

# 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.)

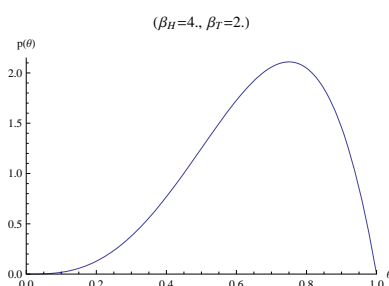


Figure 1:  $Beta(4, 2)$  distribution

★ **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!*

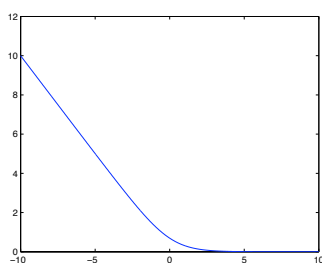
*Nirvana  
Grunge*

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

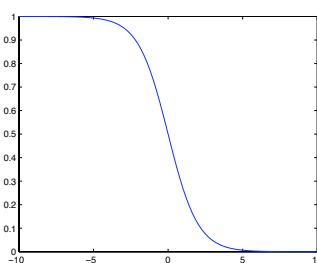
## 2 [12 points] Loss Function

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  $yF(x) = y(w_0 + \sum_{j=1}^d w_j x_j)$

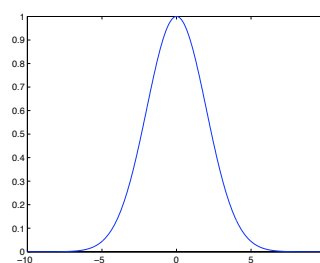
1. [4 points] Which loss functions below are appropriate to use in classification? For the ones that are not appropriate, explain why not. In general, what conditions does  $L$  have to satisfy in order to be an appropriate loss function? The x axis is  $yF(x)$ , and the y axis is  $L(yF(x))$ .



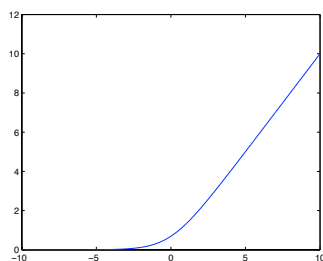
(a)



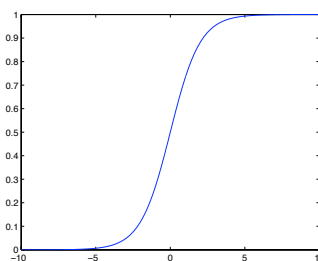
(b)



(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

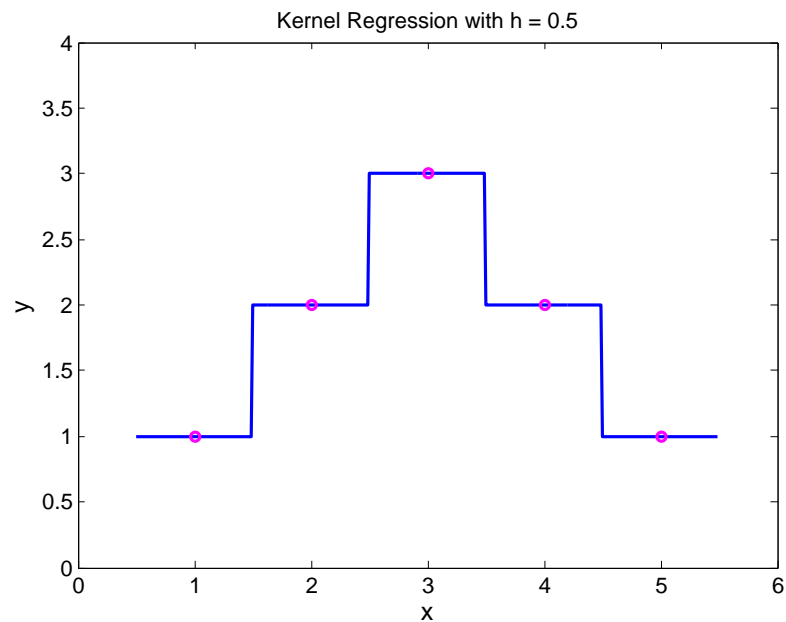
1. [4 points] Sketch the fit  $\hat{Y}$  given  $X$  for the dataset given below using kernel regression with a box kernel

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

for  $h = 0.5, 2$ .

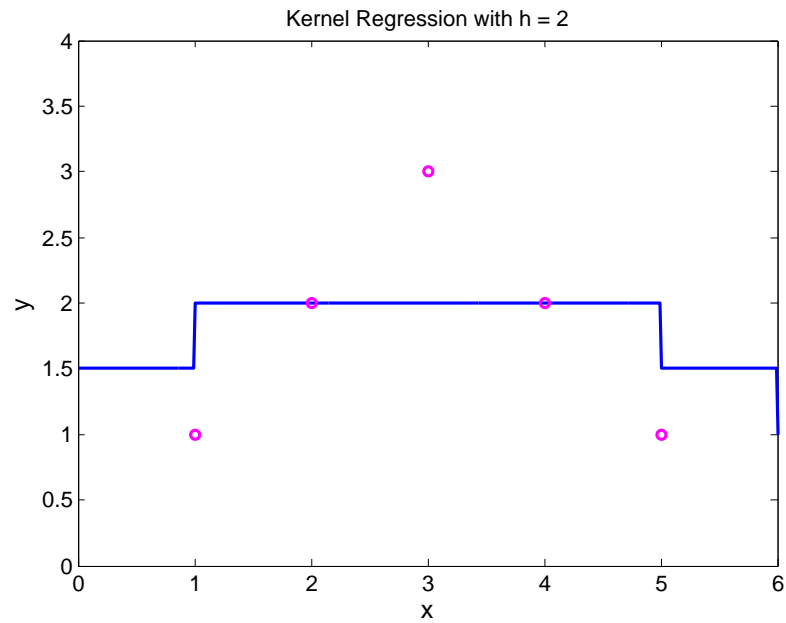
★ SOLUTION:

- $h = 0.5$



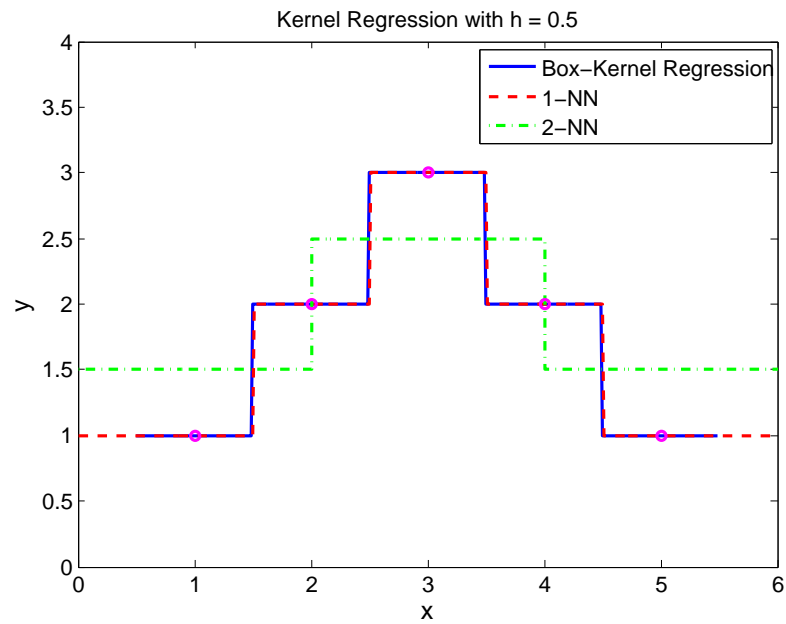


- $h = 2$

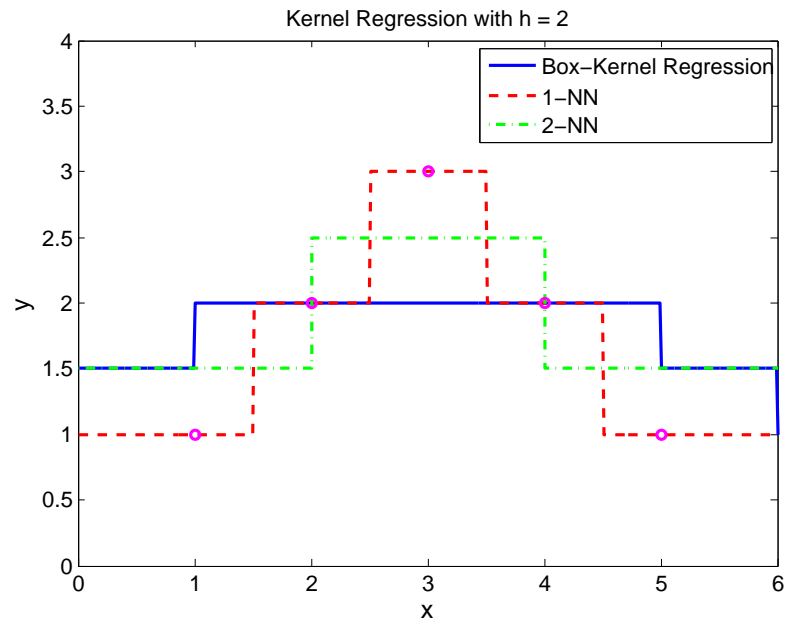


Just for fun, what happens with 1-NN and 2-NN:

- $h = 0.5$



- $h = 2$



■ **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.

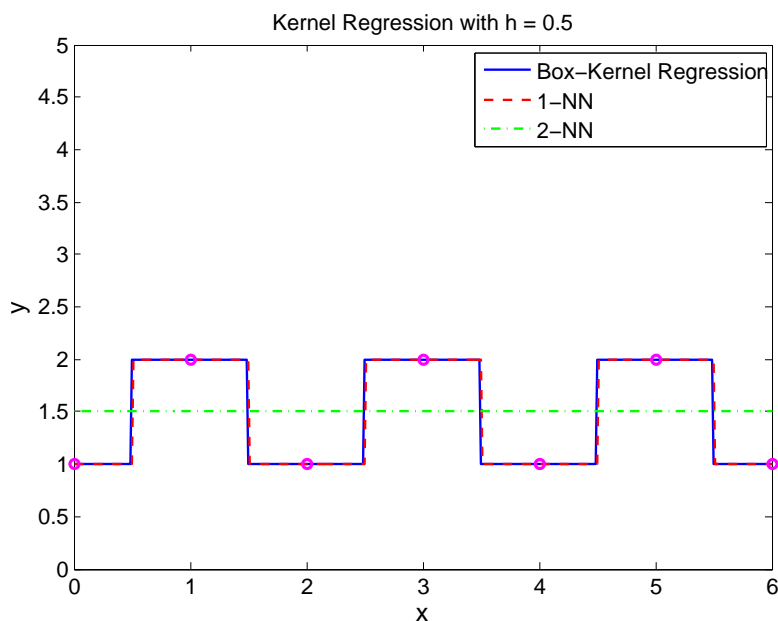
2. [4 points] Sketch or describe a dataset where kernel regression with the box kernel above with  $h = 0.5$  gives the same regression values as 1-NN but not as 2-NN in the domain  $x \in [0, 6]$  below.

★ **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).

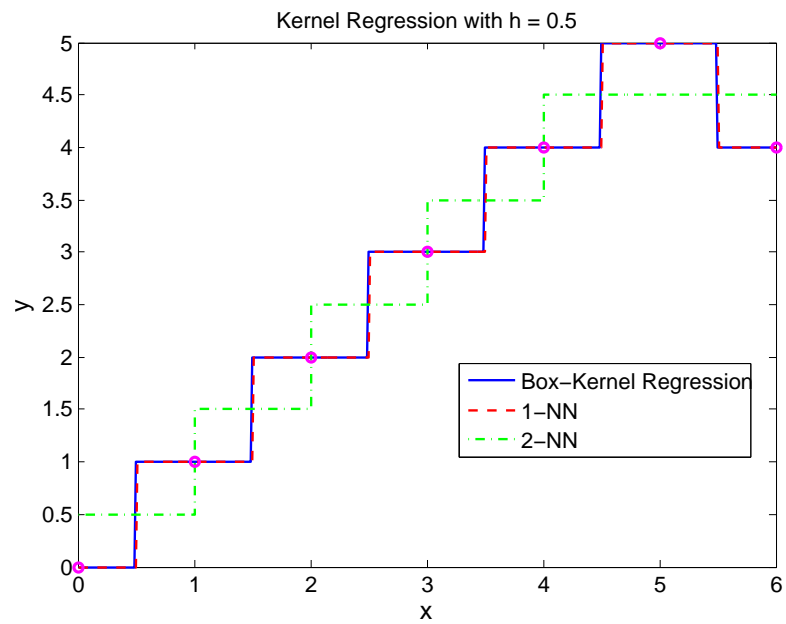
■ **COMMON MISTAKE 1:** Some people tried to come up with examples for  $k$ -NN *classification* instead of regression.

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.

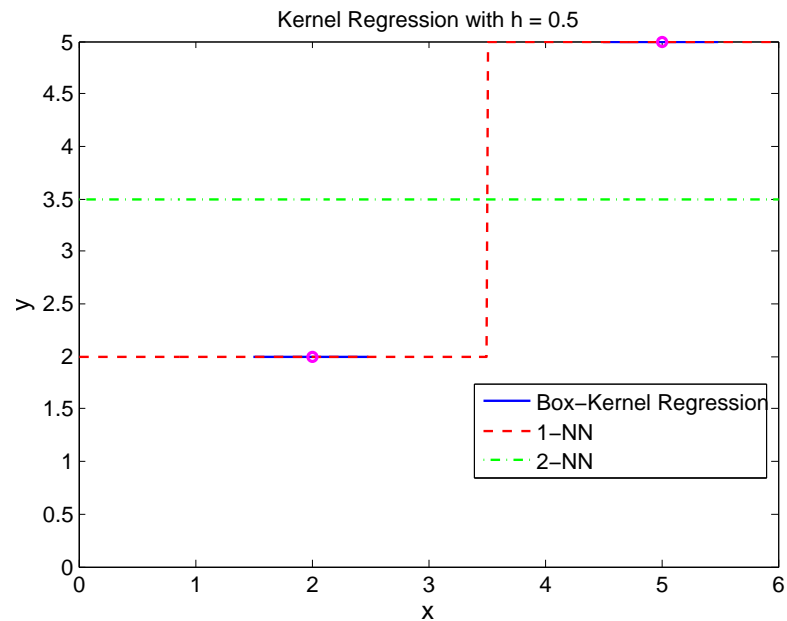
- Example 1



- Example 2



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

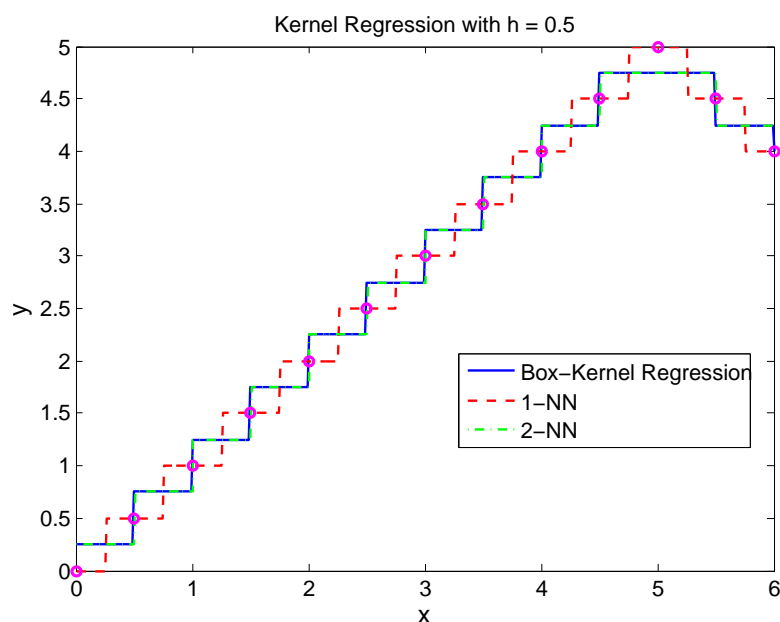


3. [4 points] Sketch or describe a dataset where kernel regression with the box kernel above with  $h = 0.5$  gives the same regression values as 2-NN but not as 1-NN in the domain  $x \in (0, 6)$  below.

