision Boundary	Parameter Estimation Algorithm	Model Complexity Reduction
Gaussian Naïve Bayes	Generative	$-\log P(X, Y)$	Equal variance: linear boundary. Unequal variance: quadratic boundary	Estimate $\hat{\mu}$ , $\hat{\sigma}^2$ , and $P(Y)$ using maximum likelihood	Place prior on parameters and use MAP estimator
Logistic Regression	Discriminative	$-\log P(Y X)$	Linear	No closed form estimate. Optimize objective function using gradient descent.	$L_2$ regularization
Decision Trees	Discriminative	Either $-\log P(Y X)$ or zero-one loss	Axis-aligned partition of feature space	Many algorithms: ID3, CART, C4.5	Prune tree or limit tree depth
$K$ -Nearest Neighbors	Discriminative	zero-one loss	Arbitrarily complicated	Must store all training data to classify new points. Choose $K$ using cross validation.	Increase $K$
Support Vector Machines (with slack variables, no kernel)	Discriminative	hinge loss: $\max\{1 - y(w^T x), 0\}$	linear (depends on kernel)	Solve quadratic program to find boundary that maximizes margin	Reduce $C$
Boosting (with decision stumps)	Discriminative	exponential loss: $\exp\{-yf(x)\}$	Axis-aligned partition of feature space	AdaBoost	Reduce the number of iterations

Table 1: Comparison of Classification Algorithms

Learning Method	Loss Function	Regression Function Shape	Parameter Estimation Algorithm	Prediction Algorithm
Linear Regression (assuming Gaussian noise model)	square loss: $(\hat{Y} - Y)^2$	Linear	Solve $\beta = (X^T X)^{-1} X^T Y$	$\hat{Y} = X\beta$
Nadaraya-Watson Kernel Regression	square loss: $(\hat{Y} - Y)^2$	Arbitrary	Store all training data. Choose kernel bandwidth $h$ using cross validation.	$f(x) = \frac{\sum_i y_i K(x_i, x)}{\sum_j K(x_j, x)}$
Regression Trees	square loss: $(\hat{Y} - Y)^2$	Axis-aligned partition of feature space	Many: ID3, CART, C4.5	Move down tree based on $x$ , predict value at the leaf.

Table 2: Comparison of Regression Algorithms

## 2 Comparison of Machine Learning Algorithms [Jayant, 15 pts]

In this problem, you will review the important aspects of the algorithms we have learned about in class since the midterm. For every algorithm listed in the two tables on the next pages, fill out the entries under each column according to the following guidelines. Do not fill out the greyed-out cells. Turn in your completed table with your problem set.

### Guidelines:

1. **Generative or Discriminative** – Choose either “generative” or “discriminative”; you may write “G” and “D” respectively to save some writing.
2. **Loss Function** – Write either the name or the form of the loss function optimized by the algorithm (e.g., “exponential loss”). For the clustering algorithms, you may alternatively write a short description of the loss function.
3. **Decision Boundary** – Describe the shape of the decision surface, e.g., “linear”. If necessary, enumerate conditions under which the decision boundary has different forms.
4. **Parameter Estimation Algorithm / Prediction Algorithm** – Name or concisely describe an algorithm for estimating the parameters or predicting the value of a new instance. Your answer should fit in the provided box.
5. **Model Complexity Reduction** – Name a technique for limiting model complexity and preventing overfitting.
6. **Number of Clusters** – Choose either “predetermined” or “data-dependent”; you may write “P” and “D” to save time.
7. **Cluster Shape** – Choose either “isotropic” (i.e., spherical) or “anisotropic”; you may write “I” and “A” to save time.

**Solution:** Completed tables are on the following pages.

Learning Method	Generative or Discriminative?	Loss Function	Parameter Estimation Algorithm	Prediction Algorithm	Model Complexity Reduction
Bayes Nets	Generative	$-\log P(X, Y)$	MLE	Variable Elimination	MAP
Hidden Markov Models	Generative	$-\log P(X, Y)$	MLE	Viterbi or Forward-Backward, depending on prediction task	MAP
Neural Networks	Discriminative	Sum-squared error	Back-Propagation	Forward Propagation	Reduce number of hidden layers, regularization, early stopping

Table 1: Comparison of Classification Algorithms

Learning Method	Loss Function	Number of clusters: Predetermined or Data-dependent	Cluster shape: isotropic or anisotropic?	Parameter Estimation Algorithm
K-means	Within-class squared distance from mean	Predetermined	Isotropic	K-means
Gaussian Mixture Models (identity covariance)	$-\log P(X)$ , (equivalent to within-class squared distance from mean)	Predetermined	Isotropic	Expectation Maximization (EM)
Single-Link Hierarchical Clustering	Maximum distance between a point and its nearest neighbor within a cluster	Data-dependent	Anisotropic	Greedy agglomerative clustering
Spectral Clustering	Balanced cut	Predetermined	Anisotropic	Run Laplacian Eigenmaps followed by K-means or thresholding eigenvector signs

Table 2: Comparison of Clustering Algorithms

First Name: \_\_\_\_\_ Last Name: \_\_\_\_\_

## Sample Machine Learning Questions, 2010

*Note: This sample exam is to indicate the style of questions you should expect. Check the topic list for the particular topics to be covered on your exam. The actual exam will have space for your answers and is likely to contain 50-60 points of questions.*

1. (8 pts) Short answer.

- (a) What is *overfitting*?
- (b) Formally, when are two events  $A$  and  $B$  (statistically) *independent*?
- (c) Assume we have a two-class (+, -) classification problem and are asked to predict the label of a new instance  $x$  where we know  $P(+ | x) = 0.6$  and  $P(- | x) = 0.4$ . How can the *Bayes optimal prediction* on  $x$  be “-”?
- (d) Which one of the following statements is most applicable for the perceptron algorithm?  
Which statement is most applicable for logistic regression?  
(note:  $x$  represents the features and  $y$  represents the label)
  - learns a  $P(y | x)$  model?
  - uses a  $P(x | y)$  assumption?
  - is not associated with conditional probabilities?

2. ( 3 pts) Describe the Backpropagation algorithm used to train Neural Networks.

3. (6 pts) Short answer, but show enough work so I can follow your calculations.

- (a) Assume that the instance consist of a single real-valued feature. What Gaussian (Normal distribution with mean  $\mu$  and standard deviation  $\sigma$ ) maximizes the likelihood of the four point dataset  $\{-1, 1, 1, 7\}$ ?
- (b) Naive Bayes. Consider the Naive Bayes algorithm when trained on the following dataset:

$x_1$	$x_2$	$x_3$	label
1	1	1	+
1	0	1	+
1	1	0	+
1	0	1	-
0	0	0	-

What prediction does it make on the instance  $x_1 = 1, x_2 = 0, x_3 = 0$ .

4. (5 pts) Assume we have a coin that has some unknown probability  $h$  of coming up heads (and probability  $1 - h$  of coming up tails). If the coin is flipped four times getting three heads and a tail (HHHT) then:
- (2 pts) What is the maximum likelihood estimate for  $h$ ?
  - (3 pts) Assume that (before flipping the coin) we have a prior density  $p(h)$  for the various values of  $h \in [0, 1]$ . Give the formula for the posterior probability density  $p(h | \text{HHHT})$  as a function of the prior  $p(h)$ .
5. (4 pts) Assume that we are running AdaBoost on a sample of 3 examples  $\{(x_1, +1), (x_2, +1), (x_3, -1)\}$ , and after iteration  $T$  we have a master hypothesis  $H_T(x_i) = \sum_t^T \alpha_t h_t(x_i)$  as follows (recall that the  $h_t$ 's are the weak hypotheses produced during previous iterations):

$x_i$	$H(x(i))$
$x_1$	$\ln(2)$
$x_2$	$-\ln(3)$
$x_3$	$-\ln(2)$

Find the distribution on these points used by AdaBoost in the next iteration (2 pts).

Next (2 pt), suppose the next weak hypothesis  $h_{T+1}$  seen by AdaBoost correctly labels all four of the points. How will  $h_{T+1}$  be incorporated into the new master hypotheses  $H_{T+1}$ ?

**Statistical Machine Learning (GU4241/GR5241)**  
Spring 2017  
<https://courseworks.columbia.edu>

**Cynthia Rush**  
cgr2130  
**Peter Lee, Gabriel Loaiza**  
jl4304, gl2480

## MIDTERM EXAM

Total time: 75 minutes. To be taken in-class, Tuesday 7 March 2017.

**Do not open this exam until instructed. Carefully read the following instructions. You may not receive help from your neighbors, your friends, the internet, or any other source beyond your own knowledge of the material and your reference sheet. If you do so, you will receive a 0 grade on the midterm and will possibly fail the class or face expulsion from the program.**

Write your name, UNI, and the course title on the cover of the blue book. All solutions should be written in the accompanying blue book. No other paper (including this exam sheet) will be graded. **To receive credit for this exam, you must submit blue book with the exam paper placed inside.** As reference you may use one sheet of  $8.5 \times 11$  in paper, on which any notes can be written (front and back). No other materials are allowed (including calculators, textbooks, computers, and other electronics). To receive full credit on multi-point problems, you must explain how you arrived at your solutions. Each problem is divided up into several parts. Many parts can be answered independently, so if you are stuck on a particular part, you may wish to skip that part and return to it later. Good luck.

1. (25 points) Please briefly explain your answer for the following questions.

(a) (3 points) Consider the convex optimization problem over  $x \in \mathbb{R}^2$ :

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Consider a point  $x^*$  with  $\nabla f(x^*) = \begin{bmatrix} 1.2 \\ 0.7 \end{bmatrix}$  and  $\nabla g(x^*) = \begin{bmatrix} 3.6 \\ 2.1 \end{bmatrix}$ . Is  $x^*$  a minimum?

(b) (3 points) Consider the convex optimization problem over  $x \in \mathbb{R}^2$ :

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \end{aligned}$$

Consider a point  $x^*$  with  $\nabla f(x^*) = \begin{bmatrix} 1.2 \\ 0.7 \end{bmatrix}$  and  $\nabla g(x^*) = \begin{bmatrix} 3.6 \\ 2.1 \end{bmatrix}$ . Is  $x^*$  a minimum?

(c) (4 points) Briefly (1 sentence) describe how an ensemble method works.

(d) (3 points) Considering a binary classifier:  $y_i \in \{-1, +1\}$ , if we are exclusively interested in minimizing misclassification rate, what loss function should we use?

(e) (4 points) In the cascade classifier used to train face detection we modified the standard loss that you identified above. In what we did we modify the loss and why did we do this?

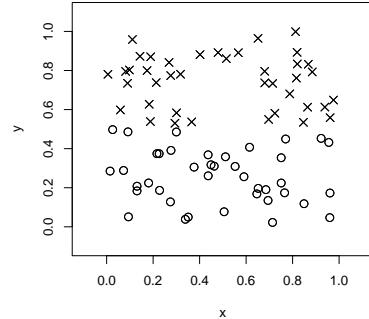
(f) (5 points) Assume we know the class conditional distributions of the data:  $p(\mathbf{x}|y = +1)$  is exactly spherical Gaussians with mean vector  $\mu_+$  and covariance  $\sigma^2 I$ , and similar for the  $-1$  class:  $p(\mathbf{x}|y = -1) = \mathcal{N}(\mathbf{x}; \mu_-, \sigma^2 I)$ . Notice that  $\sigma$  is the same for both classes, but the mean vectors are not. Describe what happens to the misclassification rate for different values of  $\mu_+$ ,  $\mu_-$ ,  $\sigma^2$ . More specifically under what scenario would a classifier achieve a low misclassification rate, and conversely a high misclassification rate? Justify your answer.

(g) (3 points) If the data is distributed according to the above assumptions, what method should we use to learn the Bayes-optimal classifier?

