Sta-395 Intro to ML (Practice Exam)

Your Name: Sketch Solution

Directions

- Answer each question clearly and concisely. Answers that are unnecessarily verbose may be penalized if the correct answer is not clearly communicated.
- Do not include superfluous information in your answers, you may be penalized if you make an inaccurate statement even if you go on to provide a correct answer. You should be careful to answer only the question that is asked.

Formulas

Performance Metrics:

- **True Positive Rate (TPR)** - $\frac{\text{True Positives}}{\text{Total Positives}}$
- **Sensitivity** and **Recall** - TPR
- **Specificity** - $1 - \text{FPR}$
- **Precision** - $\frac{\text{True Positives}}{\text{Total Predicted Positives}}$
- **Balanced Accuracy** - $\frac{\text{TPR}+\text{FPR}}{2}$
- **F1 Score** - $\frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$

Vector Calculus:

- Given $f(\mathbf{x}) = c\mathbf{x}^T\mathbf{x}$, the vector derivative is $\frac{\partial f}{\partial \mathbf{x}} = 2c\mathbf{x}$
- Given $f(\mathbf{x}) = c\mathbf{x}^T\mathbf{A}\mathbf{x}$, the vector derivative is $\frac{\partial f}{\partial \mathbf{x}} = 2c\mathbf{A}\mathbf{x}$
- Given $f(\mathbf{x}) = c\mathbf{x}^T\mathbf{a}$, the vector derivative is $\frac{\partial f}{\partial \mathbf{x}} = 2c\mathbf{a}$
- Given $f(\mathbf{x}) = c\mathbf{x}^T\mathbf{A}$, the vector derivative is $\frac{\partial f}{\partial \mathbf{x}} = 2c\mathbf{A}$

Section 1

Part A: Briefly describe the difference between *supervised* and *unsupervised learning*, then provide a short description of a realistic scenario that would be best suited for unsupervised learning.

- Supervised learning involves a pre-determined outcome variable and finding patterns among predictors that can accurately predict it. Unsupervised learning involves finding patterns in a data set without the use of pre-determined outcome variable.
- Unsupervised learning, clustering in particular, could be used to form meaningful groups of customers from an unstructured sales database.

Part B: Consider the application of 5-fold cross-validation to evaluate the performance of the decision tree classification algorithm with maximum depth of 4 using a training set consisting of $n = 100$ observations.

- i: How many times must a decision tree be trained during the evaluation process?
  - 5 times, a tree is trained once per fold
- ii: How many data-points are used to train each decision tree involved in the evaluation process?
  - 80 data-points, as all of the data other than the held out fold (containing 20 data-points) is used each time a model is trained.

Part C: Consider a machine learning application that seeks to classify emails as either "spam" or "not spam". On the test set, 980 of 1000 "not spam" emails were correctly classified but only 120 of 300 "spam" emails were correctly classified.

- i: Create a confusion matrix summarizing the performance of this classifier. Consider the outcome "spam" to be the positive class.

	pos	neg
predicted pos	120	20
predicted neg	180	980

- ii: What is the *classification accuracy* of the classifier?
  - $\frac{980+120}{1300} = 84.6\%$ accuracy
- iii: What is the *F1 score* of the classifier?
  - $textprecision = \frac{120}{120+20} = 0.8571$, $textrecall = \frac{120}{120+180} = 0.4$
  - $F1 = \frac{2*0.8571*0.4}{0.8571+0.4} = 0.5454$
- iv: Which metric, classification accuracy or F1 score, provides a more useful assessment of how effective this classifier is? Briefly explain.
  - This application contains imbalanced classes and is likely to be more focused on the positive class, "spam", thus the F1 score is more useful because it is not influenced by the class imbalance and focuses on the positive class.

Part D: Clearly mark each of the following statements as either TRUE or FALSE. If a statement is false, briefly explain why it is incorrect or how it must be modified to be correct.