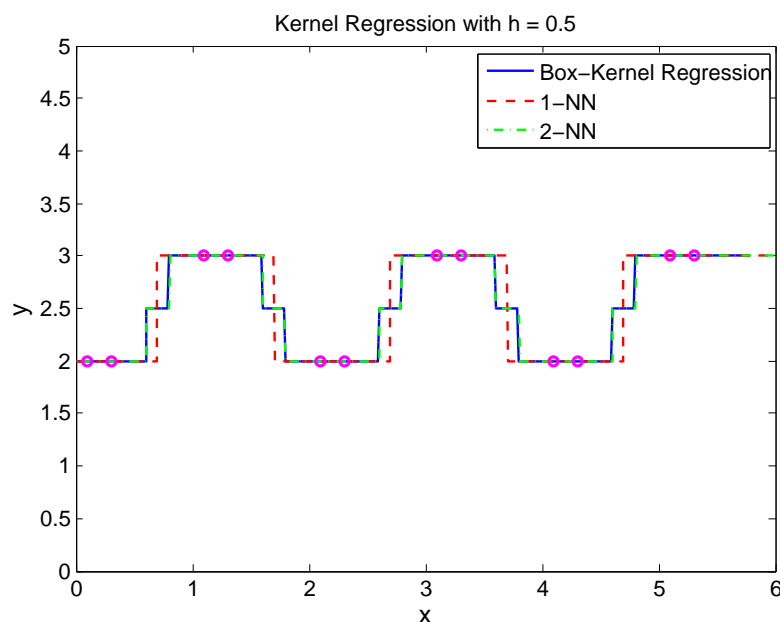
★ **SOLUTION:** As in Part 2, the basic idea is to create a dataset where in each interval of width 1 ( $2h$ ) there are 2 “training” points.

Here are some example solutions.

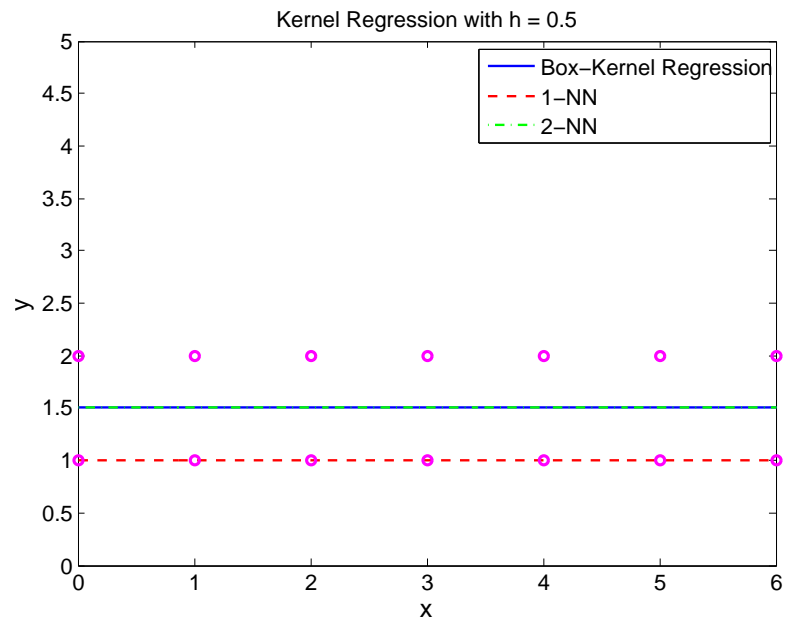
- Example 1



- 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](#).

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

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.

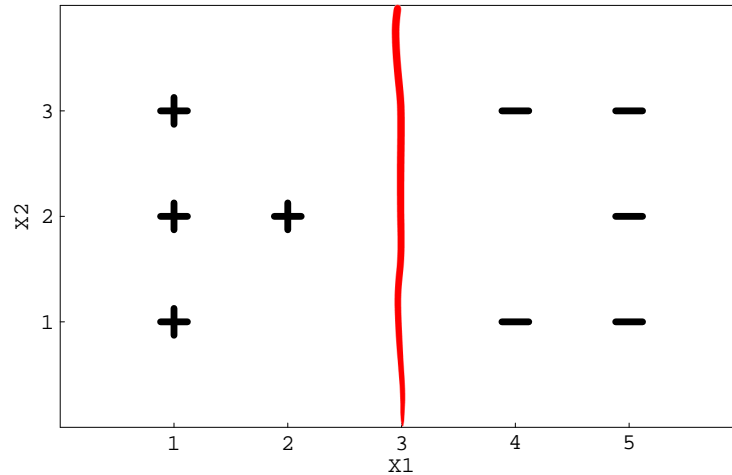
■ **COMMON MISTAKE 1:** Again, the signs are important here. Conceptually we want to minimize model complexity and maximize model fit (or likelihood). Since we always try to minimize the regularized loss we know that we want a negative sign on the likelihood and positive sign on the regularization term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

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

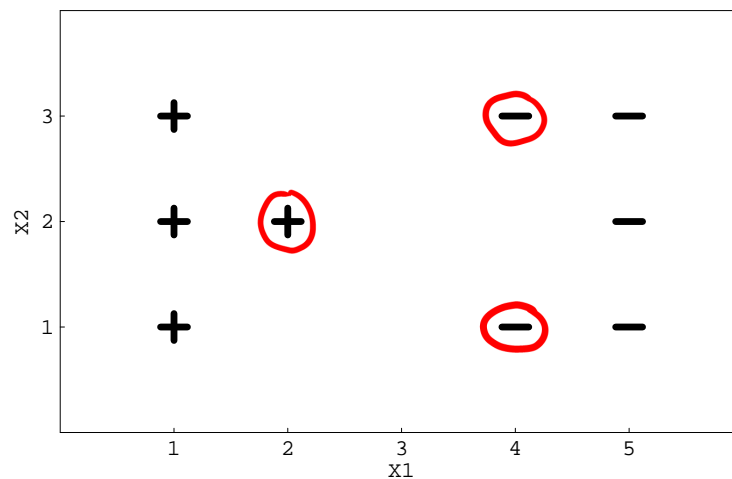
1. [2 points] Suppose we are using a linear SVM (i.e., no kernel), with some large  $C$  value, and are given the following data set.



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.



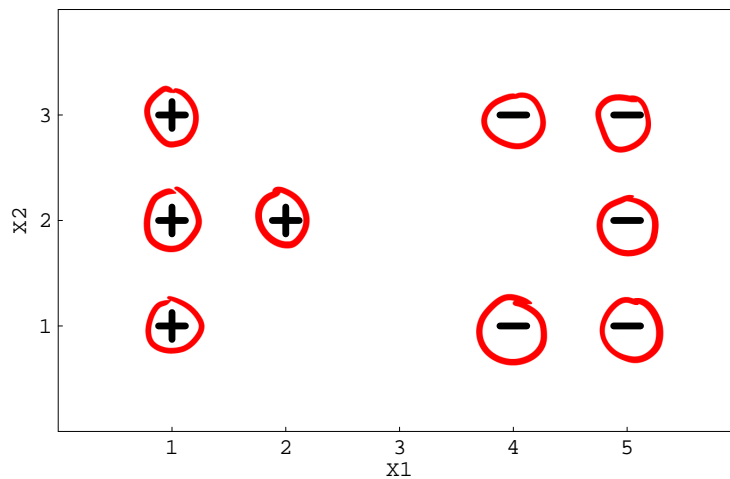
You do not need to provide a formal proof, but give a one or two sentence explanation.