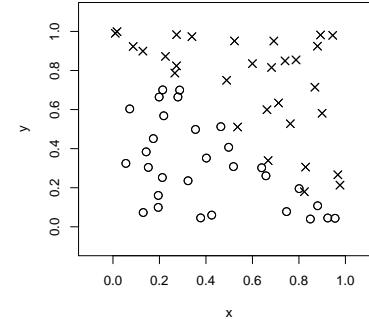
2. (16 points) Decision trees.

- (a) (8 points) For each of the following data sets, explain whether or not a basic decision tree of depth 2 will excel in classifying the data. If not, propose a classifier that will.

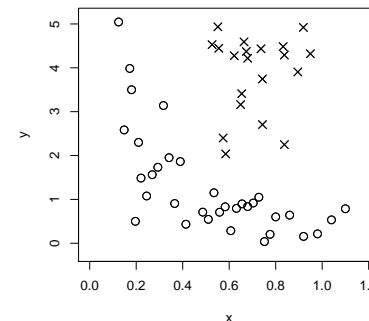
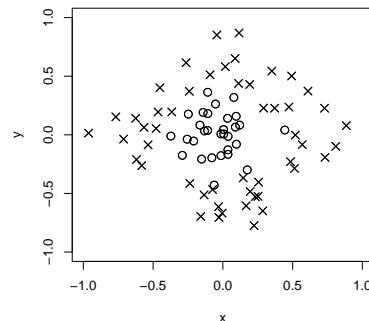
[A]



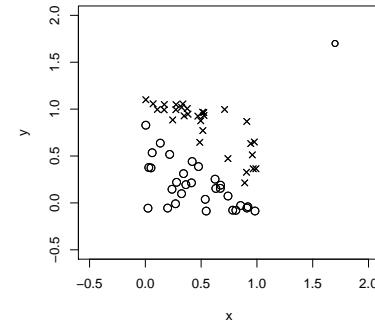
[B]



[C]



- (b) (4 points) Consider training AdaBoost on the following data set, where the data are linearly separable except for the existence of an outlier. What would happen to the weight assigned to that outlier after many boosting iterations? Would you conclude that AdaBoost is robust to outliers?



- (c) (4 points) In AdaBoost, would you stop the iteration if the error rate of the current weak learner on the weighted training data is 0? Explain.

3. (20 points) Gambling.

A particular gambling scheme involves  $n$  rounds of play. In the first round, the payoff is  $X_1 \sim \exp(\theta, \lambda)$  for some payoff parameters  $\theta$  and  $\lambda$ . For the  $i$ th round, the payoff is  $X_i | X_{i-1} \sim \exp(\theta + X_{i-1}, \lambda + X_{i-1})$ . Recall: we say  $X \sim \exp(a, b)$  if:

$$f_X(x) = ae^{-a(x-b)}\mathbb{I}\{x \geq b\}.$$

- (a) (8 points) I give you the observed payouts from all  $n$  rounds,  $X_1, X_2, \dots, X_n$ . What is the maximum likelihood estimate of  $\lambda$ , namely  $\lambda_{ML}$ ?
- (b) (6 points) Assume we know  $\lambda_{ML}$ . I give you the observed payouts from all  $n$  rounds  $X_1, X_2, \dots, X_n$ . Write down an optimization problem for  $\theta_{ML}$ . Be sure to transform the optimization to a more computationally tractable form.
- (c) (6 points) Write down an optimization algorithm that will optimize the function from the previous part. There are several choices; you may choose the simplest. Describe how the algorithm proceeds, and write an expression for the update rule, which should only involve  $\theta$ ,  $\lambda_{ML}$ , and the data  $X_1, X_2, \dots, X_n$ .

4. (14 points) Naive Bayes

- (a) (6 points) Consider the following data set with covariates  $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ :  $x_1 \in \{\text{True}, \text{False}\}$ ,  $x_2 \in \{\text{Red}, \text{Green}, \text{Blue}\}$ , and class labels,  $y \in \{-1, +1\}$ :

$y$	$x_1$	$x_2$
-1	False	Green
-1	True	Green
-1	False	Blue
-1	True	Red
+1	False	Green
+1	False	Green
+1	False	Green
+1	True	Blue

Train the Naive Bayes classifier on this data: estimate all necessary probability distributions.

- (b) (6 points) Classify the training data.  
(c) (2 points) Calculate the misclassification rate of this classifier on the training data.

5. (25 points) Kernel perceptron.

The perceptron has parameter  $z \in \mathbb{R}^{d+1}$ , and makes predictions of +1 or -1 for the input  $x$  using the classification function:

$$f(x) = \text{sgn} \left( \left\langle \begin{bmatrix} 1 \\ x \end{bmatrix}, z \right\rangle \right).$$

To learn from a labeled dataset of  $(x_i, y_i)$ ,  $i = 1, \dots, n$  where  $x_i \in \mathbb{R}^d$  and  $y_i \in \{-1, +1\}$  (with learning rate  $\alpha = 1$ ), the batch perceptron (i.e., all data points in one batch) learns by repeatedly updating  $z$  using the training rule:

$$z^{k+1} = z^k - \sum_{i=1}^n \mathbb{I}\{f(x_i) \neq y_i\} \cdot (-y_i) \begin{bmatrix} 1 \\ x_i \end{bmatrix}.$$

Recall that the above update is the gradient descent update rule for the perceptron objective:

$$C(z) = \sum_{i=1}^n \mathbb{I}\{f(x_i) \neq y_i\} \cdot \left| \left\langle z, \begin{bmatrix} 1 \\ x_i \end{bmatrix} \right\rangle \right|.$$

- (a) (5 points) The perceptron can also be trained as a single-sample algorithm, updating  $z$  one training data point at a time. Write the training rule for single-sample perceptron, i.e., how to compute  $z^{k+1}$  given  $z^k$ .
- (b) (5 points) Using the single sample perceptron update rule and beginning the updates at  $z^0 = 0$ , after some number of iterations,  $z$  can be written as  $z = \sum_{i=1}^n a_i y_i \begin{bmatrix} 1 \\ x_i \end{bmatrix}$ . What is  $a_i$ ?
- (c) (10 points) We saw that the kernel trick produces non-linear SVM with a relatively small change to the linear SVM. We want to kernelize the perceptron algorithm to provide non-linear decision boundaries. With a mapping  $\phi$  to some feature space  $\mathcal{F}$ , we rewrite the classifier as:

$$f(x) = \text{sgn} \left( \left\langle \phi(z), \phi \left( \begin{bmatrix} 1 \\ x \end{bmatrix} \right) \right\rangle_{\mathcal{F}} \right) = \text{sgn} \left( \sum_{i=1}^n w_i k(x_i, x) \right).$$

Implicit in the second equality is that we have enforced something called the “representer theorem”, namely  $\phi(z) = \sum_{i=1}^n w_i \phi \left( \begin{bmatrix} 1 \\ x_i \end{bmatrix} \right)$ . Thus we now have parameters  $w_1, \dots, w_n$ ; which, for convenience, we can initialize to  $w_i^0 = 0$  for  $i = 1, \dots, n$ .

What is the training rule for kernel perceptron; that is, how do we update  $w^{k+1}$  from  $w^k$ ?

- (d) (5 points) Do you expect kernel perceptron would generalize well? Why or why not?

---

# Machine Learning Midterm Answers

---

This exam is open book. You may bring in your homework, class notes and textbooks to help you. You will have 1 hour and 15 minutes. Write all answers in the blue books provided. Please make sure YOUR NAME is on each of your blue books. Square brackets [] denote the points for a question.

## 1. Linear Algebra

- (a) [10] Show that the *Woodbury* identity is true:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

- (b) [15] Show that if a matrix  $A$  is positive definite, its eigenvalues must be positive also.

Answer to part a:

Premultiply by  $(A + BCD)$  and cancel identity matrices on both sides, then rearrange :

$$BCDA^{-1} = B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} + BCDA^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

Cancel  $DA^{-1}$ ,

$$BC = B(C^{-1} + DA^{-1}B)^{-1} + BCDA^{-1}B(C^{-1} + DA^{-1}B)^{-1}$$

Postmultiply by  $(C^{-1} + DA^{-1}B)^{-1}$ ,

$$BC(C^{-1} + DA^{-1}B) = B + BCDA^{-1}B$$

Answer to part b:

Eigenvalue equation is:

$$Av = \lambda v$$

Premultiply by  $v^T$

$$v^T Av = \lambda v^T v$$

Both  $v^T Av$  and  $v^T v$  are positive, so  $\lambda$  must be positive also.

## 2. Entropy

- (a) [15] Show that where  $M$  is the number of states of  $\{x_i\}$ , the entropy of  $p(x_i)$  is maximized by  $p(x_i) = \frac{1}{M}$  and  $H = \ln M$ . Use the Lagrange multiplier technique.
- (b) [10] What is the Kullback-Liebler distance and why is it useful?

Answer to part a:

$$J = - \sum_{i=1}^M p(x_i) \log p(x_i) + \lambda \left( \sum_{i=1}^M p(x_i) - 1 \right) \quad (1)$$

$$\frac{\partial J}{\partial p(x_i)} = \log p(x_i) - 1 + \lambda = 0 \quad (2)$$

So

$$\lambda = 1 - \log p(x_i)$$

and since this must work for *all* the  $p(x_i)$  and there is only one  $\lambda$  then all the  $p(x_i)$  must be equal.

Answer to part b:

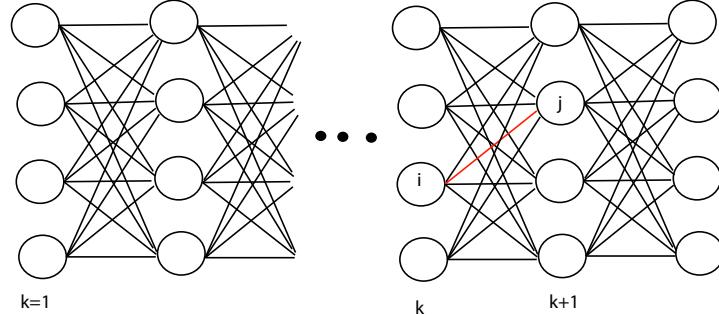
The K-L divergence, or relative entropy is a way of measuring the ‘distance’ between two distributions, since

$$\sum_k p(x_k) \log \frac{p(x_k)}{q(x_k)} \geq 0$$

This is very useful since in a very large number of applications we want to approximate an ideal distribution  $p(x_k)$  with a computable distribution  $q(x_k)$ .

### 3. Optimization

- (a) [20] The following graph represents probabilities for transiting from one state to another in stages.  $P_{ijk}$  represents the probability of transiting



from node  $i$  to node  $j$  at stage  $k$ . Specify a dynamic programming algorithm that calculates *the most probable path* through this graph from any node to the end. What is your recursion equation?

- (b) [5] In general, when might the dynamic programming method be used over the Hamiltonian method?

Answer to part a:

Let  $V(i, k)$  be the most probable path from the  $i$ th node in the  $k$ th stage until the end - lets call the last stage  $K$ . Then  $V_{iK} = 0$  and

$$V(i, k - 1) = \max_j \{P_{ijk} + V(j, k)\}$$

The most probable paths are given by  $V(i, 1)$ .

Answer to part b (one of several):

Use DP when it is feasible to represent the state space in discrete form.

## 4. Support Vector Machines

The XOR problem is given by:

$x_1 \quad x_2 \quad$  desired output  $d$

$$\begin{array}{ccc} -1 & -1 & -1 \\ +1 & -1 & +1 \\ -1 & +1 & +1 \\ +1 & +1 & -1 \end{array}$$

(a) [20] Will the following  $\psi(\mathbf{x})$  solve the XOR problem? Show all steps.

$$\psi(\mathbf{x}) = \begin{bmatrix} x_1^2 - x_2^2 \\ x_1 x_2 \\ x_1^2 + x_2^2 \end{bmatrix}$$

(b) [5] What is the Perceptron limitation and how to SVMs deal with it?

