DS-GA 1003 / CSCI-GA 2567: Machine Learning

May 15, 2018: Final Exam (110 Minutes)

Answer the questions in the spaces provided. If you run out of room for an answer, use the blank page at the end of the test. Please **don't miss the last question**, on the back of the last test page.

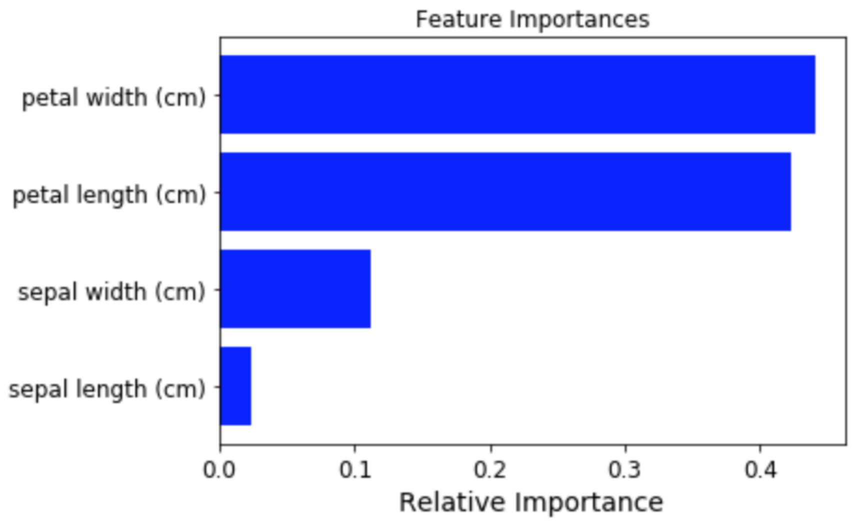
Name: ______________________________

NYU NetID: ______________________________

Question	Points	Score
Feature importance	3	
Decision Trees	8	
Ensemble Methods	9	
Bayesian Bernoulli Model	8	
EM Algorithm and Latent Variable Models	6	
Multiclass	4	
Conditional Probability Models: Beta Distribution	5	
Neural Networks	6	
Total:	49	

1. **Feature importance**:

Consider the feature importance chart shown below (for the iris classification problem):



(a) (1 point) Assume the "relative importance" reported on the x-axis is permutation importance, using classification accuracy as a metric. For which feature did permutation lead to the largest decrease in test set accuracy?

■ **petal width** ☐ petal length ☐ sepal width ☐ sepal length

(b) (1 point) Now assume "relative importance" is mean decrease impurity for a decision tree classifier. Consider the feature "sepal length (cm)". Which of the following is the least likely explanation for its low relative importance?

☐ Sepal length does not provide additional information about the target, given the other features.

☐ The target variable is independent of sepal length.

■ **Sepal length (cm) was used as the splitting variable at the root node of the tree.**

(c) (1 point) <u>**T**</u> **True or False**: Given this feature importance chart, there could be another prediction function that has similar accuracy to the given prediction function but for which "sepal length (cm)" has higher relative importance than "sepal width (cm)".

2. **Decision Trees**:
Below we give the design matrix X and response vector y for the training set of a binary classification problem with real-valued features:

$$X = \begin{bmatrix} 4 & 1 \\ 6 & 6 \\ 9 & 5 \\ 1 & 2 \\ 7 & 3 \\ 5 & 4 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

In this question, we will build a soft classification binary tree using entropy as our impurity measure. As a reminder:

The **entropy** of a probability mass function $p(x)$ on a discrete set $\mathcal{X} = \{x_1, \dots, x_k\}$ is given by

$$H(p) = -\sum_{i=1}^{k} p(x_i) \log p(x_i),$$

where we use the convention that $0 \log 0 = 0$.

(a) (2 points) Indicate which of the following would be expected to reduce overfitting in a decision tree. (Answer in general – not for the data specified in this problem.)

- ■ **Pruning the decision tree.**
- ☐ Increasing maximum tree depth.
- ■ **Decreasing the maximum number of leaf nodes.**
- ■ **Increasing the minimum number of training instances required in a leaf node.**

(b) (2 points) For the data given above, what is the entropy at the root node of the tree before any splitting has occurred?

Solution:

$$-0.5 \log 0.5 - 0.5 \log 0.5 = \log 2$$

Can either leave *log* in the answer or assume $\log_2$, in which case the answer is 1.

(c) (2 points) As we consider splitting the root node, our plan is to search over all possible splits s for the split that **minimizes** a real-valued function $f(s)$. Give an expression for the function f that corresponds to splitting using the entropy impurity measure. Please write $f(s)$ in terms of the functions $n_{0L}(s)$, $n_{1L}(s)$, $n_{0R}(s)$, and $n_{1R}(s)$, where the first two functions give the number of examples of class 0 and 1 on the left side of split s, and where the second two functions give the number of examples of class 0 and 1 on the right side of the split. For convenience, you may also use the functions $n_L(s) = n_{0L}(s) + n_{1L}(s)$ and $n_R(s) = n_{0R}(s) + n_{1R}(s)$, and you may define other intermediate functions if you wish.

Solution: Define $h(a) = -a \log a - (1-a) \log (1-a)$. Then let

$$f(s) = n_L(s) h\left(\frac{n_{0L}(s)}{n_L(s)}\right) + n_R(s) h\left(\frac{n_{0R}(s)}{n_R(s)}\right)$$

.

Also fine, possibly even more standard, to divide the whole expression by $n_L(s) + n_R(s)$.

(d) (2 points) For the data given above, we are looking for a partition of the root node based on the first feature (i.e. the first column of X) using a rule of the form $x_1 \geq c$. Select **ALL** of the following rules that are optimal with respect to the entropy impurity measure. [HINT: This problem may be solved without calculation, but may require scratch space.] (Note: No partial credit for this problem.)