★ **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.

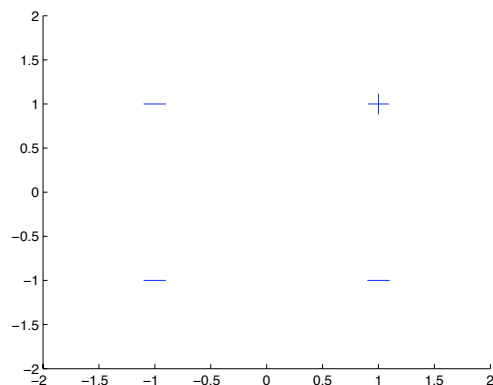
★ SOLUTION:

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

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

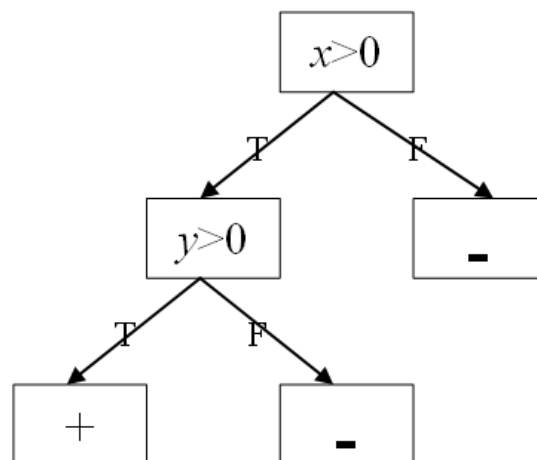
An ensemble classifier  $H_T(x)$  is a collection of  $T$  weak classifiers  $h_t(x)$ , each with some weight  $\alpha_t$ ,  $t = 1, \dots, T$ . Given a data point  $x \in \mathbb{R}^d$ ,  $H_T(x)$  predicts its label based on the weighted majority vote of the ensemble. In the binary case where the class 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;

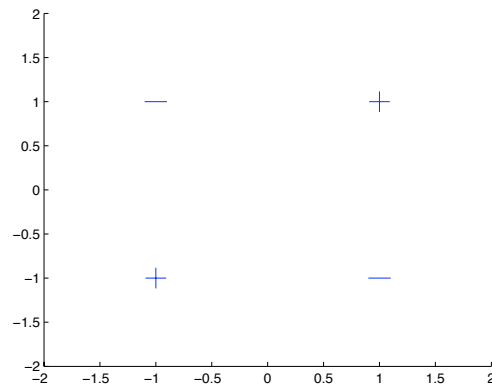
★ **SOLUTION:** The decision tree is as follows.



- 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.

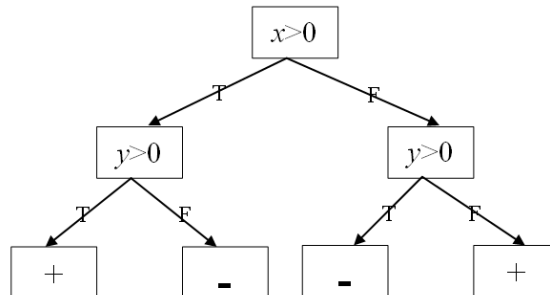
★ **SOLUTION:**  $h_1(x) = \text{sgn}(x), \alpha_1 = 1, h_2(x) = \text{sgn}(y), \alpha_2 = 1.$

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

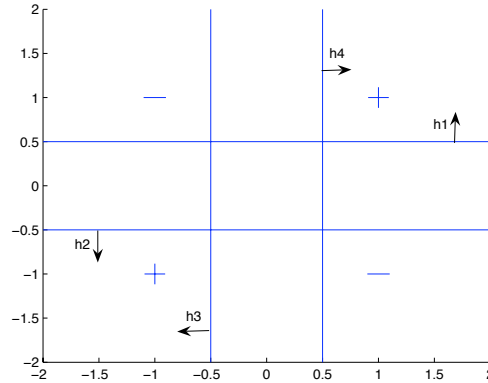


- Describe a binary decision tree with the minimum depth and consistent with the data;

★ **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.

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

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

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

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

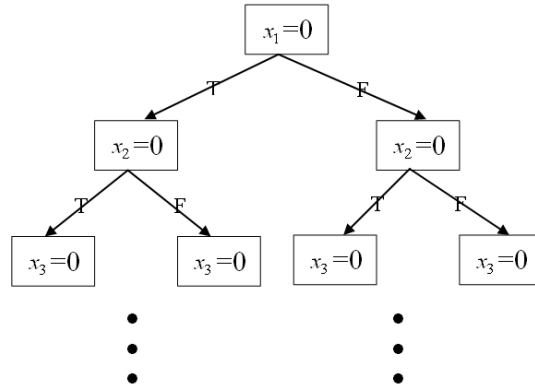
Apparently, the first and third equations can not be satisfied at the same time. Therefore, there are no weights  $\alpha_1, \dots, \alpha_4$ , that make the ensemble classifier consistent with the data.



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.  $h_i(x) = 2x_i - 1, \alpha_i = 1, i = 1, \dots, m$

# CSE 546 Midterm Exam, Fall 2014(with Solution)

1. Personal info:

- Name:
- UW NetID:
- Student ID:

2. There should be 14 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. You cannot use materials brought by other students.

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.

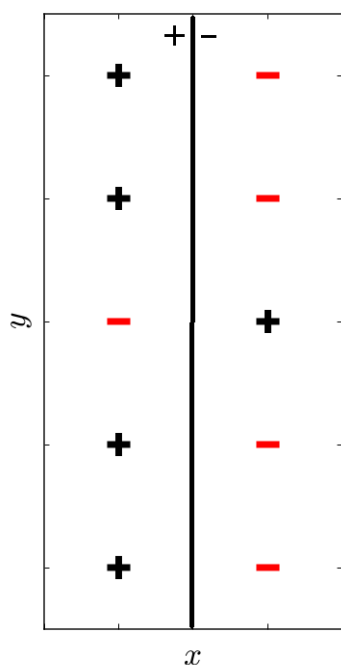
6. You have 80 minutes.

7. Good luck!

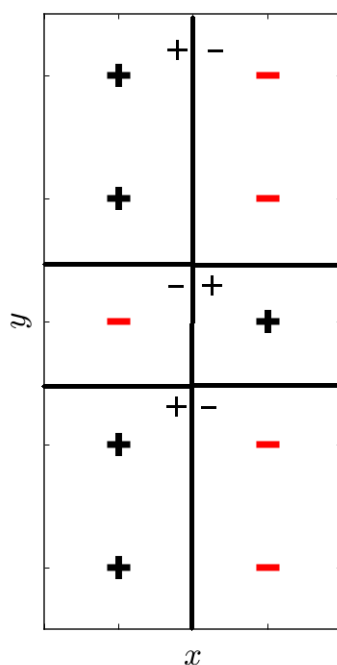
Question	Topic	Max score	Score
1	Short Answer	24	
2	Decision Trees	16	
3	Logistic Regression	20	
4	Boosting	16	
5	Elastic Net	24	
Total		100	

# 1 [24 points] Short Answer

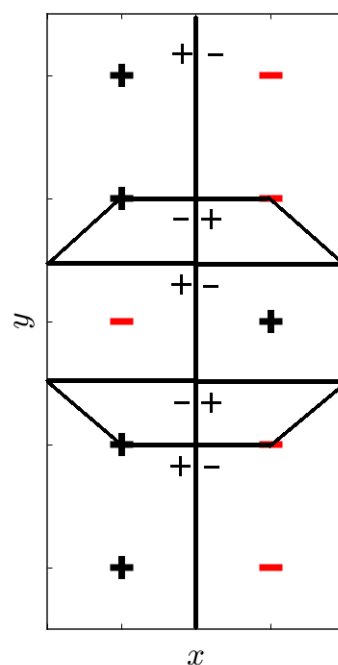
1. [6 points] On the 2D dataset below, draw the decision boundaries learned by the following algorithms (using the features  $x/y$ ). **Be sure to mark which regions are labeled positive or negative, and assume that ties are broken arbitrarily.**



(a) Logistic regression ( $\lambda = 0$ )



(b) 1-NN



(c) 3-NN

2. [6 points] A random variable follows an *exponential* distribution with parameter  $\lambda$  ( $\lambda > 0$ ) if it has the following density:

$$p(t) = \lambda e^{-\lambda t}, \quad t \in [0, \infty)$$

This distribution is often used to model waiting times between events. Imagine you are given i.i.d. data  $T = (t_1, \dots, t_n)$  where each  $t_i$  is modeled as being drawn from an exponential distribution with parameter  $\lambda$ .