Answer to part a:

$$\psi(-1, -1) = \begin{pmatrix} 0 & 1 & 2 \end{pmatrix}, \psi(-1, 1) = \begin{pmatrix} 0 & -1 & 2 \end{pmatrix}$$

$$K = \begin{bmatrix} 5 & 3 & 3 & 3 \\ 3 & 5 & 3 & 3 \\ 3 & 3 & 5 & 3 \\ 3 & 3 & 3 & 5 \end{bmatrix}$$

$$Q(\lambda) = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 - \frac{1}{2} \{ 5\lambda_1^2 - 6\lambda_1\lambda_2 - 6\lambda_1\lambda_3 + 6\lambda_1\lambda_4 + \dots \}$$

$$\frac{\partial Q}{\partial \lambda_1} = 0 = 1 - 5\lambda_1 + 3\lambda_2 + 3\lambda_3 + -3\lambda_4$$

From symmetry,

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{2}$$

So plugging in,  $Q(\lambda) = \frac{5}{2}$  and  $\|\mathbf{w}_o\| = \sqrt{5}$ . And

$$\mathbf{w}_o = \sqrt{5} \begin{pmatrix} 0 \\ 4 \\ 0 \end{pmatrix}$$

and finally,  $\mathbf{w}_o^T \psi(\mathbf{x}) = 4\sqrt{5}x_1x_2$ .

Answer to part b:

The higher dimensional space of SVMs increases class separations.

# 10702/36702 Statistical Machine Learning, Spring 2008: Homework 1 Solutions

February 6, 2008

## 1 [14 points]

Let  $\Theta$  be a finite set. Let  $L(\theta, \hat{\theta}) = 0$  if  $\theta = \hat{\theta}$  and  $L(\theta, \hat{\theta}) = 1$  otherwise. Show that the posterior mode is the Bayes estimator.

★ SOLUTION:

$$\begin{aligned}
 \hat{\theta}_{bayes} &= \operatorname{argmin}_{\hat{\theta}} \hat{r}(\theta|x) \\
 &= \operatorname{argmin}_{\hat{\theta}} \int L(\theta, \hat{\theta}) \pi(\theta|x) d\theta \\
 &= \operatorname{argmin}_{\hat{\theta}} \int I(\theta \neq \hat{\theta}) \pi(\theta|x) d\theta \\
 &= \operatorname{argmin}_{\hat{\theta}} [1 - \int I(\theta = \hat{\theta}) \pi(\theta|x) d\theta] \\
 &= \operatorname{argmin}_{\hat{\theta}} [1 - P(\hat{\theta}|x)] \\
 &= \operatorname{argmax}_{\hat{\theta}} P(\hat{\theta}|x) \\
 &= \text{posterior mean}
 \end{aligned}$$

## 2 [30 points]

Let  $X \sim N(\theta, 1)$ . Suppose that  $\theta \in \Theta = [-C, C]$  where  $C = 1/2$ . Assume squared error loss.

- (a) Verify that  $\hat{\theta} = C \tanh(CX)$  is minimax. Hint: Show that  $\hat{\theta}$  is the Bayes estimator under the prior  $\pi = (1/2)\delta_{-C} + (1/2)\delta_C$  where  $\delta_a$  denotes a distribution that puts probability 1 at  $a$ . You may assume that  $R(\theta, \hat{\theta})$  has the following properties: it is continuous, symmetric about 0 and increasing on  $[0, c]$ .

★ SOLUTION: Under the given prior, the posterior probability

$$\begin{aligned}
 \pi(\theta = C|X) &= \frac{\frac{1}{2} \cdot \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(X-C)^2}{2}\right)}{\frac{1}{2} \cdot \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(X-C)^2}{2}\right) + \frac{1}{2} \cdot \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(X+C)^2}{2}\right)} \\
 \pi(\theta = -C|X) &= \frac{\frac{1}{2} \cdot \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(X+C)^2}{2}\right)}{\frac{1}{2} \cdot \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(X-C)^2}{2}\right) + \frac{1}{2} \cdot \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(X+C)^2}{2}\right)} \\
 \pi(\theta|X) &= 0, \text{ if } \theta \neq C \text{ and } \theta \neq -C
 \end{aligned}$$

With squared error loss, the Bayes estimator is the posterior mean, i.e.

$$\mathbb{E}(\theta|X) = C \cdot \pi(\theta = C|X) - C \cdot \pi(\theta = -C|X) = C \tanh(CX) = \hat{\theta}$$

The Bayes risk  $R_\pi(\hat{\theta}) = \pi(\theta = C)R(C, \hat{\theta}) + \pi(\theta = -C)R(-C, \hat{\theta})$ .  $R(\theta, \hat{\theta}) = \int (\hat{\theta} - \theta)^2 \frac{1}{\sqrt{2\pi}} \exp(-\frac{(x-\theta)^2}{2}) dx =$

$\int (C \tanh(Cx) - \theta)^2 \frac{1}{\sqrt{2\pi}} \exp(-\frac{(x-\theta)^2}{2}) dx$ . Since  $R(\theta, \hat{\theta})$  is continuous, symmetric about 0 and increasing on  $[0, C]$ , we have  $R_\pi(\hat{\theta}) = R(C, \hat{\theta}) > R(\theta, \hat{\theta})$ ,  $\forall \theta \in [-C, C]$ . According to Theorem 4,  $\hat{\theta} = C \tanh(CX)$  is minimax.

(b) Find the mle (maximum likelihood estimator)  $\hat{\theta}$ .

★ SOLUTION: The mle:  $\hat{\theta}_{mle} = X$ , if  $-C \leq X \leq C$ ;  $\hat{\theta}_{mle} = C$ , if  $X > C$ ;  $\hat{\theta}_{mle} = -C$ , if  $X < -C$ .

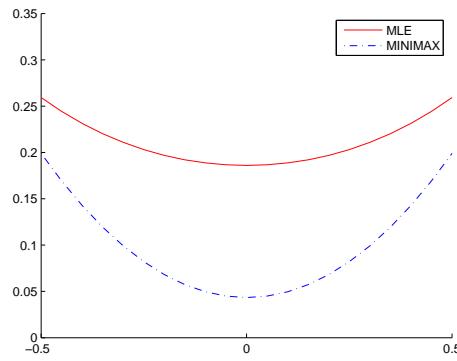
(c) Find the risk of the mle.

★ SOLUTION: The risk of the mle:

$$\begin{aligned} R(\theta, \hat{\theta}_{mle}) &= E_\theta((\theta - \hat{\theta}_{mle})^2) = \int_{-\infty}^{+\infty} (\theta - \hat{\theta}_{mle})^2 f(x; \theta) dx \\ &= \int_{-\infty}^{-C} (\theta + C)^2 \frac{1}{\sqrt{2\pi}} \exp(-\frac{(x-\theta)^2}{2}) dx + \int_C^{+\infty} (\theta - C)^2 \frac{1}{\sqrt{2\pi}} \exp(-\frac{(x-\theta)^2}{2}) dx \\ &\quad + \int_{-C}^C (\theta - x)^2 \frac{1}{\sqrt{2\pi}} \exp(-\frac{(x-\theta)^2}{2}) dx \\ &= (\theta + C)^2 \Phi(-\theta - C) + (\theta - C)^2 \Phi(\theta - C) + \int_{-C}^C (\theta - x)^2 \frac{1}{\sqrt{2\pi}} \exp(-\frac{(x-\theta)^2}{2}) dx \end{aligned}$$

Where  $\Phi(\cdot)$  is the standard normal cdf.

(d) Plot the risk functions of these two estimators.



★ SOLUTION:

### 3 [20 points]

Let  $X \sim \text{Binomial}(n, \theta)$ .

(a) Find a minimax estimator. Hint: Consider a Bayes estimator based on beta prior.

★ SOLUTION:  $X \sim \text{Binomial}(n, \theta)$        $\theta \sim \text{Beta}(\alpha, \beta)$

$$\begin{aligned}
 \pi(\theta|x) &\propto f(x|\theta)\pi(\theta) \\
 &\propto \binom{n}{x} \theta^x (1-\theta)^{n-x} \theta^{\alpha-1} (1-\theta)^{\beta-1} \\
 &\propto \theta^{\alpha+x-1} (1-\theta)^{n+\beta-x-1} \\
 \text{Hence, } \theta|x &\sim \text{Beta}(\alpha+x, n+\beta-x)
 \end{aligned}$$

$$\hat{\theta}(x)_{\text{bayes}} = E(\pi(\theta|x)) = \frac{\alpha+x}{n+\alpha+\beta} \quad (1)$$

$$\begin{aligned}
 R(\theta, \hat{\theta}) &= \text{bias}_{\hat{\theta}}^2(\hat{\theta}) + \text{Var}_{\hat{\theta}}(\hat{\theta}) \\
 &= [E_{\theta}(\theta - \frac{\alpha+x}{n+\alpha+\beta})]^2 + \text{Var}_{\theta}(\frac{\alpha+x}{n+\alpha+\beta}) \quad \dots \quad (\text{from 1}) \\
 &= \frac{\theta^2[(\alpha+\beta)^2 - n] + \theta[n - 2\alpha(\alpha+\beta)] + \alpha^2}{(n+\alpha+\beta)^2}
 \end{aligned}$$

$R(\theta, \hat{\theta})$  is constant in  $\theta$  if  $(\alpha+\beta)^2 = n$  and  $2\alpha(\alpha+\beta) = n$ .

Solving these equations we get the following values:

$$\alpha = \beta = \frac{\sqrt{n}}{2} \quad (2)$$

Substituting eq. 2 into eq. 1, we get

$$\hat{\theta}(x)_{\text{minimax}} = \frac{x + \frac{\sqrt{n}}{2}}{n + \sqrt{n}}$$

- (b) Plot the risk of the minimax estimator, the mle and the Bayes estimator using a flat prior, for  $n = 5, 50, 100$ .

★ SOLUTION: Minimax Risk:  $R(\theta, \hat{\theta}_{\text{minimax}}) = \frac{\alpha^2}{(n+\alpha+\beta)^2} = \frac{1}{4(\sqrt{n+1})^2}$

MLE estimate:

$$\hat{\theta}_{\text{mle}} = \frac{x}{n}$$

MLE Risk:

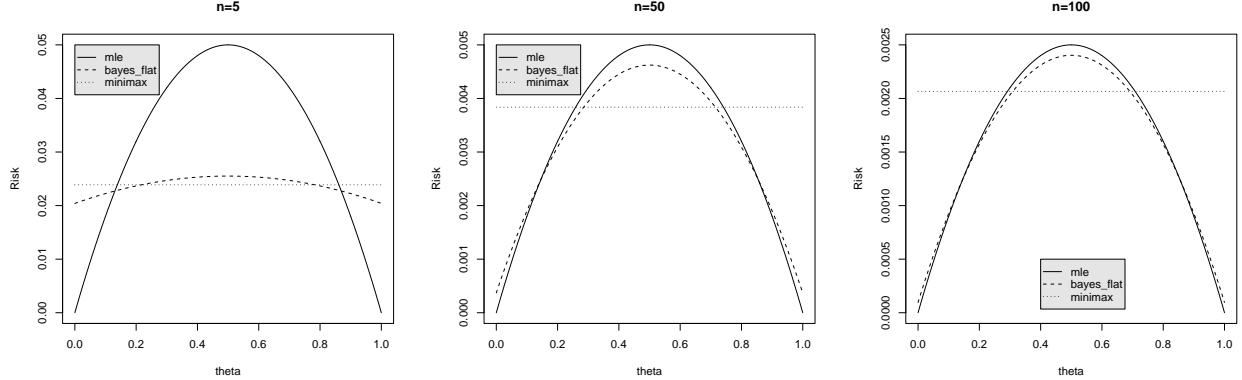
$$\begin{aligned}
 R(\theta, \hat{\theta}_{\text{mle}}) &= \text{bias}_{\hat{\theta}}^2(\hat{\theta}) + \text{Var}_{\hat{\theta}}(\hat{\theta}) \\
 &= (\theta - \frac{E(x)}{n})^2 + \text{Var}_{\theta}(\frac{x}{n}) \\
 &= (\theta - \frac{n\theta}{n})^2 + \frac{1}{n^2}n\theta(1-\theta) \\
 &= \frac{\theta(1-\theta)}{n}
 \end{aligned}$$

Bayes estimate with flat prior is equivalent to  $\text{Beta}(1, 1)$  prior. Hence using eq. 1,

$$\hat{\theta}_{\text{bayes}} = \frac{x+1}{n+2}$$

Bayes Risk:

$$\begin{aligned}
R(\theta, \hat{\theta}_{bayes}) &= \text{bias}_{\theta}^2(\hat{\theta}) + \text{Var}_{\theta}(\hat{\theta}) \\
&= (E_{\theta}(\theta - \frac{x+1}{n+2}))^2 + \text{Var}_{\theta}(\frac{x+1}{n+2}) \\
&= \frac{n\theta(1-\theta) + (2\theta-1)^2}{(n+2)^2}
\end{aligned}$$



## 4 [36 points]

This question will help you explore the differences between Bayesian and frequentist inference. Let  $X_1, \dots, X_n$  be a sample from a multivariate normal distribution with mean  $\mu = (\mu_1, \dots, \mu_p)^T$  and covariance matrix equal to the identity matrix  $I$ . Note that each  $X_i$  is a vector of length  $p$ .