☐ $x_1 \geq 5$ ☐ $x_1 \geq 5.5$ ☐ $x_1 \geq 6$ [✓ $x_1 \geq 6.5$] [✓ $x_1 \geq 7$]

3. **Ensemble Methods**:

   (a) Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

      i. (1 point) __**T**__ **True or False**: If your random forest is overfitting, using more bootstrap samples (i.e. more trees) could help.

      ii. (1 point) __**F**__ **True or False**: If your gradient boosting model is overfitting, taking additional steps is likely to help.

      iii. (1 point) __**F**__ **True or False**: In gradient boosting, if you reduce your step size, you should expect to need fewer rounds of boosting (i.e. fewer steps) to achieve the same training set loss.

      iv. (1 point) __**T**__ **True or False**: Fitting a random forest model is extremely easy to parallelize.

      v. (1 point) __**F**__ **True or False**: Fitting a gradient boosting model is extremely easy to parallelize, for any base regression algorithm.

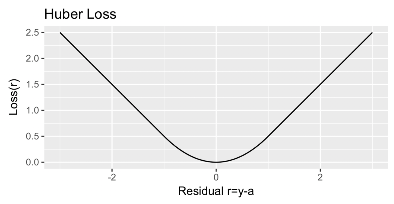
      vi. (1 point) __**T**__ **True or False**: Suppose we apply gradient boosting with absolute loss to a regression problem. If we use linear ridge regression as our base regression algorithm, the final prediction function from gradient boosting always will be an affine function of the input.

(b) (3 points) Consider the dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$ where $\mathcal{X} = \mathcal{Y} = \mathbb{R}$ and

$$(x_1, y_1) = (0, 0.5), \quad (x_2, y_2) = (3, -1), \quad (x_3, y_3) = (1, 2).$$

Our goal is to use gradient boosting and a software package that fits small regression trees to build our prediction function. We expect our data to have some outliers, so we use the following Huber regression loss:

$$\ell(y, a) = \begin{cases} y - a - \frac{1}{2} & y - a > 1 \\ -(y - a) - \frac{1}{2} & y - a < -1 \\ \frac{1}{2}(y - a)^2 & \text{otherwise.} \end{cases}$$



Suppose we start at $f_0 \equiv 0$, and we want to compute f_1, the prediction function after one round of gradient boosting. Give the dataset that will be passed into the black box regression tree algorithm to compute f_1.

Solution: Since

$$\frac{\partial \ell(y, a)}{\partial a} = \begin{cases} -1 & y - a > 1 \\ 1 & y - a < -1 \\ -(y - a) & \text{otherwise.} \end{cases},$$

For $a = 0$, we have

$$\frac{\partial \ell(y, 0)}{\partial a} = \begin{cases} -1 & y > 1 \\ 1 & y < -1 \\ -y & \text{otherwise.} \end{cases}$$

So the data set is

$$(x_1, y_1) = (0, -0.5), \quad (x_2, y_2) = (3, 1), \quad (x_3, y_3) = (1, -1).$$

Also fine to fit the regression to the negative gradient, and step in the opposite direction, in which case the answer is So the data set is

$$(x_1, y_1) = (0, 0.5), \quad (x_2, y_2) = (3, -1), \quad (x_3, y_3) = (1, 1).$$

DS-GA 1003 / CSCI-GA 2567: Machine Learning

May 15, 2018: Final Exam (110 Minutes)

Answer the questions in the spaces provided. If you run out of room for an answer, use the blank page at the end of the test. Please **don't miss the last question**, on the back of the last test page.

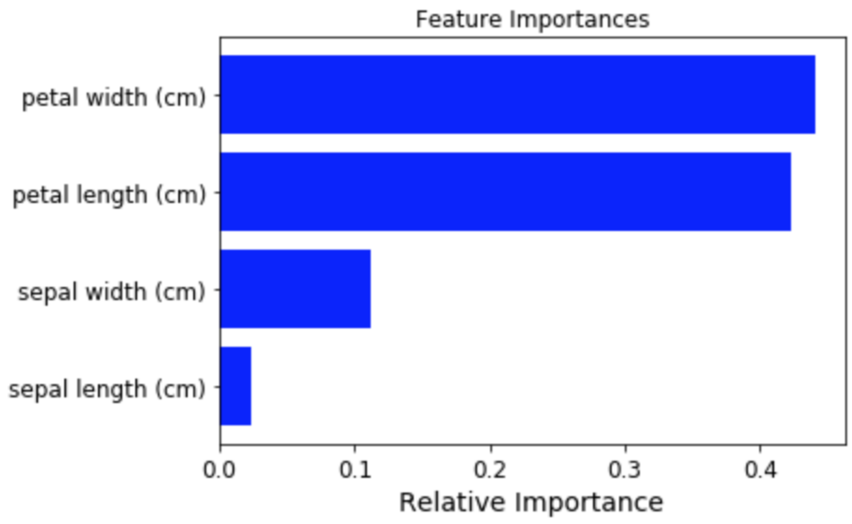
Name: ______________________________

NYU NetID: ______________________________

Question	Points	Score
Feature importance	3	
Decision Trees	8	
Ensemble Methods	9	
Bayesian Bernoulli Model	8	
EM Algorithm and Latent Variable Models	6	
Multiclass	4	
Conditional Probability Models: Beta Distribution	5	
Neural Networks	6	
Total:	49	

1. **Feature importance**:

Consider the feature importance chart shown below (for the iris classification problem):



(a) (1 point) Assume the "relative importance" reported on the x-axis is permutation importance, using classification accuracy as a metric. For which feature did permutation lead to the largest decrease in test set accuracy?

■ **petal width** ☐ petal length ☐ sepal width ☐ sepal length

(b) (1 point) Now assume "relative importance" is mean decrease impurity for a decision tree classifier. Consider the feature "sepal length (cm)". Which of the following is the least likely explanation for its low relative importance?

☐ Sepal length does not provide additional information about the target, given the other features.

☐ The target variable is independent of sepal length.

■ **Sepal length (cm) was used as the splitting variable at the root node of the tree.**

(c) (1 point) <u>T</u> **True or False**: Given this feature importance chart, there could be another prediction function that has similar accuracy to the given prediction function but for which "sepal length (cm)" has higher relative importance than "sepal width (cm)".

