A

Final Exam

GU4241/GR5241 Spring 2018

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 (3+3+4+8 points)

Short answers (about one sentence) are sufficient.

(a) What is the difference between bagging and random forests?

(b) Describe the procedure for computing the out of bag error from Bagging a regression tree.

(c) Assume that f is a convex function. $G = \{\mathbf{x} : g(\mathbf{x}) \leq 0\}$ is a convex set. For the constrained optimization problem:

$$\begin{array}{rl} \min & f(\mathbf{x}) \\ \text{subject to} & g(\mathbf{x}) \leq 0, \end{array}$$

the KKT conditions are:

$$\begin{aligned} \nabla f(\mathbf{x}) &= -\lambda \nabla g(\mathbf{x}) \\ \lambda g(\mathbf{x}) &= 0 \\ g(\mathbf{x}) &\leq 0 \\ \lambda &\geq 0 \end{aligned}$$

What is the purpose of each condition? Could you briefly explain the meaning of them?

Solution:

(a) In each case, we average the result of decision trees fit to many Bootstrap samples. However, in a Random Forest we restrict the number of variables to consider in each split. This produces a greater diversity of decision trees or *weak learners*, which reduces the variance of the averaged prediction.

(b) For each sample x_i, consider all the Bootstrap samples that do not contain x_i and average the predictions for the response y_i made by the corresponding trees; call the average $\hat{y}_i^{\text{oob}}$. Then, the out of bag error is given by:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i^{\text{oob}})^2.$$

(c) The first condition is the optimal criterion. If $\lambda = 0$, this is the criterion for achieving the minimum in the interior of G. If $\lambda > 0$, this means the directions of ∇f and ∇g are opposite, which is the optimal criterion for achieving the minimum at the bounday of G.

The second condition distinguishes the cases when the minimum is achieved in the interior and at the boundary of G.

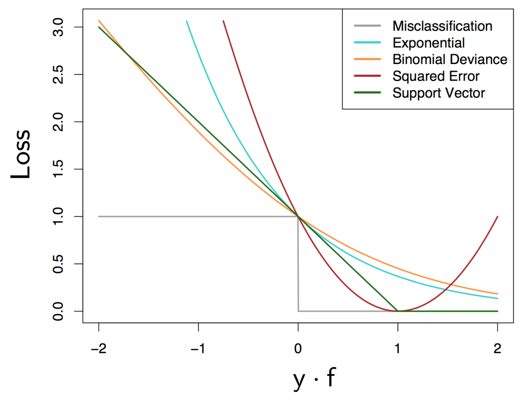
The third condition is the contraint of the problem.

The last one requires that ∇f cannot flip to orientation of ∇g.

(d) Assume that y is a binary response, $y \in \{-1, 1\}$. $f(x)$ is our classifier. The following are the loss functions that are frequently used in machine learning:

- 0-1 Loss: $L_{01}(y, f(x)) = \begin{cases} 0 & yf(x) > 0 \\ 1 & yf(x) \leq 0. \end{cases}$
- Hinge Loss (used in SVM): $L_{\mathsf{h}}(y, f(x)) = \max\{0, 1 - yf(x)\}$.
- Square Loss: $L_{\mathsf{sq}}(y, f(x)) = (1 - yf(x))^2$.
- Exponential Loss (used in boosting): $L_{\mathsf{exp}}(y, f(x)) = \exp(-yf(x))$.
- Binomial Deviance (used in logistic regression): $L_{\mathsf{bi}}(y, f(x)) = \log(1 + \exp(-yf(x)))$.

The plot of these loss functions are shown in the figure below:



- These loss functions can be thought of as convex approximation to the 0-1 loss function. Looking at the plot, which one appears intuitively to be the worst approximation to the 0-1 loss function? Which one appears to be the best?
- Consider just the 0-1 loss, the hinge loss, and the exponential loss. Rank the loss functions from highest to lowest in terms of robustness to misspecification of class labels in the data.
  Hint: consider the amount of loss assigned to a point z that is large and negative; this point corresponds to a large margin error (or a misspecification).

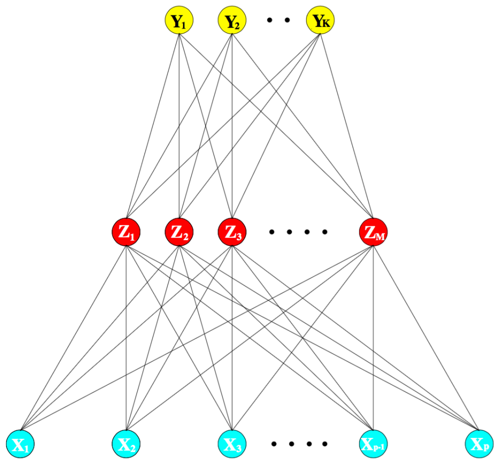
Solution:

(d)
- The squared error loss is the worst approximation, while the hinge loss is the best.
- Exponential loss is the least robust one, and 0-1 loss is the most robust one.