The following facts will be helpful. If  $Z_1, \dots, Z_k$  are independent  $N(0, 1)$  and  $a_1, \dots, a_k$  are constants, then we say that  $Y = \sum_{j=1}^k (Z_j + a_j)^2$  has a non-central  $\chi^2$  distribution with  $k$  degrees of freedom and noncentrality parameter  $\|a\|^2$ . The mean and variance of  $Y$  are  $k + \|a\|^2$  and  $2k + 4\|a\|^2$ .

- (a) Find the posterior under the improper prior  $\pi(\mu) = 1$ .

★ SOLUTION:

$$\begin{aligned}
\pi(\mu | X_1, \dots, X_n) &\propto L_n(\mu) \pi(\mu) \\
&= \prod_{i=1}^n [(2\pi)^{-p/2} \exp(-\frac{1}{2}(x_i - \theta)^T(x_i - \theta))] \\
&\propto \exp(-\frac{n}{2}(\mu - \bar{x})^T(\mu - \bar{x}))
\end{aligned}$$

where  $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ . Therefore, the posterior of  $\mu$  is multivariate Gaussian with mean  $\bar{x}$  and covariance matrix  $\frac{1}{n} I_{p \times p}$ .

- (b) Let  $\theta = \sum_{j=1}^p \mu_j^2$ . Our goal is to learn  $\theta$ . Find the posterior for  $\theta$ . Express your answers in terms of noncentral  $\chi^2$  distributions. Find the posterior mean  $\tilde{\theta}$ .

★ **SOLUTION:**  $\forall j \in [1, \dots, p]$ ,  $\pi(\sqrt{n}\mu_j | X_1, \dots, X_n) \sim N(\sqrt{n}\bar{x}_j, 1)$ , where  $\bar{x}_j = \frac{1}{n} \sum_{i=1}^n x_{ij}$ . Therefore,  $\pi(n\theta | X_1, \dots, X_n) \sim \chi_p^2(n\|\bar{x}\|^2)$ , and the posterior mean  $\tilde{\theta} = \frac{1}{n} \mathbb{E}(n\theta | X_1, \dots, X_n) = \frac{1}{n}(p + n\|\bar{x}\|^2) = \frac{p}{n} + \|\bar{x}\|^2$ .

(c) The usual frequentist estimator is  $\hat{\theta} = \|\bar{X}\|^2 - p/n$ . Show that, for any  $n$ ,

$$\frac{\mathbb{E}_\theta \|\theta - \tilde{\theta}\|^2}{\mathbb{E}_\theta \|\theta - \hat{\theta}\|^2} \rightarrow \infty$$

as  $p \rightarrow \infty$ .

★ **SOLUTION:**  $\forall j \in [1, \dots, p]$ ,  $\bar{X}_j = \frac{1}{n} \sum_{i=1}^n X_{ij}$ . It is easy to see that  $\bar{X}_j \sim N(\mu_j, 1/n)$ , and  $\sqrt{n}\bar{X}_j \sim N(\sqrt{n}\mu_j, 1)$ . Therefore,  $n\|\bar{X}\|^2 \sim \chi_p^2(n\|\mu\|^2)$ ,  $\mathbb{E}_\theta \|\bar{X}\|^2 = \frac{p}{n} + \|\mu\|^2 = \frac{p}{n} + \theta$ , and  $\mathbb{V}_\theta(\|\bar{X}\|^2) = \frac{2p}{n^2} + \frac{4\|\mu\|^2}{n} = \frac{2p}{n^2} + \frac{4\theta}{n}$ .

$$\mathbb{E}_\theta \|\theta - \tilde{\theta}\|^2 = (\theta - \mathbb{E}_\theta \tilde{\theta})^2 + \mathbb{V}_\theta(\tilde{\theta}) = (\mathbb{E}_\theta \|\bar{X}\|^2 + \frac{p}{n} - \theta)^2 + \mathbb{V}_\theta(\|\bar{X}\|^2) = \frac{4p^2}{n^2} + \frac{2p}{n^2} + \frac{4\theta}{n}.$$

$$\mathbb{E}_\theta \|\theta - \hat{\theta}\|^2 = (\theta - \mathbb{E}_\theta \hat{\theta})^2 + \mathbb{V}_\theta(\hat{\theta}) = (\mathbb{E}_\theta \|\bar{X}\|^2 - \frac{p}{n} - \theta)^2 + \mathbb{V}_\theta(\|\bar{X}\|^2) = \frac{2p}{n^2} + \frac{4\theta}{n}.$$

Therefore, for any  $n$ , as  $p \rightarrow \infty$

$$\frac{\mathbb{E}_\theta \|\theta - \tilde{\theta}\|^2}{\mathbb{E}_\theta \|\theta - \hat{\theta}\|^2} = \frac{\frac{4p^2}{n^2} + \frac{2p}{n^2} + \frac{4\theta}{n}}{\frac{2p}{n^2} + \frac{4\theta}{n}} \rightarrow \infty$$

(d) Repeat the analysis with a  $N(0, \tau^2 I)$  prior.

★ **SOLUTION:** By similar analysis, we have  $\pi(\mu | X_1, \dots, X_n) \sim N(\frac{n}{n+\frac{1}{\tau^2}}\bar{x}, \frac{1}{n+\frac{1}{\tau^2}}I_{p \times p})$ ,  $\pi((n + \frac{1}{\tau^2})\theta | X_1, \dots, X_n) \sim \chi_p^2(\frac{n^2}{n+\frac{1}{\tau^2}}\|\bar{x}\|^2)$ . Therefore, the posterior mean  $\tilde{\theta} = \frac{p}{n+\frac{1}{\tau^2}} + \frac{n^2}{(n+\frac{1}{\tau^2})^2}\|\bar{x}\|^2$ .

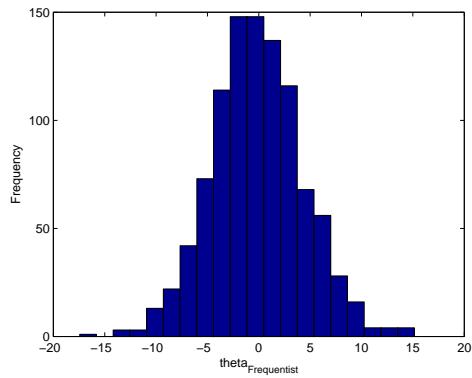
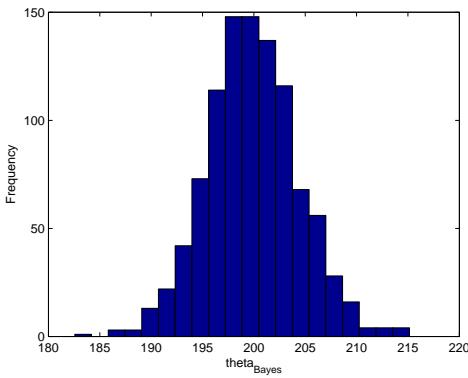
$$\mathbb{E}_\theta \|\theta - \tilde{\theta}\|^2 = (\theta - \mathbb{E}_\theta \tilde{\theta})^2 + \mathbb{V}_\theta(\tilde{\theta}) = (\mathbb{E}_\theta(\frac{n^2}{(n+\frac{1}{\tau^2})^2}\|\bar{X}\|^2) + \frac{p}{n+\frac{1}{\tau^2}} - \theta)^2 + \mathbb{V}_\theta(\frac{n^2}{(n+\frac{1}{\tau^2})^2}\|\bar{X}\|^2) = (\frac{2n+\frac{1}{\tau^2}}{(n+\frac{1}{\tau^2})^2}p - \frac{\frac{2n}{\tau^2} + \frac{1}{\tau^4}}{(n+\frac{1}{\tau^2})^2}\theta)^2 + \frac{n^2}{(n+\frac{1}{\tau^2})^4}(2p + 4n\theta).$$

Therefore, for any  $n$ , as  $p \rightarrow \infty$

$$\frac{\mathbb{E}_\theta \|\theta - \tilde{\theta}\|^2}{\mathbb{E}_\theta \|\theta - \hat{\theta}\|^2} = \frac{(\frac{2n+\frac{1}{\tau^2}}{(n+\frac{1}{\tau^2})^2}p - \frac{\frac{2n}{\tau^2} + \frac{1}{\tau^4}}{(n+\frac{1}{\tau^2})^2}\theta)^2 + \frac{n^2}{(n+\frac{1}{\tau^2})^4}(2p + 4n\theta)}{\frac{2p}{n^2} + \frac{4\theta}{n}} \rightarrow \infty$$

(e) Set  $n = 10$ ,  $p = 1000$ ,  $\theta = (0, \dots, 0)^T$ . Simulate (in R) data  $N$  times, with  $N = 1000$ . Draw a histogram of the Bayes estimator (with flat prior) and the frequentist estimator.

★ **SOLUTION:** The histograms are as follows.



(f) Interpret your findings.

★ **SOLUTION:** From the figures, we can see that the two histograms have the same shape, and the frequentist estimator has less bias compared with the Bayes estimator. According to (c), we can see that the frequentist estimator ( $\hat{\theta} = \|\bar{X}\|^2 - \frac{p}{n}$ ) is unbiased  $\theta - \mathbb{E}_{\theta}\hat{\theta} = \theta$ , whereas the Bayes estimator with flat prior ( $\tilde{\theta} = \|\bar{X}\|^2 + \frac{p}{n}$ ) is biased,  $\theta - \mathbb{E}_{\theta}\tilde{\theta} = \frac{2p}{n}$ . This bias is significant when  $p$  is much larger than  $n$ .

# Statistical Machine Learning – Solutions for Exam 2017-03-10

1.
  - i. False, but its output is quantitative.
  - ii. True,  $\Phi$  is monotonically increasing and  $\beta^T X$  is a linear function of  $\beta$ . This means that the decision boundary  $\Phi(\beta^T x) = 0.2 \Leftrightarrow \beta^T x = \Phi^{-1}(0.2)$  is linear.
  - iii. True, this is what boosting does for example.
  - iv. True, a binary splitting cannot produce open corners like the lower right corner of the upper right rectangle.
  - v. False, the term  $\beta_0\beta_1$  is not linear in the parameters.
  - vi. False,  $k$ -NN (as any other method) does not overfit by adding more data. Adding more data reduces the error in average, when keeping  $k$  constant.
  - vii. False, least squares is a solution to maximum likelihood in some special cases.
  - viii. True.
  - ix. True, AdaBoost can use any base classifier.
  - x. True, since the  $k$ -NN classifier is based on computing distances between inputs, and these distances are sensitive to the scaling of the input variables.
2.
  - (a) Linear regression since this is a regression problem and not a classification problem (the output, `price`, is quantitative).
  - (b)
    - id - neither. The price does not depend on the particular id of the wine.
    - grape - input. The price could depend on the grape used.
    - alcohol - input. Unknown if necessary but straightforward to implement and try out.