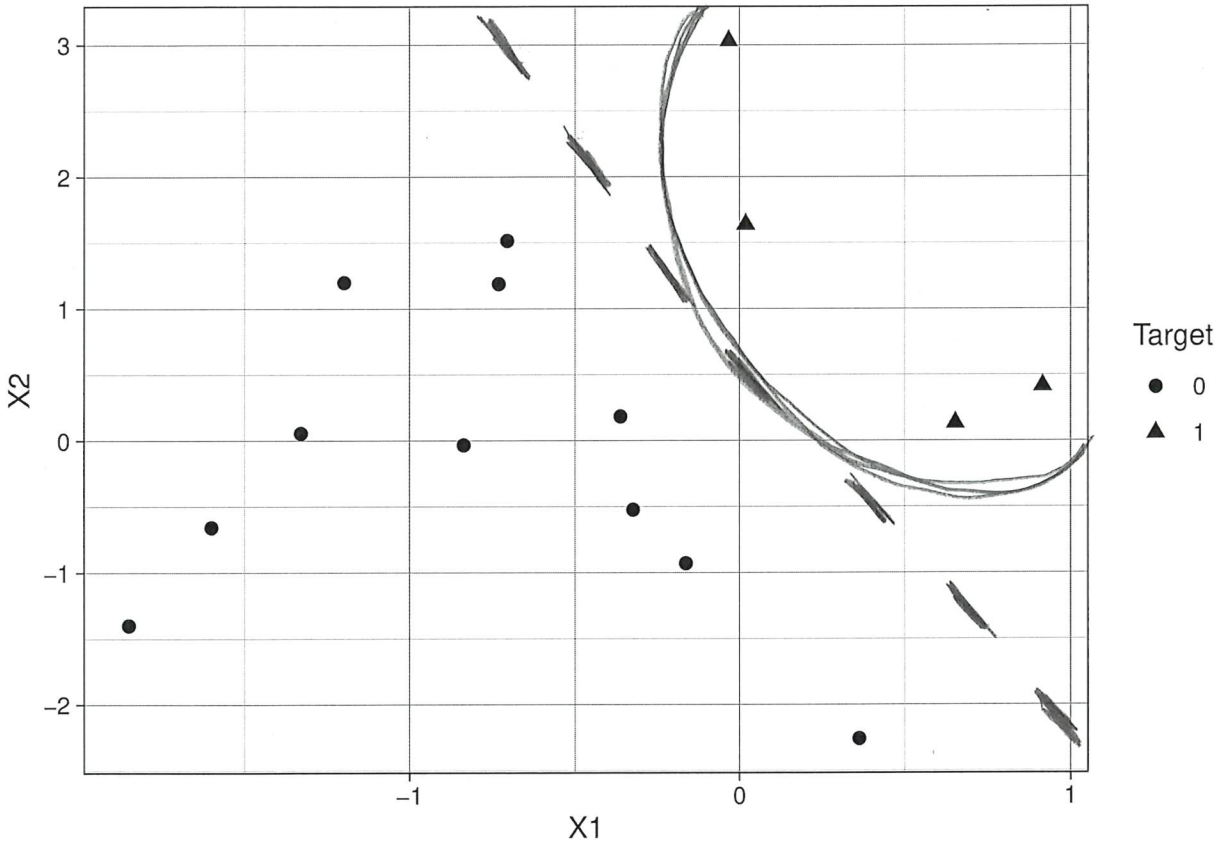
- i: Min-max scaling can be used to modify the distributional shape of a variable before it is used in machine learning model
  - False. *Normalizing transformations* such as Box-Cox or Yeo-Johnson modify the distributional shape.
- ii: Principal component analysis is most useful as a pre-processing step when data contain a large number of predictors that are not related to one another
  - False. PCA is most useful when you have a large number of *highly correlated* predictors (ie: lots of covariance among the predictors)

- iii: In k-nearest neighbors, uniform weighting allows neighbors that are closer to a new data-point to have a larger influence on the algorithm's prediction than neighbors that are further away from the new data-point
  - False. *Distance weighting* allows this.
- iv: In k-nearest neighbors, using more neighbors will decrease the bias of the algorithm's predictions.
- v: While the random forest algorithm benefits from bagging, or bootstrap aggregation, this is not an essential component of the algorithm.
  - False. Without bootstrapping the random forest would not produce useful results because all of the trees/base models in the ensemble would be identical (or similar if predictor sampling is used).
- vi: An essential step in setting up a gradient descent algorithm is differentiating the cost function with respect to the machine learning model's unknown parameters
  - True.