- (a) [3 points] Compute the log-probability of  $T$  given  $\lambda$ . (Turn products into sums when possible).

★ ANSWER:

$$\begin{aligned}\ln p(T) &= \ln \prod_i p(t_i) \\ \ln p(T) &= \sum_i \ln(\lambda e^{-\lambda t_i}) \\ \ln p(T) &= \sum_i \ln \lambda - \lambda t_i \\ \ln p(T) &= \boxed{n \ln \lambda - \lambda \sum_i t_i}\end{aligned}$$

(b) [3 points] Solve for  $\hat{\lambda}_{MLE}$ .

★ ANSWER:

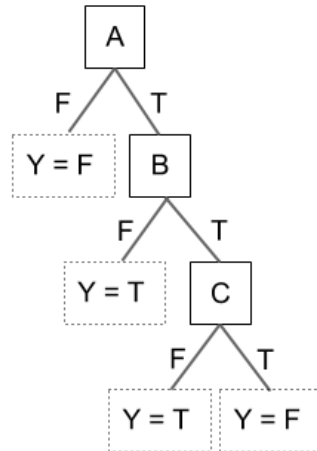
$$\begin{aligned}\frac{\partial}{\partial \lambda}(n \ln \lambda - \lambda \sum_i t_i) &= 0 \\ \frac{n}{\lambda} - \sum_i t_i &= 0 \\ \hat{\lambda}_{MLE} &= \boxed{\frac{n}{\sum_i t_i}}\end{aligned}$$

3. [6 points]

A	B	C	Y
F	F	F	F
T	F	T	T
T	T	F	T
T	T	T	F

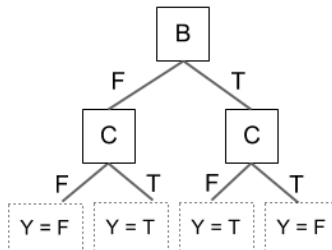
(a) [3 points] Using the dataset above, we want to build a decision tree which classifies  $Y$  as  $T/F$  given the binary variables  $A, B, C$ . Draw the tree that would be learned by the greedy algorithm with zero training error. You do not need to show any computation.

★ ANSWER:



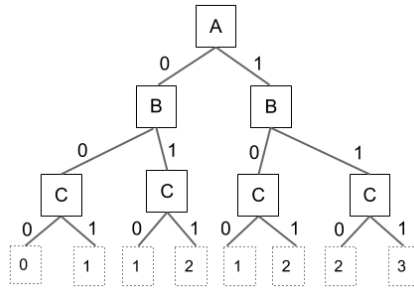
- (b) [3 points] Is this tree optimal (i.e. does it get zero training error with minimal depth)? Explain in less than two sentences. If it is not optimal, draw the optimal tree as well.

★ ANSWER: Although we get a better information gain by first splitting on  $A$ ,  $Y$  is just a function of  $B/C$ :  $Y = B \text{ xor } C$ . Thus, we can build a tree of depth 2 which classifies correctly, and is optimal.



4. [6 points] In class we used decision trees and ensemble methods for classification, but we can use them for regression as well (i.e. learning a function from features to real values). Let's imagine that our data has 3 binary features  $A, B, C$ , which take values 0/1, and we want to learn a function which counts the number of features which have value 1.
- (a) [2 points] Draw the decision tree which represents this function. How many leaf nodes does it have?

★ ANSWER:



We can represent the function with a decision tree containing 8 nodes.

- (b) [2 points] Now represent this function as a sum of decision stumps (e.g.  $\text{sgn}(A)$ ). How many terms do we need?

★ ANSWER:

$$f(x) = \text{sgn}(A) + \text{sgn}(B) + \text{sgn}(C)$$

Using a sum of decision stumps, we can represent this function using 3 terms.

- (c) [2 points] In the general case, imagine that we have  $d$  binary features, and we want to count the number of features with value 1. How many leaf nodes would a decision tree need to represent this function? If we used a sum of decision stumps, how many terms would be needed? No explanation is necessary.

★ ANSWER: Using a decision tree, we will need  $2^d$  nodes. Using a sum of decision stumps, we will need  $d$  terms.

## 2 [16 points] Decision Trees

We will use the dataset below to learn a decision tree which predicts if people pass machine learning (Yes or No), based on their previous GPA (High, Medium, or Low) and whether or not they studied.

GPA	Studied	Passed
L	F	F
L	T	T
M	F	F
M	T	T
H	F	T
H	T	T

For this problem, you can write your answers using  $\log_2$ , but it may be helpful to note that  $\log_2 3 \approx 1.6$ .

1. [4 points] What is the entropy  $H(\text{Passed})$ ?

★ ANSWER:

$$\begin{aligned}H(\text{Passed}) &= -\left(\frac{2}{6} \log_2 \frac{2}{6} + \frac{4}{6} \log_2 \frac{4}{6}\right) \\H(\text{Passed}) &= -\left(\frac{1}{3} \log_2 \frac{1}{3} + \frac{2}{3} \log_2 \frac{2}{3}\right) \\H(\text{Passed}) &= \boxed{\log_2 3 - \frac{2}{3} \approx 0.92}\end{aligned}$$

2. [4 points] What is the entropy  $H(\text{Passed} \mid \text{GPA})$ ?

★ ANSWER:

$$\begin{aligned}H(\text{Passed} \mid \text{GPA}) &= -\frac{1}{3} \left( \frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2} \right) - \frac{1}{3} \left( \frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2} \right) - \frac{1}{3} (1 \log_2 1) \\H(\text{Passed} \mid \text{GPA}) &= \frac{1}{3}(1) + \frac{1}{3}(1) + \frac{1}{3}(0) \\H(\text{Passed} \mid \text{GPA}) &= \boxed{\frac{2}{3} \approx 0.66}\end{aligned}$$

3. [4 points] What is the entropy  $H(\text{Passed} \mid \text{Studied})$ ?

★ ANSWER:

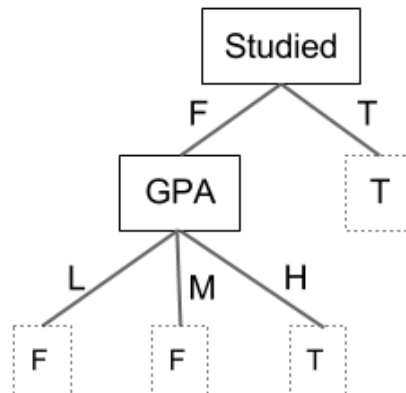
$$H(Passed|Studied) = -\frac{1}{2}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) - \frac{1}{2}(1\log_2 1)$$

$$H(Passed|Studied) = \frac{1}{2}\left(\log_2 3 - \frac{2}{3}\right)$$

$$H(Passed|Studied) = \frac{1}{2}\log_2 3 - \frac{1}{3} \approx 0.46$$

4. [4 points] Draw the full decision tree that would be learned for this dataset. You do not need to show any calculations.