- year - input. The price is probably dependent on the year of the wine.
  - region - neither or input. The region of the wine could be a useful input for predicting the price. However, since this variable has so many categories, it could be difficult to extract any useful information from it without using a very large training data set. One could consider some kind of partitioning as mentioned in the d) part, but with a better heuristic.
  - proline - input. Unknown if necessary but straightforward to implement and try out.
  - timestamp - neither. The price does not depend on when the wine was added to the database.
  - price - output.
- (c)
- grape - qualitative. The different grapes does not have any apparent order
  - alcohol - quantitative. Is of ordinal nature
  - year - quantitative or qualitative. The year could be viewed as having an ordinal nature (“the older the better”), however it could also be argued that there are certain years that are particularly good for producing wines (without any ordinal nature) and from this viewpoint the variable is qualitative.
  - region - qualitative. The different regions does not have any apparent order.
  - proline - quantitative. Is of ordinal nature.
  - price - quantitative. As argued in a).
- (d) It is intractable to enumerate all unique subsets  $I \subset R$  since there are  $2^{78} - 2$  proper subsets ( $-2$  coming from the exclusion of  $I = \emptyset$  and  $I = R$ ). Even if we could process the subsets in 1GHz it would take 10 million years.
3. (a) The LDA classifier is linear and will therefore separate the two classes with a straight line in 2D. Looking at the scatter plots in Figure 2 it is clear that dataset i and iii can be separated using a straight line, but dataset ii can not. The QDA classifier is non-linear and will hence result in a non-linear (quadratic) decision boundary. Looking at the scatter plots in Figure 2 it is clear that in all three plots the two classes can be separated by a quadratic curve.

- (b) Both LDA and QDA assume the inputs corresponding to each class have a Gaussian distribution with some mean value  $\mu_k$  for each class  $k$ . The difference between LDA and QDA is that LDA assumes that the covariance matrix  $\Sigma$  is the same for all classes  $k$  whereas QDA assumes that each class  $k$  has a unique covariance matrix  $\Sigma_k$ .

Looking at dataset i it is clear that there does not seem to be any correlation (pos or neg) in any of the classes and the spread seems to be the same. Hence the assumptions of LDA (same covariance matrix for both classes) are fulfilled by dataset i. (Since QDA is a generalization of LDA, we could also argue that dataset i satisfies the assumptions of QDA as well.)

Looking at dataset iii it is clear that there seems to be correlation in both classes. Since the correlation is positive for  $Y = 0$  and negative for  $Y = 1$  the two classes have different covariance matrices. Hence the assumptions of QDA (but not LDA) are fulfilled by dataset iii.

Looking at dataset ii, class  $Y = 0$  seems to have a Gaussian distribution whereas the class  $Y = 1$  does not have a Gaussian distribution. Hence dataset ii does not fulfill the assumptions of either LDA or QDA.

(Note that even though dataset ii does not fulfill the assumptions of QDA, a QDA decision boundary can still separate the classes in this case! The same holds for LDA and dataset iii. This shows that the LDA/QDA classifier *can* perform well in practice, even if the assumptions are not satisfied.)

- (c) Note first that none of the figures have periodic sample paths. Consequently, model M3 (which has a periodic kernel) is the model which is not represented among the four. For comparison, sample paths from model M3 are shown in the figure below.

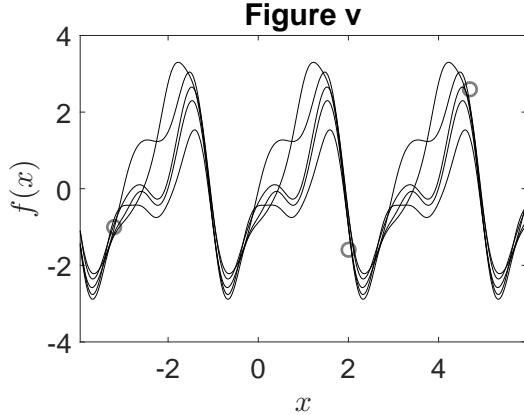


Figure 1: Sample paths from model M3, which has a periodic kernel.

Next, we note that the noise variance  $\sigma^2$  decides how concentrated the posterior distribution will be around the observed data points. Specifically, setting  $\sigma^2 = 0$  is the same as saying that the function  $f$  is observed exactly, i.e.  $y_i = f(x_i)$ . This forces the sample paths from the posterior distribution to pass exactly through the measured data. Finally, we note that the length scale  $\ell$  determines how quickly the correlation between different points in the  $x$ -space decays. This is also visible from the plots of the kernels  $k(r) = k(|x - x'|)$ , corresponding to the covariance between two input points at a distance of  $r$  from each other. Based on these insights, we can draw the following conclusions:

**Figure ii:** All sample paths pass exactly through the measured data points. Hence this plot must correspond to model M5, which is the only model with  $\sigma^2 = 0$ . It can also be observed that the sampled paths are a bit “ragged” which is typical for a Matérn kernel.

**Figure iii:** All sample paths seems to be almost independent of the measured data points (at least compared to the other three figures). Hence this plot must correspond to the distribution with the highest noise variance. M2 has the highest noise variance ( $\sigma^2 = 3^2$  compared to  $\sigma^2 = 0.3^2$  and  $\sigma^2 = 0$ ) so this must be samples from M2. It can also be observed that the sampled paths are very smooth, hence they should belong to a squared-exponential kernel.

**Figure i:** There are strong correlations between points quite far apart (compare to Figure vi where only points very close to each other are noticeably correlated). Hence this plot must correspond

to M4 which has a large length scale  $\ell = 5$  (i.e., the plot of  $k(r)$  decays slowly to zero for M4). Note also that the paths are a bit ragged and should therefore belong to a Matérn kernel.

**Figure iv:** There is very little correlation between points far apart which corresponds to a kernel with a small length scale. Note also that the paths are smooth and should therefore belong to a squared-exponential kernel. Hence this plot must correspond to M1.

4. (a) The model  $Y = \beta_0 + \beta_1 X + \varepsilon$  for training points  $\mathcal{T} = \{(x_1, y_1), (x_2, y_2)\} = \{(-1, 2), (1, 1)\}$  gives

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \beta_0 + \beta_1 x_1 + \varepsilon_1 \\ \beta_0 + \beta_1 x_2 + \varepsilon_2 \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \end{pmatrix}$$

$$\Leftrightarrow \mathbf{y} = \mathbf{X}\beta + \varepsilon$$

where (for the given training points)

$$\begin{aligned} \mathbf{X} &= \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \mathbf{X}^T = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \mathbf{X}^T \mathbf{X} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \\ (\mathbf{X}^T \mathbf{X})^{-1} &= \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \mathbf{y} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \end{aligned}$$

$\hat{\beta}_{LS}$  is then given by

$$\hat{\beta}_{LS} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} = \begin{pmatrix} 3/2 \\ -1/2 \end{pmatrix}.$$

(This problem can also be solved by noting that the least-squares estimate of  $\beta$  must correspond to the straight line which connects the two training data points.)

- (b) Same  $\mathbf{X}$  and  $\mathbf{y}$  as in a) for the given training points gives

$$\begin{aligned} \hat{\beta}_{RR} &= (\mathbf{X}^T \mathbf{X} + \lambda I)^{-1} \mathbf{X}^T \mathbf{y} = \left( \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} + \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \right)^{-1} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 2 + \lambda & 0 \\ 0 & 2 + \lambda \end{pmatrix}^{-1} \begin{pmatrix} 3 \\ -1 \end{pmatrix} = \frac{1}{2 + \lambda} \begin{pmatrix} 3 \\ -1 \end{pmatrix} \end{aligned}$$

- (c) The hint (LASSO estimate of  $\beta_j$  is either 0 or the same sign as  $\hat{\beta}_j$  for LS) gives that for LASSO  $\hat{\beta}_0$  is either zero or positive and  $\hat{\beta}_1$

is either zero or negative. LASSO aims to minimize the following sum (for given training data)

$$\begin{aligned}
S &= \sum_{i=1}^2 (y_i - (\beta_0 + \beta_1 x_i))^2 + \lambda \sum_{j=0}^1 |\beta_j| \\
&= (2 - (\beta_0 - \beta_1))^2 + (1 - (\beta_0 + \beta_1))^2 + \lambda |\beta_0| + \lambda |\beta_1| \\
&= [\text{Use hint, } \beta_0 \geq 0 \text{ and } \beta_1 \leq 0] = \\
&= 5 - 6\beta_0 + 2\beta_1 + 2\beta_0^2 + 2\beta_1^2 + \lambda\beta_0 - \lambda\beta_1, \tag{*}
\end{aligned}$$

where the last equality holds for  $\lambda$  sufficiently small, so that both  $\beta_0$  and  $\beta_1$  are non-zero. Now, minimize the sum with respect to  $\beta$  gives

$$\begin{aligned}
\frac{\partial S}{\partial \beta_0} = -6 + 4\beta_0 + \lambda &= 0 \Rightarrow \beta_0 = \begin{cases} \frac{6-\lambda}{4} & \text{if } \lambda < 6 \\ 0 & \text{if } \lambda \geq 6 \end{cases} \\
\frac{\partial S}{\partial \beta_1} = 2 + 4\beta_1 - \lambda &= 0 \Rightarrow \beta_1 = \begin{cases} -\frac{2-\lambda}{4} & \text{if } \lambda < 2 \\ 0 & \text{if } \lambda \geq 2 \end{cases}
\end{aligned}$$

(Technically, the equation  $(*)$  only holds for  $\lambda \leq 2$ , since from the solution for  $\beta_1$  there is a breakpoint at  $\lambda = 2$ . However, since there are no cross terms between  $\beta_0$  and  $\beta_1$  in  $S$ , this does not affect the solution for  $\beta_0$ . In other words, for the interval  $\lambda = [2, 6]$  we have another expression for  $S$  obtained by setting  $\beta_0 = 0$  in  $(*)$ , which results in the same solution for  $\beta_1$  as above.)