2. **Decision Trees**:
Below we give the design matrix X and response vector y for the training set of a binary classification problem with real-valued features:

$$X = \begin{bmatrix} 4 & 1 \\ 6 & 6 \\ 9 & 5 \\ 1 & 2 \\ 7 & 3 \\ 5 & 4 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

In this question, we will build a soft classification binary tree using entropy as our impurity measure. As a reminder:

The **entropy** of a probability mass function $p(x)$ on a discrete set $\mathcal{X} = \{x_1, \ldots, x_k\}$ is given by

$$H(p) = -\sum_{i=1}^{k} p(x_i) \log p(x_i),$$

where we use the convention that $0 \log 0 = 0$.

(a) (2 points) Indicate which of the following would be expected to reduce overfitting in a decision tree. (Answer in general – not for the data specified in this problem.)

■ **Pruning the decision tree.**
☐ Increasing maximum tree depth.
■ **Decreasing the maximum number of leaf nodes.**
■ **Increasing the minimum number of training instances required in a leaf node.**

(b) (2 points) For the data given above, what is the entropy at the root node of the tree before any splitting has occurred?

Solution:

$$-0.5 \log 0.5 - 0.5 \log 0.5 = \log 2$$

Can either leave *log* in the answer or assume $\log_2$, in which case the answer is 1.

(c) (2 points) As we consider splitting the root node, our plan is to search over all possible splits s for the split that **minimizes** a real-valued function $f(s)$. Give an expression for the function f that corresponds to splitting using the entropy impurity measure. Please write $f(s)$ in terms of the functions $n_{0L}(s)$, $n_{1L}(s)$, $n_{0R}(s)$, and $n_{1R}(s)$, where the first two functions give the number of examples of class 0 and 1 on the left side of split s, and where the second two functions give the number of examples of class 0 and 1 on the right side of the split. For convenience, you may also use the functions $n_L(s) = n_{0L}(s) + n_{1L}(s)$ and $n_R(s) = n_{0R}(s) + n_{1R}(s)$, and you may define other intermediate functions if you wish.

Solution: Define $h(a) = -a \log a - (1-a) \log (1-a)$. Then let

$$f(s) = n_L(s) h\left(\frac{n_{0L}(s)}{n_L(s)}\right) + n_R(s) h\left(\frac{n_{0R}(s)}{n_R(s)}\right)$$

.

Also fine, possibly even more standard, to divide the whole expression by $n_L(s) + n_R(s)$.

(d) (2 points) For the data given above, we are looking for a partition of the root node based on the first feature (i.e. the first column of X) using a rule of the form $x_1 \geq c$. Select **ALL** of the following rules that are optimal with respect to the entropy impurity measure. [HINT: This problem may be solved without calculation, but may require scratch space.] (Note: No partial credit for this problem.)

☐ $x_1 \geq 5$ ☐ $x_1 \geq 5.5$ ☐ $x_1 \geq 6$ [✓ $x_1 \geq 6.5$] [✓ $x_1 \geq 7$]

3. **Ensemble Methods**:

(a) Indicate whether each of the statements (about random forests and gradient boosting) is true or false.

i. (1 point) __**T**__ **True or False**: If your random forest is overfitting, using more bootstrap samples (i.e. more trees) could help.

ii. (1 point) __**F**__ **True or False**: If your gradient boosting model is overfitting, taking additional steps is likely to help.

iii. (1 point) __**F**__ **True or False**: In gradient boosting, if you reduce your step size, you should expect to need fewer rounds of boosting (i.e. fewer steps) to achieve the same training set loss.

iv. (1 point) __**T**__ **True or False**: Fitting a random forest model is extremely easy to parallelize.

v. (1 point) __**F**__ **True or False**: Fitting a gradient boosting model is extremely easy to parallelize, for any base regression algorithm.

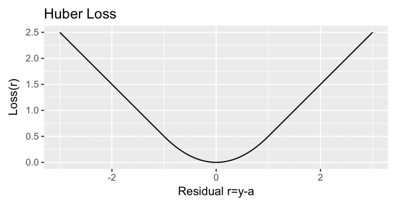
vi. (1 point) __**T**__ **True or False**: Suppose we apply gradient boosting with absolute loss to a regression problem. If we use linear ridge regression as our base regression algorithm, the final prediction function from gradient boosting always will be an affine function of the input.

(b) (3 points) Consider the dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$ where $\mathcal{X} = \mathcal{Y} = \mathbb{R}$ and

$$(x_1, y_1) = (0, 0.5), \quad (x_2, y_2) = (3, -1), \quad (x_3, y_3) = (1, 2).$$

Our goal is to use gradient boosting and a software package that fits small regression trees to build our prediction function. We expect our data to have some outliers, so we use the following Huber regression loss:

$$\ell(y, a) = \begin{cases} y - a - \frac{1}{2} & y - a > 1 \\ -(y - a) - \frac{1}{2} & y - a < -1 \\ \frac{1}{2}(y - a)^2 & \text{otherwise.} \end{cases}$$



Suppose we start at $f_0 \equiv 0$, and we want to compute f_1, the prediction function after one round of gradient boosting. Give the dataset that will be passed into the black box regression tree algorithm to compute f_1.

Solution: Since

$$\frac{\partial \ell(y, a)}{\partial a} = \begin{cases} -1 & y - a > 1 \\ 1 & y - a < -1 \\ -(y - a) & \text{otherwise.} \end{cases},$$

For $a = 0$, we have

$$\frac{\partial \ell(y, 0)}{\partial a} = \begin{cases} -1 & y > 1 \\ 1 & y < -1 \\ -y & \text{otherwise.} \end{cases}$$

So the data set is

$$(x_1, y_1) = (0, -0.5), \quad (x_2, y_2) = (3, 1), \quad (x_3, y_3) = (1, -1).$$

Also fine to fit the regression to the negative gradient, and step in the opposite direction, in which case the answer is So the data set is

$$(x_1, y_1) = (0, 0.5), \quad (x_2, y_2) = (3, -1), \quad (x_3, y_3) = (1, 1).$$

4. **Bayesian Bernoulli Model**
Suppose we have a coin with unknown probability of heads $\theta \in (0,1)$. We flip the coin n times and get a sequence $\mathcal{D}$ of coin flips with n_h heads and n_t tails.