★ ANSWER: We want to split first on the variable which maximizes the information gain  $H(Passed) - H(Passed|A)$ . This is equivalent to minimizing  $H(Passed|A)$ , so we should split on “Studied?” first.





### 3 [20 points] Logistic Regression for Sparse Data

In many real-world scenarios our data has millions of dimensions, but a given example has only hundreds of non-zero features. For example, in document analysis with word counts for features, our dictionary may have millions of words, but a given document has only hundreds of unique words. In this question we will make  $l_2$  regularized SGD efficient when our input data is sparse. Recall that in  $l_2$  regularized logistic regression, we want to maximize the following objective (**in this problem we have excluded  $w_0$  for simplicity**):

$$F(\mathbf{w}) = \frac{1}{N} \sum_{j=1}^N l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) - \frac{\lambda}{2} \sum_{i=1}^d \mathbf{w}_i^2$$

where  $l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w})$  is the logistic objective function

$$l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) = y^{(j)} \left( \sum_{i=1}^d \mathbf{w}_i \mathbf{x}_i^{(j)} \right) - \ln(1 + \exp(\sum_{i=1}^d \mathbf{w}_i \mathbf{x}_i^{(j)}))$$

and the remaining sum is our regularization penalty.

When we do stochastic gradient descent on point  $(\mathbf{x}^{(j)}, y^{(j)})$ , we are approximating the objective function as

$$F(\mathbf{w}) \approx l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) - \frac{\lambda}{2} \sum_{i=1}^d \mathbf{w}_i^2$$

**Definition of sparsity:** Assume that our input data has  $d$  features, i.e.  $\mathbf{x}^{(j)} \in \mathbb{R}^d$ . In this problem, we will consider the scenario where  $\mathbf{x}^{(j)}$  is sparse. Formally, let  $s$  be average number of nonzero elements in each example. We say the data is sparse when  $s \ll d$ . In the following questions, **your answer should take the sparsity of  $\mathbf{x}^{(j)}$  into consideration when possible**. **Note:** When we use a sparse data structure, we can iterate over the non-zero elements in  $O(s)$  time, whereas a dense data structure requires  $O(d)$  time.

1. [2 points] Let us first consider the case when  $\lambda = 0$ . Write down the SGD update rule for  $\mathbf{w}_i$  when  $\lambda = 0$ , using step size  $\eta$ , given the example  $(\mathbf{x}^{(j)}, y^{(j)})$ .

★ **ANSWER:** The update rule can be written as

$$\mathbf{w}_i^{(t+1)} \leftarrow \mathbf{w}_i^{(t)} + \eta \mathbf{x}_i^{(j)} \left( y^{(j)} - \frac{1}{1 + \exp(-\sum_k \mathbf{w}_k x_k^{(j)})} \right)$$

2. [4 points] If we use a dense data structure, what is the average time complexity to update  $\mathbf{w}_i$  when  $\lambda = 0$ ? What if we use a sparse data structure? Justify your answer in one or two sentences.

★ **ANSWER:** The time complexity to calculate  $\sum_k \mathbf{w}_k x_k^{(j)}$  is  $O(d)$  when the data structure is dense, and  $O(s)$  when the data structure is sparse. Note that even if we update  $\mathbf{w}_i$  for all  $i$ , we only need to calculate  $\sum_k \mathbf{w}_k x_k^{(j)}$  once, and then update the  $\mathbf{w}_i$  such that  $\mathbf{x}_i^{(j)} \neq 0$ . So the answer is  $\boxed{O(d)}$  for the dense case, and  $\boxed{O(s)}$  for the sparse case.

3. [2 points] Now let us consider the general case when  $\lambda > 0$ . Write down the SGD update rule for  $\mathbf{w}_i$  when  $\lambda > 0$ , using step size  $\eta$ , given the example  $(\mathbf{x}^{(j)}, y^{(j)})$ .

★ **ANSWER:**

$$\mathbf{w}_i^{(t+1)} \leftarrow \mathbf{w}_i^{(t)} - \eta \lambda \mathbf{w}_i^{(t)} + \eta \mathbf{x}_i^{(j)} \left( y^{(j)} - \frac{1}{1 + \exp(-\sum_k \mathbf{w}_k x_k^{(j)})} \right)$$

4. [2 points] If we use a dense data structure, what is the average time complexity to update  $\mathbf{w}_i$  when  $\lambda > 0$ ?

★ **ANSWER:** The time complexity is  $O(d)$

5. [4 points] Let  $\mathbf{w}_i^{(t)}$  be the weight vector after  $t$ -th update. Now imagine that we perform  $k$  SGD updates on  $\mathbf{w}$  using examples  $(\mathbf{x}^{(t+1)}, y^{(t+1)}), \dots, (\mathbf{x}^{(t+k)}, y^{(t+k)})$ , where  $\mathbf{x}_i^{(j)} = 0$  for every example in the sequence. (i.e. the  $i$ -th feature is zero for all of the examples in the sequence). Express the new weight,  $\mathbf{w}_i^{(t+k)}$  in terms of  $\mathbf{w}_i^{(t)}$ ,  $k$ ,  $\eta$ , and  $\lambda$ .

★ **ANSWER:** When  $\mathbf{x}_i^{(j)} = 0$ ,

$$\mathbf{w}_i^{(t+1)} = \mathbf{w}_i^{(t)} - \eta \lambda \mathbf{w}_i^{(t)} = \mathbf{w}_i^{(t)} (1 - \eta \lambda)$$

so the answer is

$$\mathbf{w}_i^{(t+k)} = \mathbf{w}_i^{(t)} (1 - \eta \lambda)^k$$

6. [6 points] Using your answer in the previous part, come up with an efficient algorithm for regularized SGD when we use a sparse data structure. What is the average time complexity per example? (Hint: when do you need to update  $\mathbf{w}_i$ ?)

---

**Algorithm 1:** Sparse SGD Algorithm for Logistic Regression with Regularization

---

```
Initialize  $c_i \leftarrow 0$  for  $i \in \{1, 2, \dots, d\}$ 
for  $j \in \{1, 2, \dots, n\}$  do
     $\hat{p} \leftarrow \frac{1}{1 + \exp(-\sum_k \mathbf{w}_k x_k^{(j)})}$ 
    for  $i$  such that  $\mathbf{x}_i^{(j)} \neq 0$  do
         $k \leftarrow j - c_i$ ; auxiliary variable  $c_i$  holds the index of last time we see  $\mathbf{x}_i^{(j)} \neq 0$ 
         $\mathbf{w}_i \leftarrow \mathbf{w}_i (1 - \eta \lambda)^k$ ; apply all the regularization updates
         $\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta \mathbf{x}_i^{(j)} (y^{(j)} - \hat{p})$ ; regularization is done in previous step
         $c_i \leftarrow j$ ; remember last time we see  $\mathbf{x}_i^{(j)} \neq 0$ 
    end
end
```

---

★ **ANSWER:** The idea is to only update  $\mathbf{w}_i$  when  $\mathbf{x}_i^{(j)} \neq 0$ . Before we do the update, we apply all the regularization updates we skipped before, using the answer from previous question. You can checkout Algorithm 1 for details. Using this trick, each update takes  $O(s)$  time. (Note: we can use the same trick applies for SGD with  $l_1$  regularization)

## 4 [16 points] Boosting

Recall that Adaboost learns a classifier  $H$  using a weighted sum of weak learners  $h_t$  as follows

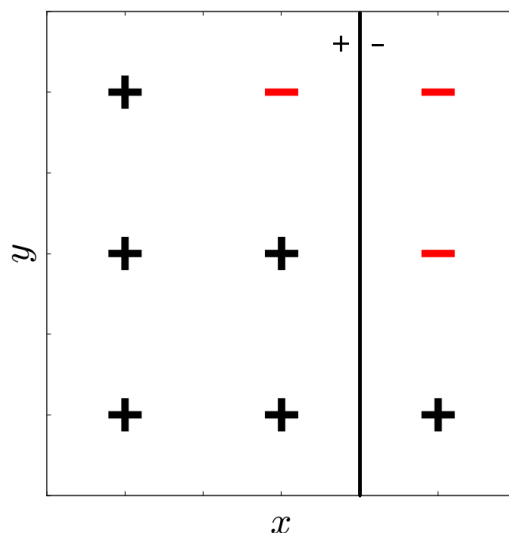
$$H(x) = \text{sgn} \left( \sum_{t=1}^T \alpha_t h_t(x) \right)$$

In this question we will use decision trees as our weak learners, which classify a point as  $\{1, -1\}$  based on a sequence of threshold splits on its features (here  $x, y$ ).