5. (a) Parametric models are parameterized using a fixed-dimensional vector of parameters. Learning a parametric model amounts to estimating the values of these parameters. These estimates are subsequently used to define the prediction model for a parametric model. Nonparametric models, on the other hand, use (in some way) the training data as parameters, i.e. the prediction model for a parametric model is expressed directly in terms of the training data. Nonparametric models can thus be viewed as having infinitely many parameters. The flexibility of a nonparametric model thus increases as we use more training data.
- (b) From the problem we have that the probability for  $Y = 1$  (according to the model) is the logistic function of  $\beta^T X$  i.e.

$$\Pr(Y = 1 | \beta, X) = \frac{e^{\beta X}}{1 + e^{\beta^T X}}.$$

Since the probability of  $Y = 0$  is the complement to the probability of  $Y = 1$  we have,

$$\Pr(Y = 0 | \beta, X) = 1 - \Pr(Y = 1 | \beta, X) = 1 - \frac{e^{\beta^T X}}{1 + e^{\beta^T X}} = \frac{1}{1 + e^{\beta^T X}}.$$

Since the first element of  $\beta$  is an intercept we insert a 1 first in  $x_*$  which leads to the estimate,

$$\hat{\beta}^T \begin{pmatrix} 1 \\ x_* \end{pmatrix} = (3 \ -1 \ 3 \ 2) \begin{pmatrix} 1 \\ 2 \\ 1 \\ -1 \end{pmatrix} = 2,$$

and finally

$$\Pr\left(Y = 0 | \beta, X = \begin{pmatrix} 1 \\ x_* \end{pmatrix}\right) = \frac{1}{1 + e^2} \approx 12\%$$

- (c) Since all data points are independent we can write the full probability as a product i.e.,

$$\begin{aligned} \Pr(Y_1 = y_1, \dots, Y_N = y_N | X_1 = x_1, \dots, X_N = x_N, \beta) \\ = \prod_{i=1}^N \Pr(Y = y_i | X = x_i, \beta). \end{aligned}$$

$\Pr(Y = y_i | X = x_i, \beta)$  can be expressed in multiple ways by splitting the two outcomes of  $Y$ . This example will use,

$$\Pr(Y = y_i | X = x_i, \beta) = p(x_i, \beta)^{y_i} (1 - p(x_i, \beta))^{1-y_i},$$

where  $p(x, \beta)$  is taken from the problem formulation (this is indeed the standard approach for deriving the likelihood for a logistic regression, see e.g. ESL(p120)). However you could also consider

$$\Pr(Y = y_i | X = x_i, \beta) = y_i p(x_i, \beta) + (1 - y_i) (1 - p(x_i, \beta)),$$

or more explicitly

$$\Pr(Y = y_i | X = x_i, \beta) = I(y_i = 1)p(x_i, \beta) + I(y_i = 0)(1 - p(x_i, \beta)),$$

where  $I(\cdot)$  is the indicator function. Using the first expression of these three we get,

$$\begin{aligned} \Pr(Y_1 = y_1, \dots, Y_N = y_N | X_1 = x_1, \dots, X_N = x_N, \beta) \\ = \prod_{i=1}^N p(x_i, \beta)^{y_i} (1 - p(x_i, \beta))^{1-y_i}. \end{aligned}$$

(d) Recall that for the logistic regression model we have  $p(x, \beta) = \frac{e^{\beta^T x}}{1+e^{\beta^T x}}$  and  $1 - p(x, \beta) = \frac{1}{1+e^{\beta^T x}}$ . Using this we get

$$\begin{aligned}
l(\beta) &= \log \Pr(Y_1 = y_1, \dots, Y_N = y_N \mid X_1 = x_1, \dots, X_N = x_N, \beta) \\
&= \log \prod_{i=1}^N p(x_i, \beta)^{y_i} (1 - p(x_i, \beta))^{1-y_i} \\
&= \sum_{i=1}^N y_i \log p(x_i, \beta) + (1 - y_i) \log (1 - p(x_i, \beta)) \\
&= \sum_{i=1}^N y_i \beta^T x_i - y_i \log(1 + e^{\beta^T x_i}) - (1 - y_i) \log(1 + e^{\beta^T x_i}) \\
&= \sum_{i=1}^N y_i \beta^T x_i - \log(1 + e^{\beta^T x_i}).
\end{aligned}$$

**Course: STAT W4400**  
**Title: Statistical Machine Learning**  
**Semester: Fall 2014**  
**Instructor: John P. Cunningham**

## MIDTERM EXAM

### Explanation

This exam is to be done in-class. You have 75 minutes to complete the entirety. All solutions should be written in the accompanying blue book. No other paper (including this exam sheet) will be graded. **To receive credit for this exam, you must submit blue book with the exam paper placed inside.** As reference you may use one sheet of  $8.5 \times 11$ in paper, on which any notes can be written (front and back). A calculator may be used for simple calculations only (e.g., no stored formulas). No other materials are allowed (including textbooks, computers, and other electronics). To receive full credit on multi-point problems, you must thoroughly explain how you arrived at your solutions. Each problem is divided up into several parts. Many parts can be answered independently, so if you are stuck on a particular part, you may wish to skip that part and return to it later. Good luck.

1. (25 points)

A particular gambling scheme involves  $n$  rounds of play. In the first round, the payoff for is  $X_1 \sim \exp(\theta, \lambda)$  for some payoff parameters  $\theta$  and  $\lambda$ . For the  $i$ th round, the payoff is  $X_i | X_{i-1} \sim \exp(\theta + X_{i-1}, \lambda + X_{i-1})$ . Recall: we say  $X \sim \exp(a, b)$  if:

$$f_X(x) = ae^{-a(x-b)} \mathbb{1}\{x \geq b\}$$

- (a) (5 points) I give you the observed payouts from all  $n$  rounds,  $X_1, \dots, X_n$ . What is the maximum likelihood estimate of  $\lambda$ , namely  $\lambda_{ML}$ ?

**Solution:**

We start by writing out the likelihood. Let  $X_0 := 0$ . Then

$$\begin{aligned} L(\theta, \lambda) &= \prod_{i=1}^n (\theta + X_{i-1}) e^{-(\theta + X_{i-1})(X_i - X_{i-1} - \lambda)} \mathbb{1}\{X_i \geq X_{i-1} + \lambda\} \\ &= \mathbb{1}\{\lambda \leq \Delta\} \prod_{i=1}^n (\theta + X_{i-1}) e^{-(\theta + X_{i-1})(X_i - X_{i-1} - \lambda)}, \end{aligned}$$

where  $\Delta := \min_i(X_i - X_{i-1})$ . For  $\lambda \leq \Delta$ , the likelihood is non-negative and strictly increasing in  $\lambda$ . For  $\lambda > \Delta$ , the likelihood is zero. So,  $\lambda_{ML} = \Delta$ .

Note that the usual approach of finding the MLE by taking the derivative of the log-likelihood, setting it equal to zero, and solving for  $\lambda$ . You could instead formulate the problem as a constrained optimization problem, with the constraint  $\lambda - \Delta \leq 0$ , in which case you immediately find that  $\lambda_{ML} = \infty$  if the constraint is not active (but is not a situation of interest), or that  $\lambda_{ML} = \Delta$  if the constraint is active.

- (b) (8 points) Assume we know  $\lambda_{ML}$ . I give you the observed payouts from all  $n$  rounds,  $X_1, \dots, X_n$ . Write down an optimization problem for  $\theta_{ML}$ . Be sure to transform the optimization to a more computationally tractable form.

**Solution:**

We found  $\lambda_{ML}$  in part (a), so we can write our log-likelihood as

$$\ell(\theta, \lambda_{ML}) = \sum_{i=1}^n [\log(\theta + X_{i-1}) - (\theta + X_{i-1})(X_i - X_{i-1} - \lambda_{ML})]$$

The optimization problem we want to solve is then

$$\theta_{ML} = \arg \max_{\theta \geq 0} \left\{ \sum_{i=1}^n [\log(\theta + X_{i-1}) - (\theta + X_{i-1})(X_i - X_{i-1} - \lambda_{ML})] \right\}$$

- (c) (8 points) Write down an optimization algorithm that will optimize the function from the previous part. There are several choices; you may choose the simplest. Describe how the algorithm proceeds, and write an expression for the update rule, which should only involve  $\theta$ ,  $\lambda_{ML}$ , and the data  $X_1, \dots, X_n$ .

**Solution:**

We can use any variant of gradient ascent to find the maximum. To do find the updates, we need the gradient of  $\ell(\theta, \lambda_{ML})$  with respect to  $\theta$ .

$$\frac{\partial \ell(\theta, \lambda_{ML})}{\partial \theta} = \sum_{i=1}^n \frac{1}{\theta + X_{i-1}} - (X_i - X_{i-1} - \lambda_{ML})$$

To simplify notation, let  $\Delta_i := X_i - X_{i-1}$ , and  $\bar{\Delta} := \frac{1}{n} \sum_{i=1}^n \Delta_i$ . Then our update is

$$\theta_{t+1} := \theta_t + \alpha_t \left( n\lambda_{ML} - n\bar{\Delta} + \sum_{i=1}^n \frac{1}{\theta_t + X_{i-1}} \right)$$

for some appropriately chosen step size,  $\alpha_t$ . Alternatively, we can use the second derivative to use the Newton-Raphson algorithm:

$$\theta_{t+1} := \theta_t + \left( - \sum_{i=1}^n \frac{1}{(\theta + X_{i-1})^2} \right)^{-1} \left( n\lambda_{ML} - n\bar{\Delta} + \sum_{i=1}^n \frac{1}{\theta_t + X_{i-1}} \right)$$

- (d) (4 points) In terms of the data  $X_1, \dots, X_n$ , what are the smallest and largest values that  $\lambda_{ML}$  can be?

**Solution:**

$$\lambda_{ML} = \min_i (X_i - X_{i-1}).$$

2. (25 points)

The following questions all consider a binary classifier  $f : \mathbb{R}^d \rightarrow \{-1, +1\}$ .

- (a) (3 points) Do most machine learning algorithms use risk  $R(f)$  or empirical risk  $\hat{R}_n(f)$ , and why?

**Solution:**

Empirical risk, due to uncertainty about the true distribution of the data.

- (b) (3 points) If the training data  $\{(x_1, y_1), \dots, (x_n, y_n)\}$  for a fixed classifier  $f$  are  $n$  iid draws from the true underlying distribution of the data, what is:

$$\lim_{n \rightarrow \infty} |R(f) - \hat{R}_n(f)|$$

Please make a simple argument; no proof is required. (Technical note: you may assume that  $R(f)$  is well behaved such that questions of convergence are all appropriately satisfied).

**Solution:**

0.

- (c) (3 points) Under the usual 01 loss, what is the range of  $R(f)$ ? With this answer, interpret  $R(f)$  in words as a probability (one sentence will suffice).

**Solution:**

$[0, 1]$ .  $R(f)$  under the 01 loss is the probability that a given classifier  $f$  makes an error.

- (d) (2 points) Training procedure 1 chooses linear classifiers  $f^1$  entirely at random. Now the risk  $R(f^1)$  is a random variable (a function of the random variable  $f^1$ ). What is  $E(R(f^1))$  under the 01 loss?

**Solution:**

0.5

- (e) (2 points) Training procedure 2 uses a soft-margin SVM to choose a linear classifier  $f^2$  according to a training set  $\{(x_1, y_1), \dots, (x_n, y_n)\}$  drawn iid from the true underlying distribution. By analogy to the previous part, you can consider that training procedure 2 chooses linear classifiers  $f^2$  *better than* entirely at random. Do you expect  $E(R(f^2))$  to be larger or smaller than  $E(R(f^1))$ , again under the same 01 loss?

**Solution:**

Smaller. Risk under the 01 loss is an error measure, so the soft-margin SVM will do better.

- (f) (8 points) Training procedure 3 repeats training procedure 2 independently  $m$  times (assume  $m$  is odd), each time with a new training set drawn iid from the true underlying distribution, producing classifiers  $f_1^2, f_2^2, \dots, f_m^2$ . If I let  $f^3(x) = \text{sign}(\sum_{k=1}^m f_k^2(x))$ . What is  $E(R(f^3))$  in terms of  $E(R(f^2))$ ? Do not try to simplify the solution entirely.

**Solution:**

Note that probability of a correct classification under the 0-1 loss is  $p = 1 - E(R(f^2))$ . Then use Condorcet's jury theorem.

$$\begin{aligned} P(\text{correct}) &= 1 - E(R(f^3)) \\ &= \sum_{j=\frac{m+1}{2}}^m \binom{m}{j} p^j (1-p)^{m-j} \\ &= \sum_{j=\frac{m+1}{2}}^m \binom{m}{j} (1 - E(R(f^2)))^j (E(R(f^2)))^{m-j} \\ &\Rightarrow \\ E(R(f^3)) &= \sum_{j=1}^{\frac{m+1}{2}} \binom{m}{j} (1 - E(R(f^2)))^j (E(R(f^2)))^{m-j} \end{aligned}$$

- (g) (4 points) Training procedure 4 uses AdaBoost, with  $m$  classifiers of type  $f^2$ , on a single training set  $\{(x_1, y_1), \dots, (x_n, y_n)\}$  to produce  $f^4$ . Do you expect  $E(R(f^4))$  to be larger or smaller than  $E(R(f^3))$ , again under the same 0-1 loss?