Recall the following:

A Beta(α, β) distribution, for shape parameters $\alpha, \beta > 0$, is a distribution supported on the interval $(0,1)$ with PDF given by

$$f(x; \alpha, \beta) \propto x^{\alpha-1}(1-x)^{\beta-1}.$$

The mean of a Beta(α, β) distribution is $\frac{\alpha}{\alpha+\beta}$. The mode is $\frac{\alpha-1}{\alpha+\beta-2}$ assuming $\alpha, \beta \geq 1$ and $\alpha + \beta > 2$. If $\alpha = \beta = 1$, then every value in $(0,1)$ is a mode.

(a) (1 point) Which **ONE** of the following prior distributions on θ corresponds to a strong belief that the coin is approximately fair (i.e. has an equal probability of heads and tails)?

[✓ **Beta**$(50, 50)$] ☐ Beta$(0.1, 0.1)$ ☐ Beta$(1, 100)$

(b) (1 point) Give an expression for the likelihood function $L_{\mathcal{D}}(\theta)$ for this sequence of flips.

Solution:

$$L_{\mathcal{D}}(\theta) = \theta^{n_h}(1-\theta)^{n_t}$$

(c) (1 point) If your posterior distribution on θ is Beta$(3, 6)$, what is your MAP estimate of θ?

Solution: Based on information box above, the mode of the beta distribution is $\frac{\alpha-1}{\alpha+\beta-2}$ for $\alpha, \beta > 1$. So the MAP estimate is $\frac{2}{7}$.

(d) Suppose you have a prior density $p(\theta) = 3\theta^2$ supported on $(0, 1)$. You observe a sequence of flips $\mathcal{D}$ with $n_h = 5$ heads and $n_t = 9$ tails.

i. (2 points) The posterior distribution $p(\theta \mid \mathcal{D})$ is also a beta distribution. Give the posterior distribution in the form $\text{Beta}(\alpha, \beta)$, where you fill in the correct numbers for α and β.

Solution:

$$\begin{aligned} p(\theta|\mathcal{D}) &\propto p(\theta)L_{\mathcal{D}}(\theta) \\ &= 3\theta^2\theta^5(1-\theta)^9 \\ &\propto \theta^7(1-\theta)^9 \end{aligned}$$

which is $\text{Beta}(8, 10)$

ii. (1 point) What is the MLE (maximum likelihood estimate) for θ?

Solution: $\frac{n_h}{n_h+n_t} = \frac{5}{5+9} = \frac{5}{14}$

(e) (2 points) Let $\mathcal{D}$ be a sequence of coin flips. Let $\hat{\theta}_{\text{MLE}}(\mathcal{D})$ be the MLE of θ, and let $\hat{\theta}_{\text{MAP}}(\mathcal{D})$ be the MAP estimate of θ for some prior on θ. Is there a prior $p(\theta)$ in the Beta family of distributions for which $\hat{\theta}_{\text{MLE}}(\mathcal{D}) = \hat{\theta}_{\text{MAP}}(\mathcal{D})$ for all observed sequences $\mathcal{D}$ (assuming at least one coin flip)? If not, explain why not. If so, give the distribution in the form Beta(α, β), where you fill in the α and β.

Solution: Easy by writing down the expressions for the MAP and MLE and equating. If we take $\alpha = \beta = 1$, then $f(x; \alpha, \beta) = 1$ on $(0, 1)$.

5. **EM Algorithm and Latent Variable Models**:

(a) (1 point) __F__ **True or False**: The EM algorithm is equivalent to gradient ascent on the marginal log-likelihood of the observed data.

(b) (1 point) __T__ **True or False**: EM Algorithm may be worth trying when you want to find the MLE in a latent variable model, but the marginal likelihood is difficult to maximize directly.

(c) (1 point) __T__ **True or False**: If θ^i and θ^{i+1} are parameter estimates we get from two consecutive steps of the EM algorithm applied to an MLE problem, then the likelihood of θ^{i+1} is always greater than or equal to the likelihood of θ^i, even when the log-likelihood function is not convex.

(d) (3 points) Suppose we have a latent variable $z \in \{1, 2, 3\}$ and an observed variable $x \in (0, \infty)$ generated as follows:

$$z \sim \text{Categorical}(\pi_1, \pi_2, \pi_3)$$

$$x \mid z \sim \text{Gamma}(2, \beta_z),$$

where $(\beta_1, \beta_2, \beta_3) \in (0, \infty)^3$, and Gamma$(2, \beta)$ is supported on $(0, \infty)$ and has density $p(x) = \beta^2 x e^{-\beta x}$. Suppose we know that $\beta_1 = 1, \beta_2 = 2, \beta_3 = 4$. Give an explicit expression for $p(z = 1|x = 1)$ in terms of the unknown parameters π_1, π_2, π_3.

Solution:

$$\begin{aligned} p(z=1|x=1) &\propto p(z=1|x=1)p(z=1) = \pi_1 e^{-1} \\ p(z=2|x=1) &\propto p(z=2|x=1)p(z=2) = \pi_2 4e^{-2} \\ p(z=3|x=1) &\propto p(z=3|x=1)p(z=3) = \pi_3 16e^{-4} \end{aligned}$$

$$p(z=1|x=1) = \frac{\pi_1 e^{-1}}{\pi_1 e^{-1} + \pi_2 4e^{-2} + \pi_3 16e^{-4}}$$

6. **Multiclass**: We are given the dataset $\mathcal{D} = ((x_1, y_1), \ldots, (x_n, y_n))$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{1, 2, 3\}$. Using a one-vs-all methodology, we have fit the score functions $f_i(x) = w_i^T x$ for $i = 1, 2, 3$, where

$$w_1 = (5, -3)^T, \quad w_2 = (-0.2, 0.6)^T, \quad w_3 = (-0.6, -0.2)^T.$$

(a) (1 point) To fit each w_i, we used a standard linear SVM with regularization parameter $c = 100$. Suppose we have the following multiclass training data:

$$\big((-2, -3), 3\big), \big((2, -1), 1\big), \big((1, 2), 2\big)$$

What dataset was given to the SVM to find w_3? (Assume that the SVM software expects labels to be in $\{-1, 1\}$.)