**In the questions below, be sure to mark which regions are marked positive/negative, and assume that ties are broken arbitrarily.**

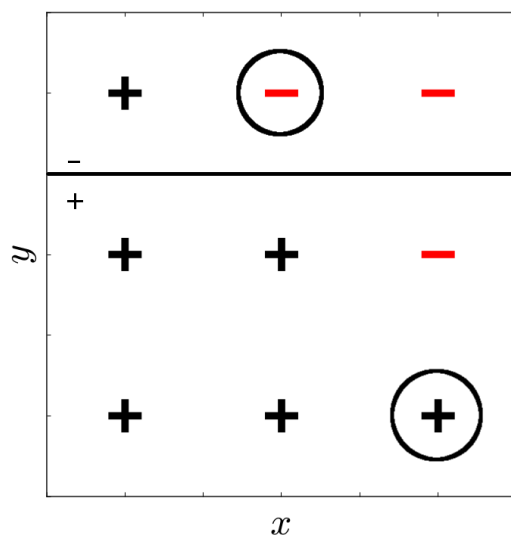
★ **COMMENT:** The solutions below are one of several possible answers. You got full credit as long as you were consistent and found a boundary with the lowest training error for  $h_1, h_2$ .

1. [2 points] Assume that our weak learners are decision trees of depth 1 (i.e. decision stumps), which minimize the weighted training error. Using the dataset below, draw the decision boundary learned by  $h_1$ .



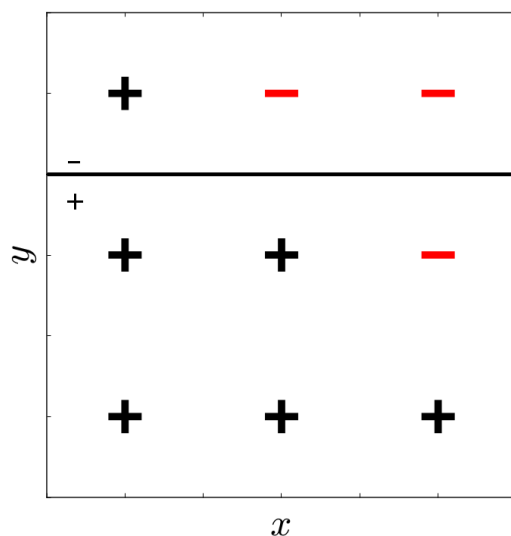
2. [3 points] On the dataset below, circle the point(s) with the highest weights on the second iteration, and draw the decision boundary learned by  $h_2$ .

★ **ANSWER:** The points with the highest weight will be those that were classified incorrectly by  $h_1$ .

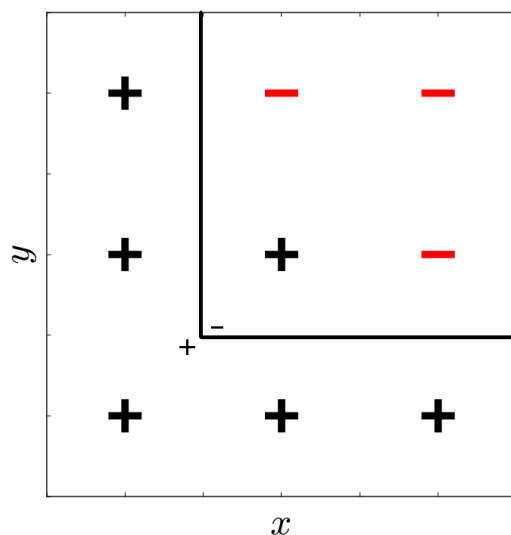


3. [3 points] On the dataset below, draw the decision boundary of  $H = \text{sgn}(\alpha_1 h_1 + \alpha_2 h_2)$ . (Hint, you do not need to explicitly compute the  $\alpha$ 's).

★ **ANSWER:** Although  $h_1$  and  $h_2$  misclassify the same number of points, the points which  $h_2$  gets wrong have been downweighted. Thus, we have  $\epsilon_1 > \epsilon_2$ , so  $\alpha_2 > \alpha_1$ , and  $h_2$  will dominate over  $h_1$  whenever there is a conflict. In other words,  $H$  has the same boundary as  $h_2$ .

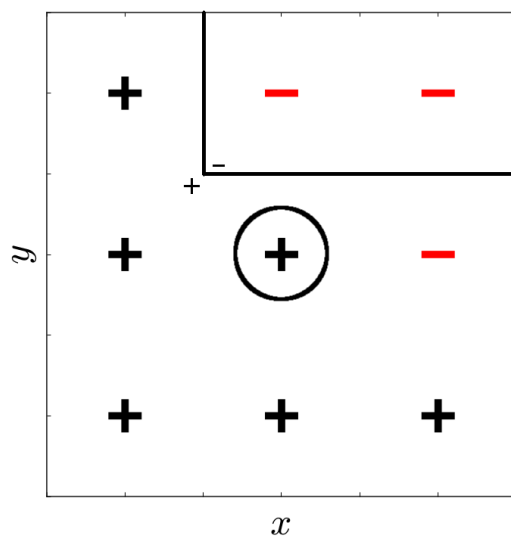


4. [2 points] Now assume that our weak learners are decision trees of maximum depth 2, which minimize the weighted training error. Using the dataset below, draw the decision boundary learned by  $h_1$ .



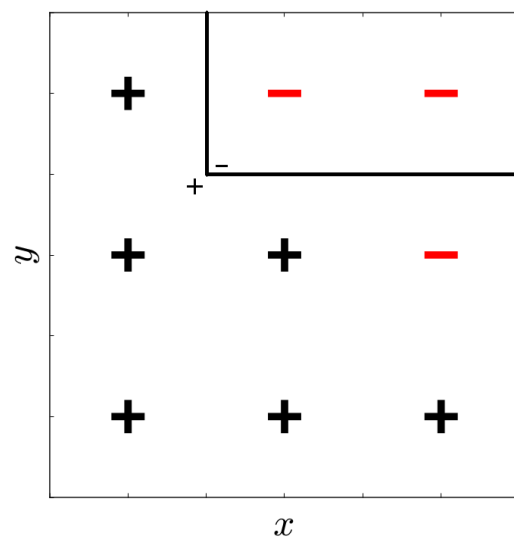
5. [3 points] On the dataset below, circle the point(s) with the highest weights on the second iteration, and draw the decision boundary learned by  $h_2$ .

★ **ANSWER:** Again, the points with the highest weight will be those that were classified incorrectly by  $h_1$ .



6. [3 points] On the dataset below, draw the decision boundary of  $H = \text{sgn}(\alpha_1 h_1 + \alpha_2 h_2)$ . (Hint, you do not need to explicitly compute the  $\alpha$ 's).

★ **ANSWER:** Again, we have  $\alpha_2 > \alpha_1$ , so  $h_2$  will dominate over  $h_1$  whenever there's a conflict.