**Solution:**

Larger. Adaboost has correlated data sets.

3. (25 points)

Consider a soft-margin, linear support vector machine:

$$\begin{aligned} \min_{\mathbf{v}_H, b, \xi} \quad & \|\mathbf{v}_H\|^2 + C \sum_{i=1}^n \xi_i^2 \\ \text{s.t.} \quad & y_i(\langle \mathbf{v}_H, x_i \rangle - b) \geq 1 - \xi_i \quad \text{for } i = 1, \dots, n \\ & \xi_i \geq 0, \quad \text{for } i = 1, \dots, n \end{aligned}$$

- (a) (3 points) For increasing  $C$ , will the margin increase or decrease, and why?

**Solution:**

Errors cost more, so the margin has to decrease to avoid making errors.

- (b) (3 points) For increasing  $C$ , will  $\|\mathbf{v}_H\|$  increase or decrease, and why?

**Solution:**

Decreasing margin means increasing  $\|\mathbf{v}_H\|$ .

- (c) (4 points) For increasing  $C$ , will training error increase or decrease, and why?

**Solution:**

Errors cost more, so training error will go down.

- (d) (4 points) For increasing  $C$ , should testing error increase or decrease, and why?

**Solution:**

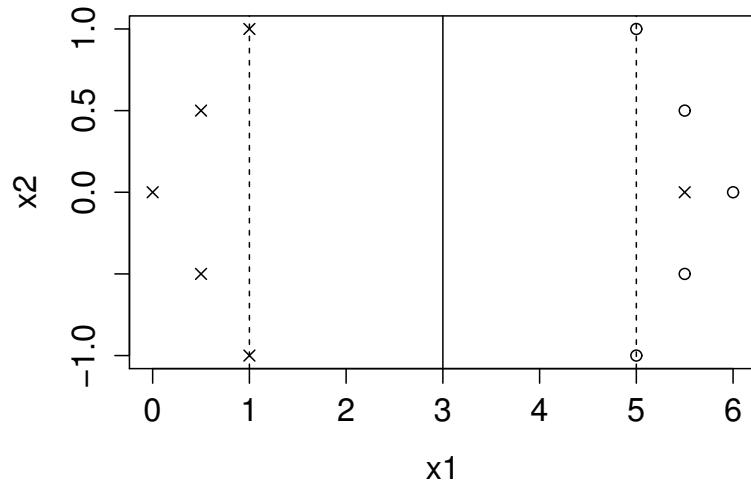
Testing error should increase, as we risk overfitting by being oversensitive to errors.

- (e) (7 points) Consider the following training data in  $\mathbb{R}^2$ . For a large but finite value of  $C$ , what will be the training error? (Note: do not try to run an SVM by hand. You should draw the data and argue the answer in a few sentences.)

$x_i^1$	0.0	1.0	1.0	0.5	0.5	5.0	5.0	6.0	5.5	5.5	5.5
$x_i^2$	0.0	-1.0	1.0	0.5	-0.5	1.0	-1.0	0.0	-0.5	0.5	0.0
$y_i$	+1	+1	+1	+1	+1	-1	-1	-1	-1	-1	+1

**Solution:**

$\frac{1}{11}$ . With a drawing, as below.



- (f) (4 points) Consider the data in the previous part. For a large but finite value of  $C$ , what are the support vectors?

**Solution:**

Points 2,3,6,7,11. The points 2,3 and 5,6 will be on the margin, and point 11 is misclassified and so is a support vector.

4. (25 points)

Let  $\mathcal{G}$  be a convex, differentiable constraint set on  $\mathbb{R}^d$ , and  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be the following convex objective:

$$f(x) = \frac{1}{2}x^\top Px + a^\top x + b.$$

Let the constraint function  $g(x)$  be:

$$g(x) = \begin{cases} < 0 & x \in \text{in}(\mathcal{G}) \\ = 0 & x \in \partial(\mathcal{G}) \\ > 0 & x \notin \mathcal{G}. \end{cases}$$

Let  $x^* = \arg \min_x f(x)$  (unconstrained) and  $x_{\mathcal{G}}^*$  be the constrained solution to:

$$\begin{array}{ll} \min_x & f(x) \\ \text{s.t.} & g(x) \leq 0 \end{array}$$

(a) (3 points) Say  $g(x^*) \leq 0$ . What is  $\nabla f(x^*)$ ?

**Solution:**

0.

(b) (3 points) Say  $g(x^*) > 0$ . What is  $\nabla f(x^*)$ ?

**Solution:**

0.

(c) (4 points) Using the given form of  $f(x)$ , what is the update step for a Newton's method in the unconstrained problem?

**Solution:**

The update step is

$$\begin{aligned} x_{t+1} &:= x_t - [\nabla^2 f(x_t)]^{-1} \nabla f(x_t) \\ &:= x_t - P^{-1}(Px_t + a) \\ &:= x_t - x_t - P^{-1}a \\ &:= -P^{-1}a \end{aligned}$$

(d) (4 points) What does this answer indicate about the convergence of Newton's method for this particular choice of  $f(x)$ ?

**Solution:**

The local Hessian approximation is globally exact, and Newton's method will converge in one step. This is expected, as  $f(x)$  is a quadratic function.

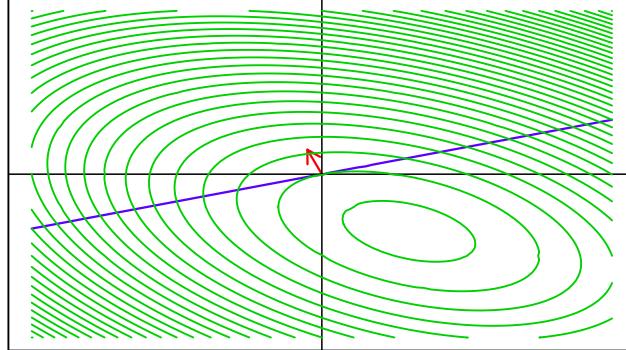
- (e) (4 points) Let  $g(x) = |c^\top x|$ . Draw the constrained problem if  $d = 2$ . Include representations of  $f(x), g(x), c$  (you need not represent  $P, a, b$  explicitly).

**Solution:**

One acceptable example is the following picture, with

$$P = \begin{bmatrix} 0.2 & 0.1 \\ 0.1 & 0.3 \end{bmatrix}, a = \begin{bmatrix} -0.1 \\ 0.3 \end{bmatrix}, b = 0, c = \begin{bmatrix} -0.2 \\ 0.6 \end{bmatrix}.$$

The green contours represent  $f(x)$ , the blue line is  $g(x) = 0$ , and the small red line is the vector  $c$ , which is orthogonal to  $g(x) = 0$ .



- (f) (7 points) Write out the optimality conditions explicitly, and simplify as much as possible. Hint: it is possible to write these conditions as a single matrix-vector solve, which allows us to find  $\begin{bmatrix} x^* \\ \lambda \end{bmatrix}$  in closed form.

**Solution:**

It is easiest to notice that  $g(x) = |c^\top x|$  implies that  $in(\mathcal{G})$  is empty, and thus we have an equality constraint  $g(x) = c^\top x = 0$ . This fact simplifies the KKT

conditions:

$$\begin{aligned} c^\top x_{\mathcal{G}}^* &= 0 \\ \nabla f(x_{\mathcal{G}}^*) + \lambda \nabla g(x_{\mathcal{G}}^*) &= 0 \end{aligned}$$

which in our setting becomes:

$$\begin{aligned} c^\top x_{\mathcal{G}}^* &= 0 \\ Px_{\mathcal{G}}^* + a + \lambda c &= 0. \end{aligned}$$

In a single matrix-vector solve, that looks like:

$$\begin{bmatrix} P & c \\ c^\top & 0 \end{bmatrix} \begin{bmatrix} x_{\mathcal{G}}^* \\ \lambda \end{bmatrix} = \begin{bmatrix} -a \\ 0 \end{bmatrix}.$$

**Course: STAT W4400**  
**Title: Statistical Machine Learning**  
**Semester: Spring 2015**  
**Instructor: John P. Cunningham**

## **MIDTERM EXAM**

**Do not open this exam until instructed. Carefully read the following instructions.**

This exam is to be done in-class. You have 75 minutes to complete the entirety. Write your name, UNI, and the course title on the cover of the blue book. All solutions should be written in the accompanying blue book. No other paper (including this exam sheet) will be graded. **To receive credit for this exam, you must submit blue book with the exam paper placed inside.** As reference you may use one sheet of  $8.5 \times 11$ in paper, on which any notes can be written (front and back). No other materials are allowed (including calculators, textbooks, computers, and other electronics). To receive full credit on multi-point problems, you must thoroughly explain how you arrived at your solutions. Each problem is divided up into several parts. Many parts can be answered independently, so if you are stuck on a particular part, you may wish to skip that part and return to it later. Good luck.

## 1. Short Answers (20 points)

- (a) (3 points) All ensemble methods are built atop underlying weak learners that are trained and evaluated. What additional requirement does boosting place on the underlying weak learners?

**Solution:**

It must be possible to train the weak learner on a weighted training set.

- (b) (1 point) Does the random forest method use bagging or boosting?

**Solution:**

Bagging.

- (c) (3 points) Can regularization reduce training MSE?

**Solution:**

No. Regularization moves the result away from the least squares solution, which by definition has the minimum training MSE.

- (d) (3 points) Can regularization reduce testing MSE?

**Solution:**

Yes. Regularization can help prevent overfitting to training data.

- (e) (3 points) Consider the convex optimization problem over  $x \in \mathbb{R}^2$ :

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \end{aligned}$$

Consider a point  $x^*$  with  $\nabla f(x^*) = \begin{bmatrix} 1.2 \\ 0.7 \end{bmatrix}$  and  $\nabla g(x^*) = \begin{bmatrix} 3.6 \\ 2.1 \end{bmatrix}$ . Is  $x^*$  a minimum?

**Solution:**

Yes. The gradient of the objective and the gradient of the constraint function are collinear.

- (f) (3 points) Consider the convex optimization problem over  $x \in \mathbb{R}^2$ :

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \end{aligned}$$

Consider a point  $x^*$  with  $\nabla f(x^*) = \begin{bmatrix} 1.2 \\ 0.7 \end{bmatrix}$  and  $\nabla g(x^*) = \begin{bmatrix} 3.6 \\ 2.1 \end{bmatrix}$ . Is  $x^*$  a minimum?

**Solution:**

No. The gradient of the objective and the gradient of the constraint function must be collinear and in opposite directions for  $x^*$  to be a constrained minimum.

- (g) (2 points) What is one advantage of LASSO over ridge regression?

**Solution:**

LASSO tends to produce sparse parameter solutions, which can be computationally beneficial.

- (h) (2 points) What is one advantage of ridge regression over LASSO?

**Solution:**

Ridge regression can be calculated in closed form (as an augmented least squares problem), unlike LASSO.

## 2. Understanding Misclassification (40 points)

The misclassification rate of machine learning classifiers depends substantially on the distribution of the data, the classification technique being used, and more. This question probes this dependency. In all problems below we will consider a binary classifier:  $y_i \in \{-1, +1\}$ .

- (a) (3 points) If we are exclusively interested in minimizing misclassification rate, what loss function should we use?

**Solution:**

The usual 01 loss:  $L^{01}(y_i, f(x_i)) = \mathbb{I}\{y_i \neq f(x_i)\}$ .

- (b) (5 points) Suppose that we are interested in the classification of an extremely rare event. It is usually the case in these settings that false positives are far less of a concern than false negatives. Would minimizing the misclassification rate still be an appropriate goal? Why or why not? If it is not appropriate, describe how you would modify the empirical risk function to better achieve our goals.