Problem 2: Neural Networks(10 points)

Assume that we fit a single layer hidden neural network in a regression problem on $\mathbb{R}^p$. Recall our model is:

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), \quad m = 1, \ldots, M. \\ f_k(X) &= \beta_{0k} + \beta_k^T Z, \quad k = 1, \ldots, K. \end{aligned}$$



The parameters of the model are $\theta = \{\alpha_{0m}, \alpha_m, \beta_{0k}, \beta_k\}$ (each α_m is a p-dimensional vector and β_k is an M-dimensional vector. We use gradient descent to minimize the squared error loss

$$R(\theta) = \sum_{i=1}^{n} \sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2.$$

A gradient update at the $(r+1)$st iteration has the form

$$\begin{aligned} \beta_{km}^{(r+1)} &= \beta_{km}^{(r)} - \frac{\partial R}{\partial \beta_{km}^{(r)}}, \\ \alpha_{ml}^{(r+1)} &= \alpha_{ml}^{(r)} - \frac{\partial R}{\partial \alpha_{ml}^{(r)}}. \end{aligned}$$

An issue of neural networks is that they have too many weights and will overfit the data. Therefore, regularization is necessary. Instead of minimizing the emprical risk $R(\theta)$, we add a penalty $J(\theta)$ to it with the form

$$J(\theta) = \sum_{k,m} \beta_{km}^2 + \sum_{m,l} \alpha_{ml}^2.$$

Now the object function of the optimization problem becomes

$$R(\theta) + \lambda J(\theta).$$

Write down the gradient update for this regularized problem. (You DO NOT NEED to calculate $\frac{\partial R}{\partial \beta_{km}}$ and $\frac{\partial R}{\partial \alpha_{ml}}$)

Solution:

$$\begin{aligned} \beta_{km}^{(r+1)} &= \beta_{km}^{(r)} - \left(\frac{\partial R}{\partial \beta_{km}^{(r)}} + 2\lambda \beta_{km}^{(r)} \right), \\ \alpha_{ml}^{(r+1)} &= \alpha_{ml}^{(r)} - \left(\frac{\partial R}{\partial \alpha_{ml}^{(r)}} + 2\lambda \alpha_{ml}^{(r)} \right). \end{aligned}$$

Problem 3: Classification Trees(10 points)

We have some data about when people go hiking. The data take into effect, whether hike is on a weekend or not, if the weather is rainy or sunny, and if the person will have company during the hike. Find the optimum decision tree for hiking habits, using the training data below. When you split the decision tree at each node, maximize the drop of impurity by using the entropy as the impurity measure, i.e.,

$$\text{maximize} \quad [I(D) - (I(D_L) + I(D_R))]$$

where D is parent node, and D_L and D_R are two child nodes, and $I(D)$ is:

$$I(D) = mH(\frac{m^+}{m}) = mH(\frac{m^-}{m})$$

and $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$, $0 \le x \le 1$, is the entropy function and $m = m^+ + m^-$ with m^+ and m^- being the total number of positive and negative cases at the node.

You may find the following useful in your calculations: $H(x) = H(1-x)$, $H(0) = 0$, $H(1/5) = 0.72$, $H(1/4) = 0.8$, $H(1/3) = 0.92$, $H(2/5) = 0.97$, $H(3/7) = 0.99$, $H(0.5) = 1$.

Weekend?	Company?	Weather	Go Hiking ?
Y	N	R	N
Y	Y	R	N
Y	Y	R	Y
Y	Y	S	Y
Y	N	S	Y
Y	N	S	N
Y	Y	R	N
Y	Y	S	Y
N	Y	S	N
N	Y	R	N
N	N	S	N

(a) Build a decision tree of depth 3. Draw your decision tree.

(b) According to your decision tree, what is the probability of going to hike on a rainy week day, without any company?

(c) How about the probability of going to hike on a rainy weekend when having some company?