Solution:

$$\{\big((-2, -3), 1\big), \big((2, -1), -1\big), \big((1, 2), -1\big)\}$$

(b) (1 point) For each of the following new datapoints x, state which class will be predicted.

i. __**1**__ $x = (1, 1)$

ii. __**3**__ $x = (-2, 0)$

(c) (2 points) We want $\psi : \mathbb{R}^2 \times \{1, 2, 3\} \to \mathbb{R}^D$, for some D, and $\tilde{w} \in \mathbb{R}^D$ so that

$$x \mapsto \arg\max_y \tilde{w}^T \psi(x, y)$$

gives the same classification function as the one-vs-all method described above. Give explicit values for $\tilde{w}$, $\psi(x, 1)$, $\psi(x, 2)$, and $\psi(x, 3)$ for which this is the case. If needed, you may refer to the components of x by $x = (x^1, x^2)$.

Solution:

$$\begin{aligned} \tilde{w} &= (5, -3, -0.2, -0.6, -0.6, -0.2) \\ \psi(x, 1) &= (x^1, x^2, 0, 0, 0, 0) \\ \psi(x, 2) &= (0, 0, x^1, x^2, 0, 0) \\ \psi(x, 3) &= (0, 0, 0, 0, x^1, x^2) \end{aligned}$$

7. **Conditional Probability Models: Beta Distribution**

(a) (1 point) We want to model the conditional distribution of $y \in \mathcal{Y} = (0,1)$ given $x \in \mathcal{X} = \mathbf{R}^d$ as a beta distribution (see Problem 4) with parameters $(\alpha, \beta) \in (0,\infty) \times (0,\infty)$. Our approach will be to determine two "scores" $s_\alpha, s_\beta \in \mathbf{R}$ based on the input x, and then predict the beta distribution parameters as follows:

$$(\alpha, \beta) = (\psi(s_\alpha), \psi(s_\beta)).$$

Give a differentiable transfer function $\psi : \mathbf{R} \to \mathbf{R}$ that is appropriate for the model described above. (An incorrect answer in the correct form would be $\psi(s) = 1/s$.)

Solution:

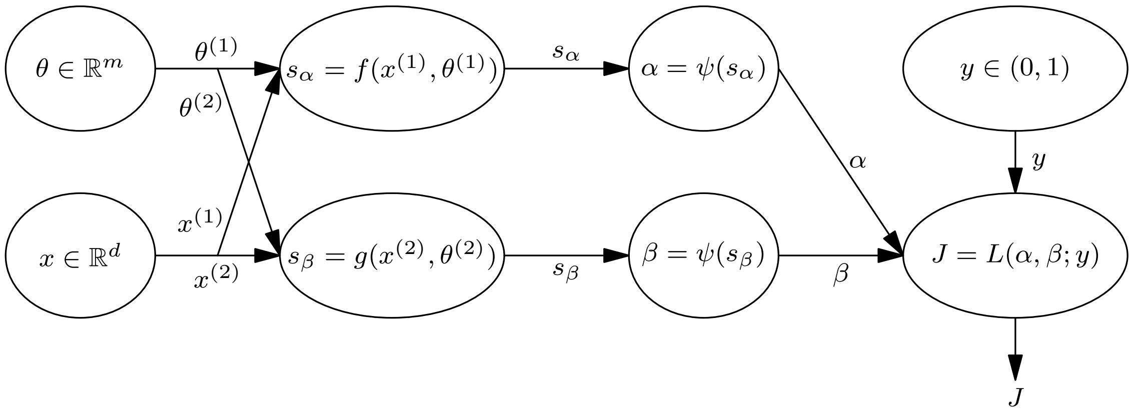
$$\psi(s) = \exp(s)$$

(b) (2 points) We will use two parametrized score functions to produce our scores: $s_\alpha = f(x;\theta)$ and $s_\beta = g(x;\theta)$, for some parameter vector $\theta = (\theta_1, \ldots, \theta_m) \in \mathbf{R}^m$. Give an expression for $L(\theta; x, y)$, the **log-likelihood** of θ for a single observation $(x,y) \in \mathbf{R}^d \times (0,1)$. You may write $\text{Beta}(y;\alpha,\beta)$ for the $\text{Beta}(\alpha,\beta)$ density evaluated at y.

Solution:

$$L(\theta; x, y) = \log\left[\text{Beta}(y; \psi(f(x;\theta)), \psi(g(x;\theta))\right].$$

(c) (2 points) Suppose $f(x;\theta)$ and $g(x;\theta)$ are differentiable with respect to θ. For any example (x, y), the log-likelihood function is $J = L(\theta; x, y)$. Below is a computation graph for computing J. Give an expression for the **scalar** value $\frac{\partial J}{\partial \theta_i}$ in terms of the "local" partial derivatives at each node. That is, you may assume you know the partial derivative of the output of any node with respect to each of its scalar inputs. For example, you may write your expression in terms of $\frac{\partial J}{\partial \alpha}$, $\frac{\partial J}{\partial \beta}$, $\frac{\partial \alpha}{\partial s_\alpha}$, $\frac{\partial s_\alpha}{\partial \theta_i^{(1)}}$, etc. Note that, as discussed in lecture, $\theta^{(1)}$ and $\theta^{(2)}$ represent identical copies of θ, and similarly for $x^{(1)}$ and $x^{(2)}$.