Section 2

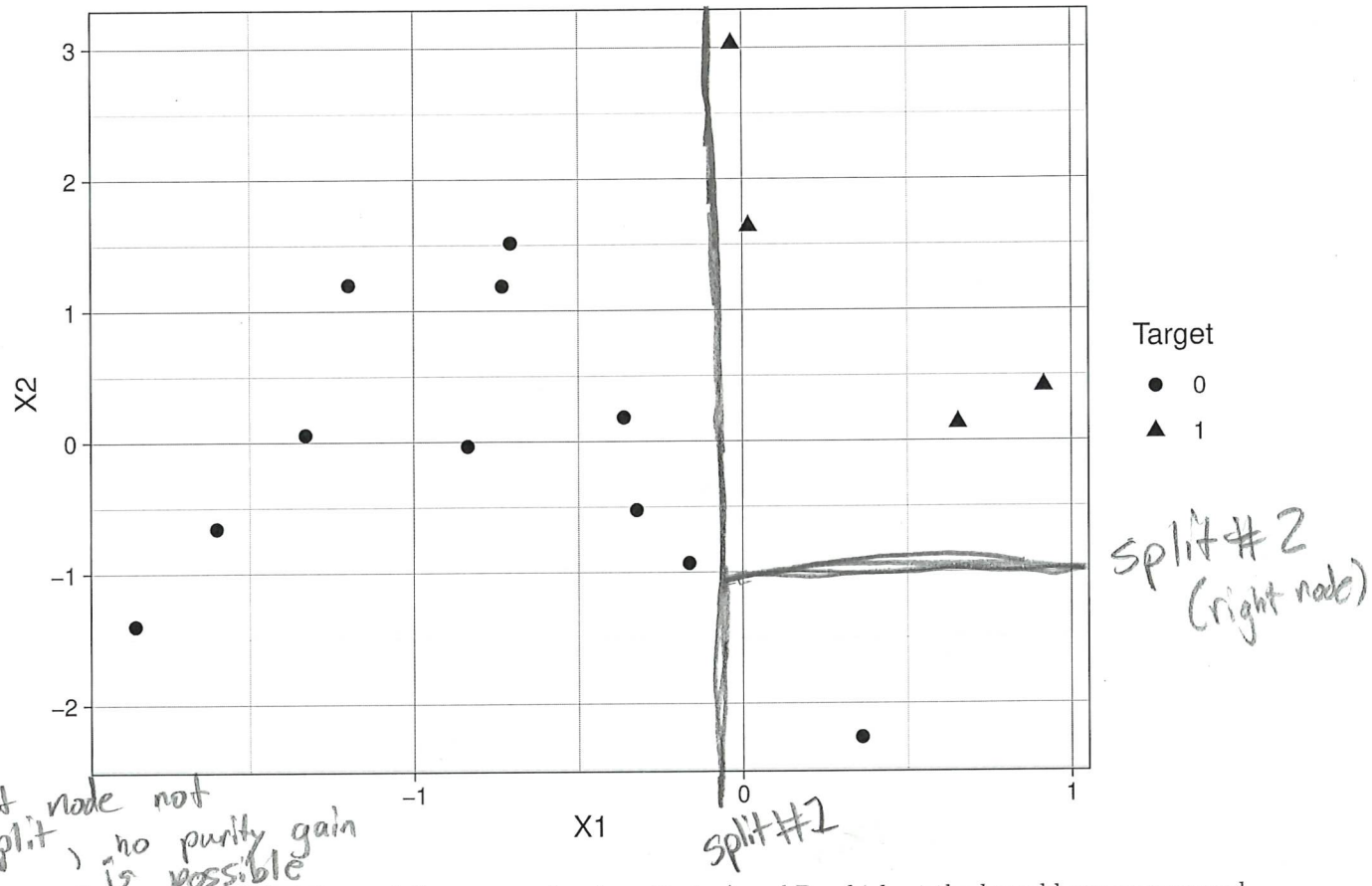
Part A: Consider the training data shown below and the goal of correctly classifying "Target" as either 1 or 0. On the given graph