## 5 [24 points] Elastic Net

In class we discussed two types of regularization,  $l_1$  and  $l_2$ . Both are useful, and sometimes it is helpful combine them, giving the objective function below (**in this problem we have excluded  $w_0$  for simplicity**):

$$F(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^n (y^{(j)} - \sum_{i=1}^d \mathbf{w}_i \mathbf{x}_i^{(j)})^2 + \alpha \sum_{i=1}^d |\mathbf{w}_i| + \frac{\lambda}{2} \sum_{i=1}^d \mathbf{w}_i^2 \quad (1)$$

Here  $(\mathbf{x}^{(j)}, y^{(j)})$  is  $j$ -th example in the training data,  $\mathbf{w}$  is a  $d$  dimensional weight vector,  $\lambda$  is a regularization parameter for the  $l_2$  norm of  $\mathbf{w}$ , and  $\alpha$  is a regularization parameter for the  $l_1$  norm of  $\mathbf{w}$ . This approach is called the Elastic Net, and you can see that it is a generalization of Ridge and Lasso regression: It reverts to Lasso when  $\lambda = 0$ , and it reverts to Ridge when  $\alpha = 0$ . In this question, we are going to derive the coordinate descent (CD) update rule for this objective.

Let  $g, h, c$  be real constants, and consider the function of  $x$

$$f_1(x) = c + gx + \frac{1}{2}hx^2 \quad (h > 0) \quad (2)$$

1. [2 points] What is the  $x^*$  that minimizes  $f_1(x)$ ? (i.e. calculate  $x^* = \arg \min f_1(x)$ )

★ **ANSWER:** Take the gradient of  $f_1(x)$  and set it to 0.

$$g + hx = 0$$

$$x^* = \boxed{-\frac{g}{h}}$$

Let  $\alpha$  be an additional real constant, and consider another function of  $x$

$$f_2(x) = c + gx + \frac{1}{2}hx^2 + \alpha|x| \quad (h > 0, \alpha > 0) \quad (3)$$

This is a piecewise function, composed of two quadratic functions:

$$f_2^-(x) = c + gx + \frac{1}{2}hx^2 - \alpha x$$

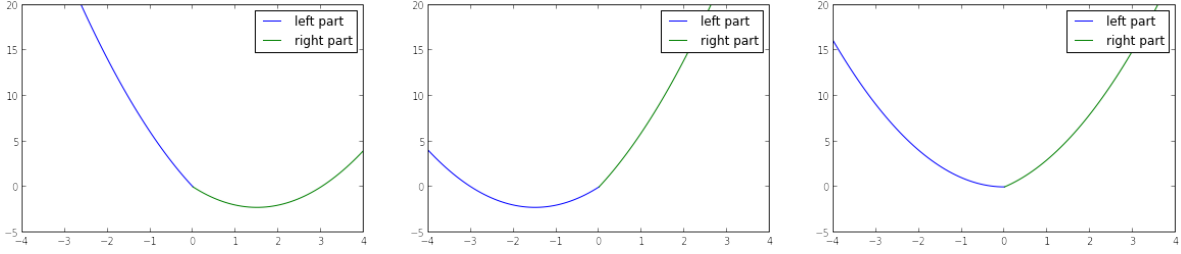
and

$$f_2^+(x) = c + gx + \frac{1}{2}hx^2 + \alpha x$$

Let  $\tilde{x}^- = \arg \min_{x \in \mathbb{R}} f_2^-(x)$  and  $\tilde{x}^+ = \arg \min_{x \in \mathbb{R}} f_2^+(x)$ .

(**Note:** The argmin is taken over  $(-\infty, +\infty)$  here)





(a)  $\tilde{x}^- > 0, \tilde{x}^+ > 0$ , the minima is at  $\tilde{x}^+$  (b)  $\tilde{x}^- < 0, \tilde{x}^+ < 0$ , the minima is at  $\tilde{x}^-$  (c)  $\tilde{x}^- > 0, \tilde{x}^+ < 0$ , the minima is at 0

Figure 1: Example picture of three cases

2. [3 points] What are  $\tilde{x}^+$  and  $\tilde{x}^-$ ? Show that  $\tilde{x}^- \geq \tilde{x}^+$ .

★ **ANSWER:** Using the answer from part 1), we get  $\tilde{x}^+ = -\frac{g^+\alpha}{h}$  and  $\tilde{x}^- = -\frac{g^-\alpha}{h}$ . Since  $\tilde{x}^- - \tilde{x}^+ = \frac{2\alpha}{h} \geq 0$ , we have  $\tilde{x}^- \geq \tilde{x}^+$ .

3. [6 points] Draw a picture of  $f_2(x)$  in each of the three cases below.

- (a)  $\tilde{x}^- > 0, \tilde{x}^+ > 0$
- (b)  $\tilde{x}^- < 0, \tilde{x}^+ < 0$
- (c)  $\tilde{x}^- > 0, \tilde{x}^+ < 0$ .

For each case, mark the minimum as either 0,  $\tilde{x}^-$ , or  $\tilde{x}^+$ . (You do not need to draw perfect curves, just get the rough shape and the relative locations of the minima to the  $x$ -axis)

★ **ANSWER:** An example of the curves is shown in Figure 1. To understand the answer, note the following:

- $f_2^+(0) = f_2^-(0)$ , this means the two piece of  $f_2$  match each other at  $x = 0$ .
- When  $\tilde{x}^- \geq 0$ , the minimum of left part is at 0, i.e.  $\arg \min_{x \in (-\infty, 0]} f_2^-(x) = 0$
- When  $\tilde{x}^- \leq 0$ , the minimum of left part is at  $\tilde{x}^-$ , i.e.  $\arg \min_{x \in (-\infty, 0]} f_2^-(x) = \tilde{x}^-$
- Similar rules holds for the right part of the curve.

4. [4 points] Write  $x^*$ , the minimum of  $f_2(x)$ , as a piecewise function of  $g$ . (Hint: what is  $g$  in the cases above?)

---

**Algorithm 2:** Coordinate Descent for Elastic Net

---

```
while not converged do
  for  $k \in \{1, 2, \dots, d\}$  do
     $g = \sum_{j=1}^n x_k^{(j)} (\sum_{i \neq k} \mathbf{w}_i \mathbf{x}_i^{(j)} - y_i)$ 
     $h = \lambda + \sum_{j=1}^n (\mathbf{x}_k^{(j)})^2$ 
     $w_k = \begin{cases} -\frac{g+\alpha}{h} & g < -\alpha \\ 0 & g \in [-\alpha, \alpha] \\ -\frac{g-\alpha}{h} & g > \alpha \end{cases}$ 
  end
end
```

---

★ **ANSWER:** Summarizing the result from the previous question, we have

$$x^* = \begin{cases} \tilde{x}^+ & \text{if } \tilde{x}^+ > 0 \\ \tilde{x}^- & \text{if } \tilde{x}^- < 0 \\ 0 & \text{if } \tilde{x}^+ < 0, \tilde{x}^- > 0 \end{cases}$$

This is equivalent to the soft-threshold function

$$x^* = \begin{cases} -\frac{g+\alpha}{h} & \text{if } g < -\alpha \\ 0 & \text{if } g \in [-\alpha, \alpha] \\ -\frac{g-\alpha}{h} & \text{if } g > \alpha \end{cases}$$

5. [4 points] Now let's derive the update rule for  $\mathbf{w}_k$ . Fixing all parameters but  $\mathbf{w}_k$ , express the objective function in the form of Eq (3) (here  $\mathbf{w}_k$  is our  $x$ ). You do not need to explicitly write constant factors, you can put them all together as  $c$ . What are  $g$  and  $h$ ?

$$g = \sum_{j=1}^n x_k^{(j)} (\sum_{i \neq k} \mathbf{w}_i \mathbf{x}_i^{(j)} - y_i) \quad (4)$$

$$h = \lambda + \sum_{j=1}^n (\mathbf{x}_k^{(j)})^2 \quad (5)$$

6. [5 points] Putting it all together, write down the coordinate descent algorithm for the Elastic Net (**again excluding**  $w_0$ ). You can write the update for  $\mathbf{w}_k$  in terms of the  $g$  and  $h$  you calculated in the previous question.

★ **ANSWER:** The algorithm is shown in Algorithm 2. You can compare it to the CD algorithm for Lasso, the only difference is adding  $\lambda$  to  $h$ .