$\theta \in \mathbb{R}^m$
$\theta^{(1)}$
$\theta^{(2)}$
$s_\alpha = f(x^{(1)}, \theta^{(1)})$
s_α
$\alpha = \psi(s_\alpha)$
$y \in (0,1)$
α
y
$x^{(1)}$
$x \in \mathbb{R}^d$
$x^{(2)}$
$s_\beta = g(x^{(2)}, \theta^{(2)})$
s_β
$\beta = \psi(s_\beta)$
β
$J = L(\alpha, \beta; y)$
J

Solution:

$$\frac{\partial J}{\partial \theta_i} = \frac{\partial J}{\partial \alpha}\frac{\partial \alpha}{\partial s_\alpha}\frac{\partial s_\alpha}{\partial \theta_i^{(1)}} + \frac{\partial J}{\partial \beta}\frac{\partial \beta}{\partial s_\beta}\frac{\partial s_\beta}{\partial \theta_i^{(2)}}$$

8. **Neural Networks**:

(a) (1 point) SKIP **True or False**: Backpropagation is an algorithm for computing the gradient of a scalar-valued function represented by a computation graph. [Could also be used for computing the derivative of a vector-valued function, though we use it for gradient descent optimization, which only makes sense for scalar-valued functions.]

(b) (1 point) __F__ **True or False**: Consider a hypothesis space $\mathcal{H}$ of prediction functions $f : \mathbf{R}^d \to \mathbf{R}$ given by a multilayer perceptron (MLP) with 3 hidden layers, each consisting of m nodes, for which the activation function is $\sigma(x) = cx$, for some fixed $c \in \mathbf{R}$. Then this hypothesis space is strictly larger than the set of all affine functions mapping $\mathbf{R}^d$ to $\mathbf{R}$.

(c) (1 point) __T__ **True or False**: Let $g : [0,1]^d \to \mathbf{R}$ be any continuous function on the compact set $[0,1]^d$. Then for any $\varepsilon > 0$, there exists $m \in \{1,2,3,\ldots\}$, $a = (a_1,\ldots,a_m) \in \mathbf{R}^m$,$b = (b_1,\ldots,b_m) \in \mathbf{R}^m$, and $W = \begin{pmatrix} - & w_1^T & - \\ \vdots & \vdots & \vdots \\ - & w_m^T & - \end{pmatrix} \in \mathbf{R}^{m \times d}$ for which the function $f : [0,1]^d \to \mathbf{R}$ given by

$$f(x) = \sum_{i=1}^{m} a_i \max(0, w_i^T x + b_i)$$

satisfies $|f(x) - g(x)| < \varepsilon$ for all $x \in [0,1]^d$.

(d) (1 point) SKIP **True or False**: Suppose $f(x)$ was found by running gradient descent to minimize an objective function over a hypothesis space of multilayer perceptrons. Then f will always be a continuous function, while a prediction function from a gradient boosted regression tree will generally not be continuous. [I should have added "with continuous activation functions". There are discontinuous activation functions for which the MLP is still differentiable with respect to the parameters, such as a step function where the size of the step is the parameter.]

(e) (2 points) Suppose we have fit a kernelized SVM to some training data $(x_1,y_1),\ldots,(x_n,y_n) \in \mathbf{R} \times \{-1,1\}$, and we end up with a score function of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i),$$

where $k(x,x') = \varphi(x - x')$, for some function $\varphi : \mathbf{R} \to \mathbf{R}$. Write $f : \mathbf{R} \to \mathbf{R}$ as a multilayer perceptron by specifying how to set m and a_i, w_i, b_i for $i = 1,\ldots,m$, and the activation function $\sigma : \mathbf{R} \to \mathbf{R}$ in the following expression:

$$f(x) = \sum_{i=1}^{m} a_i \sigma(w_i x + b_i)$$

Solution: Simply take $m = n$. Then $a_i = \alpha_i$, $w_i = 1$, and $b_i = -x_i$. Finally take $\sigma = \varphi$.

4. **Bayesian Bernoulli Model**
Suppose we have a coin with unknown probability of heads $\theta \in (0,1)$. We flip the coin n times and get a sequence $\mathcal{D}$ of coin flips with n_h heads and n_t tails.

Recall the following:

> A Beta(α,β) distribution, for shape parameters $\alpha,\beta > 0$, is a distribution supported on the interval $(0,1)$ with PDF given by
>
> $$f(x;\alpha,\beta) \propto x^{\alpha-1}(1-x)^{\beta-1}.$$
>
> The mean of a Beta(α,β) distribution is $\frac{\alpha}{\alpha+\beta}$. The mode is $\frac{\alpha-1}{\alpha+\beta-2}$ assuming $\alpha,\beta \geq 1$ and $\alpha+\beta > 2$. If $\alpha=\beta=1$, then every value in $(0,1)$ is a mode.

(a) (1 point) Which **ONE** of the following prior distributions on θ corresponds to a strong belief that the coin is approximately fair (i.e. has an equal probability of heads and tails)?

[✓ **Beta**$(50,50)$] ☐ Beta$(0.1,0.1)$ ☐ Beta$(1,100)$

(b) (1 point) Give an expression for the likelihood function $L_{\mathcal{D}}(\theta)$ for this sequence of flips.

Solution:
$$L_{\mathcal{D}}(\theta) = \theta^{n_h}(1-\theta)^{n_t}$$

(c) (1 point) If your posterior distribution on θ is Beta$(3,6)$, what is your MAP estimate of θ?

Solution: Based on information box above, the mode of the beta distribution is $\frac{\alpha-1}{\alpha+\beta-2}$ for $\alpha,\beta > 1$. So the MAP estimate is $\frac{2}{7}$.

(d) Suppose you have a prior density $p(\theta) = 3\theta^2$ supported on $(0, 1)$. You observe a sequence of flips $\mathcal{D}$ with $n_h = 5$ heads and $n_t = 9$ tails.

i. (2 points) The posterior distribution $p(\theta \mid \mathcal{D})$ is also a beta distribution. Give the posterior distribution in the form $\text{Beta}(\alpha, \beta)$, where you fill in the correct numbers for α and β.

Solution:

$$\begin{aligned} p(\theta|\mathcal{D}) &\propto p(\theta)L_{\mathcal{D}}(\theta) \\ &= 3\theta^2\theta^5(1-\theta)^9 \\ &\propto \theta^7(1-\theta)^9 \end{aligned}$$

which is $\text{Beta}(8, 10)$

ii. (1 point) What is the MLE (maximum likelihood estimate) for θ?