Solution:

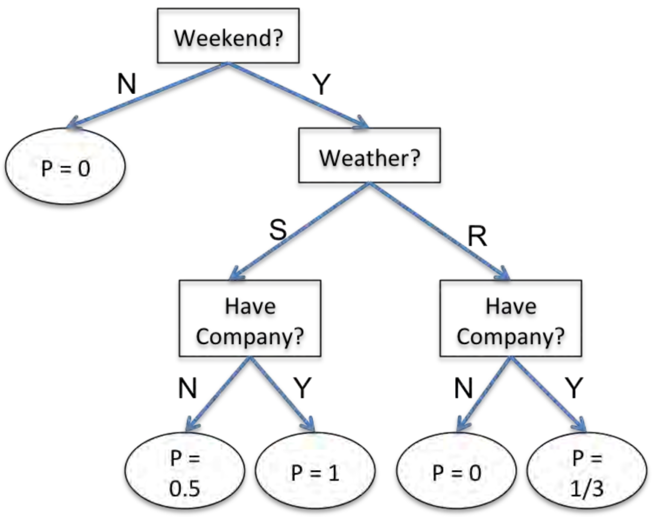
(a) We ant to choose attributes that maximize $mH(p) - m_lH(p_l) - m_r(H(p_r)$, where p, p_l, and p_r are the fraction of positive cases in parent nodes and two child nodes, and m, m_l, and m_r are the total number of points in parent node and two child nodes. This means that at each step, we need to choose the predictor for which $m_rH(p_r) + m_lH(p_l)$ is minimum. For the first step, the predictor *Weekend* achieve this:

- *Weekend*: $m_rH(p_r) + m_lH(p_l) = 8H(1/2) + 3H(0) = 8$
- *Weather*: $m_rH(p_r) + m_lH(p_l) = 5H(1/5) + 6H(1/2) \approx 9.6$
- *Company*: $m_rH(p_r) + m_lH(p_l) = 4H(1/4) + 7H(3/7) \approx 10.1$

Therefore the first split is on *Weekend*. If *Weekend* = N, then the probability of going hiking is 0. If *weekend* = Y, we need to choose the second predictor to split on:

- *Weather*: $m_rH(p_r) + m_lH(p_l) = 4H(1/4) + 4H(1/4) \approx 6.4$
- *Company*: $m_rH(p_r) + m_lH(p_l) = 5H(2/5) + 3H(1/3) \approx 7.6$

Therefore, the second split will be on *Weather*, and the third one will be *Company*. The decision tree is as follows:

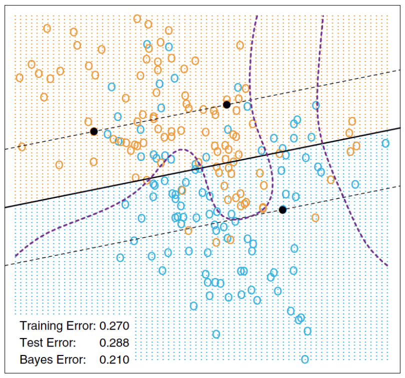


(b) Based on the decision tree, the probability is 0.

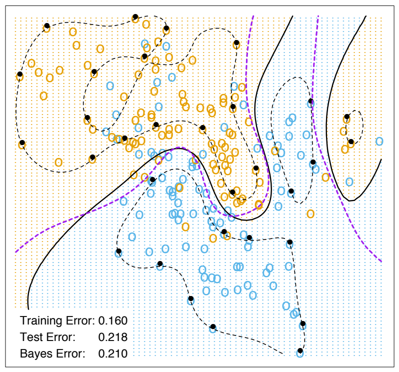
(c) The probability is 1/3.

Problem 4: Decision boundaries (10 points)

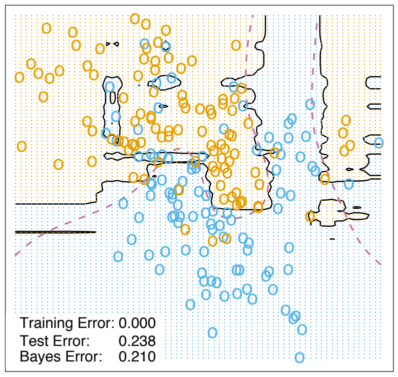
The following pictures, which we have all seen in class, show the output of several different classifiers. Recall that the thick line is the decision boundary determined by the classifier; you can ignore the dashed lines.



(a)



(b)



(c)

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

Problem 5: Decision Trees(10 points)

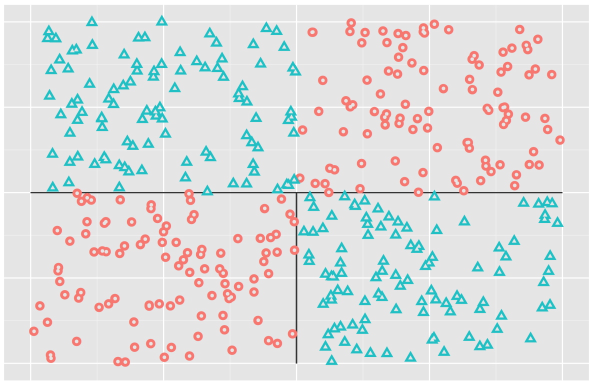
The standard method for fitting a decision tree involves:

- Growing the tree split by split. We maximize the reduction of the training error at each step until there are at most 5 samples per region.
- Pruning the tree to obtain a sequence of trees of decreasing size.
- Selecting the optimal size by cross-validation.