- i: Sketch the decision boundary of a support vector machine classifier using a radial basis function (RBF) kernel *using a solid line*. Your boundary doesn't need to be perfect, but it should clearly illustrate that you understand this method.
  - See graph
- ii: Sketch the decision boundary of a support vector machine classifier using a linear kernel *using a dashed line*. Your boundary doesn't need to be perfect, but it should clearly illustrate that you understand this method.
  - See graph



Part B: Shown below are the same data used in Part A

- i: Sketch the splitting rules of a decision tree classifier with a maximum depth of 2. Your split locations don't need to be perfect, but they should clearly illustrate that you understand this method.
  - See graph
- ii: Consider *decreasing* the maximum depth of the classifier from Part A to a value of 1. What consequences does this change have on the *bias* and *variance* as the classification method?
  - This would increase the bias but decrease the variance of the classifier.



Part C: Considering all of the modeling approaches from Parts A and B, which method would you recommend for a client who is interested in achieving maximum classification accuracy *on new data* that arises from the same collection procedure that gave rise to the data shown in Parts A and B? Briefly explain your answer. - The RBF kernel SVM seems to produce the decision boundary that is most robust and generalizable, but the linear kernel SVM also seems acceptable. The decision tree model in Part B is overly specific to these data and contains boundaries that aren't as likely to generalize well to new data.

Section 3

Consider the application of simple neural network consisting of one hidden layer with only one neuron and the sigmoid activation function to a data set involving 2 predictors. Furthermore, you may assume the outcome variable is numeric and the network is trained using the squared error cost function: Cost $= \frac{1}{n}(\mathbf{y}-\hat{\mathbf{y}})^T(\mathbf{y}-\hat{\mathbf{y}})$

Part A: How many weight parameters are present in this network?

3 — 2 in first layer, 1 in second layer

Part B: Assume that biases are used in both the hidden layer and the final layer. How many bias parameters are present in this network?

2 — 1 in each layer

Part C: Under the simplifying assumption that we'll be training this network using batches of size $n = 1$ (ie: a single 2-dimensional vector $\mathbf{x}_i$ containing the values of each predictor for the i^{th} sample), calculate the gradient components corresponding to each weight and bias parameter involved in this network using chain rule. Clearly label each component. You may denote the sigmoid function as $\sigma()$ and it's derivative as $\sigma() * (1 - \sigma())$ when forming your answer.

$$\frac{\partial Cost}{\partial b^2} = \frac{\partial Cost}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b^2}$$

$$\frac{\partial Cost}{\partial b^2} = (2y + 2\hat{y}) \cdot 1$$

$$\frac{\partial Cost}{\partial w^2} = \frac{\partial Cost}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w^2}$$

$$\frac{\partial Cost}{\partial w^2} = (2y + 2\hat{y}) \cdot a_1 \quad \text{where } a_1 = \sigma(b' + x w')$$

$$\frac{\partial Cost}{\partial b'} = \frac{\partial Cost}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a'} \cdot \frac{\partial a'}{\partial z'} \cdot \frac{\partial z'}{\partial b'}$$

$$\frac{\partial Cost}{\partial b'} = (2y + 2\hat{y}) \cdot w^2 \cdot \sigma(z')(1 - \sigma(z')) \cdot 1$$

$$\frac{\partial Cost}{\partial w'} = \frac{\partial Cost}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a'} \cdot \frac{\partial a'}{\partial z'} \cdot \frac{\partial z'}{\partial w'}$$

$$\frac{\partial Cost}{\partial w'} = (2y + 2\hat{y}) w^2 \sigma(z')(1 - \sigma(z')) \cdot x$$

note! $x = [x_1, x_2]$

So this expression is a 1x2 vector for w_{11} and w_{12}

Part D: Briefly explain how the gradient calculated in Part C is used to update the weight parameters of the network during training.

- The gradient from Part C describes slope of the tangent line to the cost function with respect to each of the model's parameters. We can use this slope to update these parameters in a direction that improves the cost. More specifically, we can make the updates of the form: $parameter^{i+1} = parameter^{i} - \alpha * gradient(parameter^{i})$ where α is the learning rate (step size)