Written Assignment 3

Due: Friday 11/14/2025 @ 11:59pm EST

Disclaimer

I encourage you to work together, I am a firm believer that we are at our best (and learn better) when we communicate with our peers. Perspective is incredibly important when it comes to solving problems, and sometimes it takes talking to other humans (or rubber ducks in the case of programmers) to gain a perspective we normally would not be able to achieve on our own. The only thing I ask is that you report who you work with: this is **not** to punish anyone, but instead will help me figure out what topics I need to spend extra time on/who to help. When you turn in your solutions (please use some form of typesetting: do **NOT** turn in handwritten solutions), please note who you worked with at the very beginning of the pdf.

Question 1: Which Points Belong in the Test Set? (25 points)

Suppose you have a learning algorithm that you want to test the performance of. The labels of your data are boolean, and you have n positive examples as well as n negative examples in your dataset. In order to test the performance of your model, you are going to compare the accuracy of your model against the accuracy of a *majority classifier* (i.e. a model that predicts the majority class contained within the training data). You will run a *leave-one-out* cross-validaton experiment (i.e. if you have x samples in your dataset, you will train your model on $x - 1$ samples and test on the remaining "left out" sample. You will repeat this process until each sample gets to be the "left out" test point, training the model from scratch each time). You expect the majority classifier to score about 50% every time, but it scores 0%. Why?

Note: this is a **proof** question, meaning you must follow formal proof structure (see the examples of piazza for guidance).

Question 2: Combining Multiple Models into an Ensemble (25 points)

Let us say that you have K separate classifiers, each trained on the same data. You combine them together into a single model by letting them vote: given a test point, predict that test point using all K classifiers, and then choose the most-frequently predicted class as your prediction. Such a model is called a *majority voting ensemble*. Suppose that each classifier has error ϵ, and that the errors made by each classifier are independent of the others'. Derive a formula for the error of the ensemble as a function of K and ϵ.

Note: this is a **proof** question, meaning you must follow formal proof structure (see the examples of piazza for guidance).

Question 3: Linear Activations (25 points)

Suppose you had a neural network where every unit is equipped with a linear activation function, i.e. the output of a unit is some constant c times the weighted sum of its inputs:

1. Assume that the network has one hidden layer. For a given assignment of parameters, derive equations for the output of the units in the output layer as a function of the units in the input layer without any explicit mention of the output of the hidden layer. Show that there is a network with no hidden units that computes the same function.

2. Repeat part 1. but this time for a network with an arbitrary number of hidden layers. Conclude that for a neural network to learn any kind of nonlinear relationships, there must be at least a single unit with a nonlinear activation function.

Note: this is a **proof** question, meaning you must follow formal proof structure (see the examples of piazza for guidance).

Question 4: Datasets with Weights (25 points)

Consider a dataset in which each data point $\left(x^{(i)}, y_{gt}^{(i)}\right)$ is associated with some weight $r^{(i)} > 0$. If we want to use a mean squared error for our loss function (like we want to do in temporal difference learning), our objective now becomes:

$$L(\vec{\theta}) = \frac{1}{2N} \sum_{i=1}^{N} r^{(i)} \left(y_{gt}^{(i)} - f_{\vec{\theta}}(x^{(i)})\right)^2$$

For now, lets simplify $f_{\vec{\theta}}$ to be a linear model (which in an earlier homework you showed that any completely-linear neural network could be reduced to this) $f_{\vec{\theta}}(x) = \vec{\theta}^T \phi(x)$. Plugging this in:

$$L(\vec{\theta}) = \frac{1}{2N} \sum_{i=1}^{N} r^{(i)} \left(y_{gt}^{(i)} - \vec{\theta}^T \phi(x^{(i)})\right)^2$$

Derive an expression for the optimum $\vec{\theta}^*$ that minimizes this loss function.

Note: this is a **proof** question, meaning you must follow formal proof structure (see the examples of piazza for guidance).

Extra Credit: Decision Trees with Missing Values (25 points)

Standard decision trees are not able to handle examples where one (or more) of the attributes contain an unknown value. Any unknown or unfilled entry in an example is called a "missing value", and our decision tree will fail if presented with such an example:

1. First, we need a way to classify any example that contains missing value(s). Suppose an example $\vec{x}$ which has a missing value for attribute A, and that the decision tree test for A at a node that $\vec{x}$ reaches. One way to handle this missing value is to pretend that $\vec{x}$ has *all possible* values of A, and to weight each value according to the frequency that the values appear in the dataset the node was constructed from. This classification algorithm should follow all branches at any node for which a value is missing and should multiply weights along each path. Design a classification algorithm for decision trees that has this behavior.

2. Now, modify the information gain calculation so that when constructing a node from dataset D, the examples with missing values for any of the remaining attributes are given "as-if" values according to the frequencies of those values in D.

Note: this is a **proof** question, meaning you must follow formal proof structure (see the examples of piazza for guidance).