Solution: $\frac{n_h}{n_h+n_t} = \frac{5}{5+9} = \frac{5}{14}$

(e) (2 points) Let $\mathcal{D}$ be a sequence of coin flips. Let $\hat{\theta}_{\text{MLE}}(\mathcal{D})$ be the MLE of θ, and let $\hat{\theta}_{\text{MAP}}(\mathcal{D})$ be the MAP estimate of θ for some prior on θ. Is there a prior $p(\theta)$ in the Beta family of distributions for which $\hat{\theta}_{\text{MLE}}(\mathcal{D}) = \hat{\theta}_{\text{MAP}}(\mathcal{D})$ for all observed sequences $\mathcal{D}$ (assuming at least one coin flip)? If not, explain why not. If so, give the distribution in the form Beta(α, β), where you fill in the α and β.

Solution: Easy by writing down the expressions for the MAP and MLE and equating. If we take $\alpha = \beta = 1$, then $f(x; \alpha, \beta) = 1$ on $(0, 1)$.

5. **EM Algorithm and Latent Variable Models**:

(a) (1 point) <u>F</u> **True or False**: The EM algorithm is equivalent to gradient ascent on the marginal log-likelihood of the observed data.

(b) (1 point) <u>T</u> **True or False**: EM Algorithm may be worth trying when you want to find the MLE in a latent variable model, but the marginal likelihood is difficult to maximize directly.

(c) (1 point) <u>T</u> **True or False**: If θ^i and θ^{i+1} are parameter estimates we get from two consecutive steps of the EM algorithm applied to an MLE problem, then the likelihood of θ^{i+1} is always greater than or equal to the likelihood of θ^i, even when the log-likelihood function is not convex.

(d) (3 points) Suppose we have a latent variable $z \in \{1, 2, 3\}$ and an observed variable $x \in (0, \infty)$ generated as follows:

$$z \sim \text{Categorical}(\pi_1, \pi_2, \pi_3)$$
$$x \mid z \sim \text{Gamma}(2, \beta_z),$$

where $(\beta_1, \beta_2, \beta_3) \in (0, \infty)^3$, and Gamma$(2, \beta)$ is supported on $(0, \infty)$ and has density $p(x) = \beta^2 x e^{-\beta x}$. Suppose we know that $\beta_1 = 1, \beta_2 = 2, \beta_3 = 4$. Give an explicit expression for $p(z = 1|x = 1)$ in terms of the unknown parameters π_1, π_2, π_3.

Solution:

$$p(z = 1|x = 1) \propto p(z = 1|x = 1)p(z = 1) = \pi_1 e^{-1}$$
$$p(z = 2|x = 1) \propto p(z = 2|x = 1)p(z = 2) = \pi_2 4e^{-2}$$
$$p(z = 3|x = 1) \propto p(z = 3|x = 1)p(z = 3) = \pi_3 16e^{-4}$$

$$p(z = 1|x = 1) = \frac{\pi_1 e^{-1}}{\pi_1 e^{-1} + \pi_2 4e^{-2} + \pi_3 16e^{-4}}$$

6. **Multiclass**: We are given the dataset $\mathcal{D} = ((x_1, y_1), \ldots, (x_n, y_n))$ where $x_i \in \mathbb{R}^2$ and $y_i \in \{1, 2, 3\}$. Using a one-vs-all methodology, we have fit the score functions $f_i(x) = w_i^T x$ for $i = 1, 2, 3$, where

$$w_1 = (5, -3)^T, \quad w_2 = (-0.2, 0.6)^T, \quad w_3 = (-0.6, -0.2)^T.$$

(a) (1 point) To fit each w_i, we used a standard linear SVM with regularization parameter $c = 100$. Suppose we have the following multiclass training data:

$$\big((-2, -3), 3\big), \big((2, -1), 1\big), \big((1, 2), 2\big)$$

What dataset was given to the SVM to find w_3? (Assume that the SVM software expects labels to be in $\{-1, 1\}$.)

Solution:

$$\{\big((-2, -3), 1\big), \big((2, -1), -1\big), \big((1, 2), -1\big)\}$$

(b) (1 point) For each of the following new datapoints x, state which class will be predicted.

i. __**1**__ $x = (1, 1)$

ii. __**3**__ $x = (-2, 0)$

(c) (2 points) We want $\psi : \mathbb{R}^2 \times \{1, 2, 3\} \to \mathbb{R}^D$, for some D, and $\tilde{w} \in \mathbb{R}^D$ so that

$$x \mapsto \arg\max_y \tilde{w}^T \psi(x, y)$$

gives the same classification function as the one-vs-all method described above. Give explicit values for $\tilde{w}$, $\psi(x, 1)$, $\psi(x, 2)$, and $\psi(x, 3)$ for which this is the case. If needed, you may refer to the components of x by $x = (x^1, x^2)$.

Solution:

$$\begin{aligned}
\tilde{w} &= (5, -3, -0.2, -0.6, -0.6, -0.2) \\
\psi(x, 1) &= (x^1, x^2, 0, 0, 0, 0) \\
\psi(x, 2) &= (0, 0, x^1, x^2, 0, 0) \\
\psi(x, 3) &= (0, 0, 0, 0, x^1, x^2)
\end{aligned}$$

7. **Conditional Probability Models: Beta Distribution**

(a) (1 point) We want to model the conditional distribution of $y \in \mathcal{Y} = (0,1)$ given $x \in \mathcal{X} = \mathbf{R}^d$ as a beta distribution (see Problem 4) with parameters $(\alpha, \beta) \in (0,\infty) \times (0,\infty)$. Our approach will be to determine two "scores" $s_\alpha, s_\beta \in \mathbf{R}$ based on the input x, and then predict the beta distribution parameters as follows:

$$(\alpha, \beta) = (\psi(s_\alpha), \psi(s_\beta)).$$

Give a differentiable transfer function $\psi : \mathbf{R} \to \mathbf{R}$ that is appropriate for the model described above. (An incorrect answer in the correct form would be $\psi(s) = 1/s$.)