Consider the following alternative approach. Grow the tree split by split until the reduction in the training error produced by the next split is smaller than some threshold. This approach may lead to bad results because it is possible to make a split which does not decrease the error by much, and then make a second split which reduces the error significantly.

Draw an example dataset where this happens with two predictors X_1 and X_2, and a binary categorical response.

Solution: The figure shows the partition produced by a tree with 2 splits. The first split (horizontal line) barely reduces the classification error. However, the second split (vertical line) decreases the error significantly.



Problem 6: The Kernel Trick(10 points)

In this problem, we will apply the kernel trick to ridge regression and derive kernel ridge regression.

Consider a linear regression problem with n data points each in p dimensions, corresponding to the data matrix $\mathbf{X} \in \mathbb{R}^{n\times(p+1)}$ and response vector $\mathbf{y} \in \mathbb{R}^n$. Just as we extended a linear SVM to a nonlinear SVM with the kernel trick, we can do the same to create nonlinear kernel regression. Specifically, we want to find the solution to:

$$\arg\min_{\beta} \sum_{i=1}^{n} (y_i - \langle \beta, \phi(x_i)\rangle_{\mathcal{F}})^2 + \lambda \|\beta\|_{\mathcal{F}}^2,$$

where $\phi(\cdot)$ is the feature map and $\langle \cdot, \cdot \rangle_{\mathcal{F}} = k(\cdot,\cdot)$ in the usual "kernel trick" way.

A very important property of the solution is that β can be written in the form $\beta = \sum_{i=1}^{n} \alpha_i \phi(x_i) = \Phi^T \alpha$ with $\Phi = [\phi(x_1)\cdots\phi(x_n)]^T$ (this general fact in often called the representation theorem).

Then, using this property, write how to predict $\hat{f}(x^*)$ and a new point x^*. Your prediction $f(x^*)$ should be in terms of the kernel matrix $\mathbf{K} = \{\langle \phi(x_i), \phi(x_j)\rangle_{\mathcal{F}}\}_{i,j=1,\ldots,n}$, and elements in $\mathcal{F}$ should not appeaer explicitly in this prediction (since that could be infinite dimensional).

Solution: Assume that $\Phi = [\phi(x_1), \ldots, \phi(x_n)]^T$ is the feature matrix, where ϕ is the feature map.

$$\begin{aligned}
\sum_{i=1}^{n} (y_i - \langle \beta, \phi(x_i)\rangle_{\mathcal{F}})^2 + \lambda \|\beta\|_{\mathcal{F}}^2 &= \|y - \langle \beta, \Phi^T\rangle\|_2^2 + \lambda\|\beta\|_{\mathcal{F}}^2 \\
&= \|y - \langle \Phi^T\alpha, \Phi^T\rangle\|_2^2 + \lambda\|\Phi^T\alpha\|_{\mathcal{F}}^2 \\
&= y^T y - 2\alpha^T\Phi\Phi^T y + \alpha^T\Phi\Phi^T\Phi\Phi^T\alpha + \alpha^T\Phi\Phi^T\alpha \\
&= y^T y - 2\alpha^T K y + \alpha^T(K^2 + \lambda K)\alpha
\end{aligned}$$

Taking derivative, we have

$$2Ky = 2K(K + \lambda I)\alpha.$$

Therefore, the solution is

$$\alpha_{\text{KRR}} = (K + \lambda I)^{-1} y,$$

which has the same form as the ridge regression solution in the original space. Then we have $\beta_{\text{KRR}} = \Phi^T \alpha_{\text{KRR}}$, and thus:

$$\begin{aligned}
f(x^*) &= \langle \beta, \phi(x^*)\rangle_{\mathcal{F}} \\
&= \langle \Phi^T\alpha, \phi(x^*)\rangle_{\mathcal{F}} \\
&= \begin{bmatrix} k(x^*, x_1) \\ \vdots \\ k(x^*, x_n) \end{bmatrix} (K + \lambda I)^{-1} y.
\end{aligned}$$

Problem 7: Missing Data(10 points)

Assume you are given a data set consisting of variables having more than 30% missing values? Let's say, out of 50 variables, 8 variables have missing values higher than 30%. How will you deal with them? List at least two approaches.

Solution:

(a) We can replace the missing values by the mean of each variable.

(b) We can impute the missing values by running a regression on the other varibles.

(c) We can view this as a matrix completion problem, and use the low rank approximation as our data matrix.