**Solution:**

Our problem is that the loss function we are using is symmetric, we can change this in our setting by modifying the empirical risk function to penalize false negatives far more than false positives. Note that this essentially is equivalent to attaching a larger weight to our observations in the rare event class.

- (c) (5 points) Now assume we know the class conditional distributions of the data:  $p(\mathbf{x}|y = +1)$  is exactly spherical Gaussians with mean vector  $\mu_+$  and covariance  $\sigma^2 I$ , and similar for the  $-1$  class:  $p(\mathbf{x}|y = -1) = \mathcal{N}(\mathbf{x}; \mu_-, \sigma^2 I)$ . Notice that  $\sigma$  is the same for both classes, but the mean vectors are not. Describe what happens to the misclassification rate for different values of  $\mu_+, \mu_-, \sigma^2$ . More specifically under what scenario would a classifier achieve a low misclassification rate, and conversely a high misclassification rate? Justify your answer.

**Solution:**

Any classifier works best when the distance between the entries in our mean vectors is high, and when the variance is low. Conversely we have a high misclassification rate when the entries of the vector are largely the same and the variance is high.

- (d) (5 points) If the data is distributed according to the above assumptions, what method should we use to learn the Bayes-optimal classifier?

**Solution:**

# 变量之间是相互独立的

In this setting the Naive Bayes assumption holds true. It is thus the estimator which minimizes Bayes risk (The Bayes estimator).

- (e) (12 points) Consider the following data set with covariates  $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ :  $x_1 \in \{\text{True}, \text{False}\}$ ,  $x_2 \in \{\text{Red}, \text{Green}, \text{Blue}\}$ , and class labels,  $y \in \{-1, +1\}$ :

$y$	$x_1$	$x_2$
-1	False	Green
-1	True	Green
-1	False	Blue
-1	True	Red
+1	False	Green
+1	False	Green
+1	False	Green
+1	True	Blue

Train the Naive Bayes classifier on this data: estimate all necessary probability distributions.

## Solution:

The Naive Bayes classifier is

$$f(\mathbf{x}) = \arg \max_{y \in \{-1, +1\}} Pr(y) \prod_{j=1}^2 Pr(x_j|y)$$

Estimate each component from the data:

$$\begin{aligned} \hat{Pr}(y = -1) &= 1/2 \\ \hat{Pr}(x_1 = \text{True}|y = -1) &= 1/2 \\ \hat{Pr}(x_1 = \text{False}|y = -1) &= 1/2 \\ \hat{Pr}(x_2 = \text{Red}|y = -1) &= 1/4 \\ \hat{Pr}(x_2 = \text{Green}|y = -1) &= 1/2 \\ \hat{Pr}(x_2 = \text{Blue}|y = -1) &= 1/4 \\ \hat{Pr}(y = +1) &= 1/2 \\ \hat{Pr}(x_1 = \text{True}|y = +1) &= 1/4 \\ \hat{Pr}(x_1 = \text{False}|y = +1) &= 3/4 \\ \hat{Pr}(x_2 = \text{Red}|y = +1) &= 0 \\ \hat{Pr}(x_2 = \text{Green}|y = +1) &= 3/4 \\ \hat{Pr}(x_2 = \text{Blue}|y = +1) &= 1/4 \end{aligned}$$

- (f) (10 points) Classify the training data and calculate the misclassification rate of this classifier on the training data.

**Solution:**

The posterior probabilities given the data and the classification rules are:

$y$	$x_1$	$x_2$	$\hat{Pr}(y = -1 \mathbf{x}) \propto$	$\hat{Pr}(y = +1 \mathbf{x}) \propto$	$\hat{y}$
-1	False	Green	1/8	9/32	+1
-1	True	Green	1/8	3/32	-1
-1	False	Blue	1/16	3/32	+1
-1	True	Red	1/16	0	-1
+1	False	Green	1/8	9/32	+1
+1	False	Green	1/8	9/32	+1
+1	False	Green	1/8	9/32	+1
+1	True	Blue	1/16	1/32	-1

We misclassify 3 data points, so the misclassification rate is  $\frac{3}{8}$ .

### 3. Nearest Neighbor Regression (40 points)

Consider a regression problem with training data  $X = \{(\tilde{\mathbf{x}}_1, \tilde{y}_1), \dots, (\tilde{\mathbf{x}}_n, \tilde{y}_n)\}$ , for  $\tilde{\mathbf{x}} \in \mathbb{R}^d$  and  $y \in \mathbb{R}$ . Given an integer parameter  $k$  and some data  $X'$ , a  $k$ -nearest neighbor ( $k$ NN) regression is the function  $f(\mathbf{x}; k, X') : \mathbb{R}^d \rightarrow \mathbb{R}$  that is computed by:

1. Finding the  $k$  points in  $X'$  closest to  $\mathbf{x}$ . That is, define  $\eta_k(\mathbf{x}; X')$  as the function that returns the indices  $i$  corresponding to these closest points in terms of Euclidean distance  $d(\mathbf{x}, \tilde{\mathbf{x}}_i) = \|\mathbf{x} - \tilde{\mathbf{x}}_i\|_2$ .
  2. Assigning  $f(\mathbf{x}; k, X')$  as the average of those  $k$  nearest neighbors.
- Mathematically, the above two operations are written as:

$$f(\mathbf{x}; k, X') = \frac{1}{k} \sum_{i \in \eta_k(\mathbf{x}; X')} y_i$$

- For example, consider the data  $X'$  and a query point  $\mathbf{x} = 2.1$ :

$\tilde{\mathbf{x}}_i$	2.0	1.4	1.5	2.5	3.6	1.2	1.9
$y_i$	11.3	12.1	10.1	14.1	10.1	19.3	10.6

Given  $k = 3$ ,  $\mathbf{x}$  has  $k$  nearest neighbors  $\{\tilde{\mathbf{x}}_1 = 2.0, \tilde{\mathbf{x}}_4 = 2.5, \tilde{\mathbf{x}}_7 = 1.9\}$  (the function  $\eta_k(\mathbf{x}; X') = \{1, 4, 7\}$ ). The regression  $f(\mathbf{x}; 3, X')$  is then the average of  $\{\tilde{y}_1 = 11.3, \tilde{y}_4 = 14.1, \tilde{y}_7 = 10.6\}$ , namely  $f(\mathbf{x}; 3, X') = 12.0$ .

Notice that, for a given choice of  $k$ , the  $k$ NN classifier requires no training; each classification result is computed directly from  $X'$ .

- (a) (5 points) Using the above data  $X'$ , what is  $f(\mathbf{x}; 2, X')$  for  $\mathbf{x} = 1.4$ ?

**Solution:**

The nearest neighbors are  $\tilde{\mathbf{x}}_2$  and  $\tilde{\mathbf{x}}_3$ , and thus  $f(\mathbf{x}; 2, X') = \frac{1}{2}(\tilde{y}_2 + \tilde{y}_3) = 11.1$ .

- (b) Describe a procedure to choose an optimal value for the parameter  $k$  out of the values  $\{1, 3, 5, 7, 9\}$ , using some data  $X$ . Hint: note that  $k$  in some way controls the complexity of the regression function. In detail:
- i. (5 points) Specify an empirical risk function you will use to compare different versions of the regression function. Give a formula for this quantity when evaluated on a subset  $X_{\text{val}}$  of  $X$ .

**Solution:**

The regressors are compared by estimating their risk on the validation data set. For a validation set  $X_{\text{val}} \subset X$ , this means we estimate the risk

$$R(f) = \int L^2(y, f(x))p(x, y)dx dy$$

from the data in  $X_{val}$  as

$$\hat{R}(f) = \frac{1}{|X_{val}|} \sum_{(x,y) \in X_{val}} L^2(y, f(x)) = \frac{1}{|X_{val}|} \sum_{(x,y) \in X_{val}} (y - f(x))^2.$$

- ii. (10 points) Write down a step-by-step procedure to choose  $k$ .

**Solution:**

Split data into training data set  $X_{train}$ , validation set  $X_{val}$  and test set  $X_{test}$ .

- For each  $k \in \{1, 3, 5, 7, 9\}$ , compute  $\hat{R}(f)$  on  $X_{val}$ , where  $f$  is the  $k$ NN classifier using  $X_{train}$ . More precisely:

$$\hat{R}(f(\cdot; k, X_{train})) = \frac{1}{|X_{train}|} \sum_{(x,y) \in X_{train}} L^2(y, f(x; k, X_{train}))$$

- Select the optimal value  $k^*$ :

$$k^* := \arg \min_{k \in \{1, 3, 5, 7, 9\}} \hat{R}(f(\cdot; k, X_{train}))$$

- Report the risk estimate for  $f(\cdot; k^*, X_{train})$  (or, ideally, for  $f(\cdot; k^*, X_{train} \cup X_{val})$ ) computed on  $X_{test}$ .

For  $m$ -fold cross validation, we first split of a test data set  $X_{test}$ . The remaining data  $X \setminus X_{test}$  is split into  $m$  blocks, each of which is used as validation set in turn in the above procedure. The risk estimate  $\hat{R}(f(\cdot; k, X_{train}))$  is computed once for each fold, then averaged over all blocks. From there on (selection of  $k^*$  etc) the two procedures are identical.

- iii. (3 points)  $k$ -nearest neighbor regression often works surprisingly well. Can you think of a reason why this approach may nonetheless be a bad choice for an application running, for example, on a phone or a digital camera?

**Solution:**

The classifier is efficient in the sense that it requires no training, but the classification step is expensive: The device executing the classification step has to store the entire training data set, and for each input data point, search over all training points for the  $k$  closest points. **For implementation on a camera or phone, a classifier with possibly expensive training but efficient classification step would be preferable.**

- (c) Since  $k$ NN only computes distances between points, your next task is to exploit

that fact to make a kernel version of this classifier, using the same kernel trick that we did in SVM. You may now assume  $k$  is given (or determined from your previous result). To derive *kernel kNN*, you must:

- i. (3 points) Consider the squared Euclidean distance between any two points  $\mathbf{x}$  and  $\mathbf{x}'$ . Expand the expression  $d^2(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2$  in terms of vector inner products.

**Solution:**

$$d^2(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2 = \langle \mathbf{x}, \mathbf{x} \rangle - 2\langle \mathbf{x}, \mathbf{x}' \rangle + \langle \mathbf{x}', \mathbf{x}' \rangle$$

- ii. (8 points) Assume you are given a kernel  $K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{F}} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ . Use the kernel trick to make the squared Euclidean distance from the previous part into a kernel squared distance. Write down this expression and call it  $d_K^2(\mathbf{x}, \mathbf{x}')$ .

**Solution:**

$$\begin{aligned} d_K^2(\mathbf{x}, \mathbf{x}') &= \langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle_{\mathcal{F}} - 2\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{F}} + \langle \phi(\mathbf{x}'), \phi(\mathbf{x}') \rangle_{\mathcal{F}} \\ &= K(\mathbf{x}, \mathbf{x}) - 2K(\mathbf{x}, \mathbf{x}') + K(\mathbf{x}', \mathbf{x}') \end{aligned}$$

- iii. (3 points)  $d_K$  is itself a distance measure. What distance does it calculate?

**Solution:**

The distance between two points in feature space, namely  $d_K(\mathbf{x}, \mathbf{x}') = \|\phi(\mathbf{x}) - \phi(\mathbf{x}')\|_{\mathcal{F}}$ .

- iv. (3 points) Algorithmically, which operations in *kNN* are affected by this kernelized version of the method: the form of the function  $f(\mathbf{x}; k, X')$ , the form of the function  $\eta_k(\mathbf{x}, X')$ , neither, or both?

**Solution:**

The neighbor function  $\eta$  is affected; it becomes closest neighbors in feature space. The regression function  $f$  is unchanged (though its results will change, of course).