Solution:

$$\psi(s) = \exp(s)$$

(b) (2 points) We will use two parametrized score functions to produce our scores: $s_\alpha = f(x;\theta)$ and $s_\beta = g(x;\theta)$, for some parameter vector $\theta = (\theta_1, \ldots, \theta_m) \in \mathbf{R}^m$. Give an expression for $L(\theta; x, y)$, the **log-likelihood** of θ for a single observation $(x,y) \in \mathbf{R}^d \times (0,1)$. You may write $\text{Beta}(y;\alpha,\beta)$ for the $\text{Beta}(\alpha,\beta)$ density evaluated at y.

Solution:

$$L(\theta; x, y) = \log\left[\text{Beta}(y; \psi(f(x;\theta)), \psi(g(x;\theta))\right].$$

(c) (2 points) Suppose $f(x;\theta)$ and $g(x;\theta)$ are differentiable with respect to θ. For any example (x, y), the log-likelihood function is $J = L(\theta; x, y)$. Below is a computation graph for computing J. Give an expression for the **scalar** value $\frac{\partial J}{\partial \theta_i}$ in terms of the "local" partial derivatives at each node. That is, you may assume you know the partial derivative of the output of any node with respect to each of its scalar inputs. For example, you may write your expression in terms of $\frac{\partial J}{\partial \alpha}$, $\frac{\partial J}{\partial \beta}$, $\frac{\partial \alpha}{\partial s_\alpha}$, $\frac{\partial s_\alpha}{\partial \theta_i^{(1)}}$, etc. Note that, as discussed in lecture, $\theta^{(1)}$ and $\theta^{(2)}$ represent identical copies of θ, and similarly for $x^{(1)}$ and $x^{(2)}$.

$$\begin{tikzcd}
\theta \in \mathbb{R}^m \arrow[r, "\theta^{(1)}"] \arrow[dr, "\theta^{(2)}"'] & s_\alpha = f(x^{(1)}, \theta^{(1)}) \arrow[r, "s_\alpha"] & \alpha = \psi(s_\alpha) \arrow[dr, "\alpha"] & y \in (0,1) \arrow[d, "y"] \\
x \in \mathbb{R}^d \arrow[r, "x^{(2)}"'] \arrow[ur, "x^{(1)}"] & s_\beta = g(x^{(2)}, \theta^{(2)}) \arrow[r, "s_\beta"'] & \beta = \psi(s_\beta) \arrow[r, "\beta"'] & J = L(\alpha, \beta; y) \arrow[d] \\
 & & & J
\end{tikzcd}$$

Solution:

$$\frac{\partial J}{\partial \theta_i} = \frac{\partial J}{\partial \alpha}\frac{\partial \alpha}{\partial s_\alpha}\frac{\partial s_\alpha}{\partial \theta_i^{(1)}} + \frac{\partial J}{\partial \beta}\frac{\partial \beta}{\partial s_\beta}\frac{\partial s_\beta}{\partial \theta_i^{(2)}}$$

8. **Neural Networks**:

(a) (1 point) SKIP **True or False**: Backpropagation is an algorithm for computing the gradient of a scalar-valued function represented by a computation graph. [Could also be used for computing the derivative of a vector-valued function, though we use it for gradient descent optimization, which only makes sense for scalar-valued functions.]

(b) (1 point) __F__ **True or False**: Consider a hypothesis space $\mathcal{H}$ of prediction functions $f : \mathbf{R}^d \to \mathbf{R}$ given by a multilayer perceptron (MLP) with 3 hidden layers, each consisting of m nodes, for which the activation function is $\sigma(x) = cx$, for some fixed $c \in \mathbf{R}$. Then this hypothesis space is strictly larger than the set of all affine functions mapping $\mathbf{R}^d$ to $\mathbf{R}$.

(c) (1 point) __T__ **True or False**: Let $g : [0,1]^d \to \mathbf{R}$ be any continuous function on the compact set $[0,1]^d$. Then for any $\varepsilon > 0$, there exists $m \in \{1,2,3,\ldots\}$, $a = (a_1,\ldots,a_m) \in \mathbf{R}^m$,$b = (b_1,\ldots,b_m) \in \mathbf{R}^m$, and $W = \begin{pmatrix} - & w_1^T & - \\ \vdots & \vdots & \vdots \\ - & w_m^T & - \end{pmatrix} \in \mathbf{R}^{m\times d}$ for which the function $f : [0,1]^d \to \mathbf{R}$ given by

$$f(x) = \sum_{i=1}^{m} a_i \max(0, w_i^T x + b_i)$$

satisfies $|f(x) - g(x)| < \varepsilon$ for all $x \in [0,1]^d$.

(d) (1 point) SKIP **True or False**: Suppose $f(x)$ was found by running gradient descent to minimize an objective function over a hypothesis space of multilayer perceptrons. Then f will always be a continuous function, while a prediction function from a gradient boosted regression tree will generally not be continuous. [I should have added "with continuous activation functions". There are discontinuous activation functions for which the MLP is still differentiable with respect to the parameters, such as a step function where the size of the step is the parameter.]

(e) (2 points) Suppose we have fit a kernelized SVM to some training data $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbf{R} \times \{-1, 1\}$, and we end up with a score function of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i),$$

where $k(x, x') = \varphi(x - x')$, for some function $\varphi : \mathbf{R} \to \mathbf{R}$. Write $f : \mathbf{R} \to \mathbf{R}$ as a multilayer perceptron by specifying how to set m and a_i, w_i, b_i for $i = 1, \ldots, m$, and the activation function $\sigma : \mathbf{R} \to \mathbf{R}$ in the following expression:

$$f(x) = \sum_{i=1}^{m} a_i \sigma(w_i x + b_i)$$

Solution: Simply take $m = n$. Then $a_i = \alpha_i$, $w_i = 1$, and $b_i = -x_i$. Finally take $\sigma = \varphi$.