HW #4

Uziel Rivera-Lopez

04/01/2024

Problem 1

We will need to take the derivate of the equation given, $\frac{dE}{dw}$, and we will use the equation $w_{new} = w_{old} - \eta \cdot \frac{dE}{dw}$ for the gradient descent.
$\frac{dE}{dw} = 2w - 4$

1. The first iteration $w_0 = 0, \eta = 0.1$
   - (I) $w_{new} = w_0 - \eta \cdot (\frac{dE}{dw})$
   - (II) $w_{new} = 0 - 0.1 \cdot (2 \cdot 0 - 4)$
   - (III) $w_{new} = w_1 = 0.4$
2. Second iteration $w_1 = 0.4, \eta = 0.1$
   - (I) $w_{new} = w_1 - \eta \cdot (\frac{dE}{dw})$
   - (II) $w_{new} = 0.4 - 0.1 \cdot (2 \cdot 0.4 - 4)$
   - (III) $w_{new} = w_2 = 0.72$
3. Last iteration $w_2 = 0.72, \eta = 0.1$
   - (I) $w_{new} = w_2 - \eta \cdot (\frac{dE}{dw})$
   - (II) $w_{new} = 0.72 - 0.1 \cdot (2 \cdot 0.72 - 4)$
   - (III) $w_{new} = w_3 = 0.976$

Problem 2

Wee need to calculate the value of g(x) for each point indiviually. Because we have two class case we will need to use $g(x) = w^T \cdot x + w_0$ although you guys state g(x,y) we can simply make it into a vector, unless thats not what you guys are referring to ignore this part, v=(x,y)

$$v_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} v_2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

(a) $g(v_1) = [2 - 3] \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 5 = [2(1) + (-3)(2)] + 5 = -4 + 5 = 1$

$g(v_2) = [2 - 3] \cdot \begin{bmatrix} 3 \\ 4 \end{bmatrix} + 5 = [2(3) + (-3)(4)] + 5 = -6 + 5 - 1$

(b) So v_1 is goes in C1 while v_2 goes in to C2

Problem 3

(a) Gradient Descent is an algorithm that helps us find the lowest possible points/values of a function. This doesn't necessarily mean we are finding the absolute minimium as we can get stuck in a local min. It can be used to optimize our machine learning models as it can minimize the amount of errors of between our training dataset and test dataset

(b) The learning is basically how fast we can apporach the minimium of the function. So like in P2 we had a learning rate of 0.1 so we only move/learn 0.1 each time we iterate.

Problem 4

(a) It all depends on the coefficients assigned to each input feature. So in the linear discriminant model (LDA) we pass them through the model, and determine how much of an impact each feature has on the discriminant. Usually higher coefficients tend to have a higher impact, and are the more important features.

(b) The way I see ranking is like recommendation system, let's say I liked the movie Dune, then the system/model will recommend me movies that are similar, like Star Wars. Ranking wants to sort the items/features in a way that is the most important to the least important, most relevant to least relevant. Now in the case of classification, we want to group things that are similar, so like Dune is similar to Star Wars, so we can group them together, which is not the same as ranking since we are not sorting them based on importance. The regression task cares about the relationship between the input and output, so like again wtih Dune, we want to know how much I will like the movie based on the features of the movie.

Problem 5

(a) Pruning is like cutting off branches in a tree. So we do this all the time with binary trees, or path finding algorithms. We want to cut off the branches that we know we don't need to go down, so we can save time and resources. With decision trees, we want to cut off branches, in this case nodes, that will have a small amount of instances, or that don't really help

us in our classification; otherwise, this can lead to trees being too complex leading to overfitting. The two methods used are prepruning, where we stop growing the tree when we reach a certain depth, or a threshold that we set, and postpruning, where we grow the tree to its full extent, and then we start cutting off branches that we don't need. With prepruning we are saving the time of growing the tree fully, but because we do not see the full tree we might be cutting smaller branches that could have been useful. With postpruning we are growing the tree to its full extent, so we can see all the branches, but we might be wasting time growing the tree to its full extent, and then cutting off branches that we don't need. Which leads to prepruning being faster, but less accurate, and postpruning being slower, but more accurate. Usually this is done to big datasets, but we shouldn't do it to smaller datasets as they are already small enough, and going any further will lead to inaccuracy.

(b) Yes it is possible for regression on a trees, which are regression trees. So let's use the MSE error function, and we want to minimize the error. So we set a threshold, and we want to split the data into two groups, one where we can store the data that is less than the threshold, and the other where we store the data that is greater than the threshold.

(c) The rule induction is a way to generate rules from the data, so we can use these rules to classify the data. These rules need to describe the "path" the data takes and what the outcome should be. To extract the rules we need to determine it based on conditions, if-else statements, and the outcome. Rule inductions converts each of these paths of the data into a rule. This method is preferred alot since it's easy for us to understand, gives us a good insight as to what is happening through each step, and if there's some sort of error we can easily go back and see where we went wrong.

Problem 6

(a) Entropy deals with measure the level of disorder in a dataset. So in the context of trees, entropy measure how much impurity/disorder within a dataset is there. So then the goal will be to reduce the entropy at each step by selecting features that help with information gain.

(b) Since the target variable is "Play Tennis", we need to see the amount of total instances, 14, and compare it to the amount of yes, 9, and no, 5.
$Entropy = -p(yes) \cdot log_2(p(yes)) - p(no) \cdot log_2(p(no))$
$Entropy = -\frac{9}{14} \cdot log_2(\frac{9}{14}) - \frac{5}{14} \cdot log_2(\frac{5}{14})$
$Entropy = 0.94029$