Washington State University
School of Electrical Engineering and Computer Science
CptS 315 – Introduction to Data Mining
Online

Ananth Jillepalli

Homework 4

Name: Nam Jun Lee

Student Number: 11606459

Q1. Suppose you are given the following multi-class classification training data, where each input example has three features and output label takes a value from good, bad, and ugly.

• x_1=(0, 1, 0) and y_1=good

• x_2=(1, 0, 1) and y_2=bad

• x_3=(1, 1, 1) and y_3=ugly

• x_4=(1, 0, 0) and y_4=bad

• x_5=(0, 0, 1) and y_5=good

Suppose we want to learn a linear classifier using multi-class perceptron algorithm and start from the following weights: w_{good}=(0,0,0); w_{bad}=(0,0,0); and w_{ugly}=(0,0,0). Please do hand calculations to show how weights change after processing examples in the same order (i.e., one single pass over the five training examples).

x1: y1=good

Score:

Score(good) = 0

Score(bad) = 0

Score(ugly) = 0

w(good) = w(good) + x1 = (0,0,0) + (0,1,0) = (0,1,0)

w(bad) = w(bad) – x1 = (0,0,0) – (0,1,0) = (0,-1,0)

w(ugly) = w(ugly) = (0,0,0)

Initial State:

w(good) = (0,0,0) => (0,1,0)

w(bad) = (0,0,0) => (0,-1,0)

w(ugly) = (0,0,) => (0,0,0)

x2: y2 = bad

Score(good) = w(good) * x2 = (0,1,0) * (1,0,1) = 0

Score(bad) = w(bad) * x2 = (0,-1,0) * (1,0,1) = 0

Score(ugly) = w(ugly) * x2 = (0,0,0) * (0,0,0) = 0

So, initial state and score does not change.

x3: y3 = ugly

Score:

Score(good) = w(good) * x3 = (0,1,0) * (1,1,1) = +1

Score(bad) = w(bad) * x3 = (0,-1,0) * (1,1,1) = -1

Score(ugly) = w(ugly) * x3 = (0,0,0) * (1,1,1) = 0

So,

w(good) = w(good) – x3 = (0,1,0) – (1,1,1) = (-1,0,-1)

w(bad) = w(bad) = (0,-1,0)

w(ugly) = w(ugly) + x3 = (0,0,0) + (1,1,1) = (1,1,1)

Initial State:

w(good) = (0,0,0) => (0,1,0) => (-1,0,-1)

w(bad) = (0,0,0) => (0,-1,0) == (0,-1,0)

w(ugly) = (0,0,0) => (1,1,1)

x4: y4 = bad

Score:

Score(good) = w(good) * x4 = (-1,0,-1) * (1,0,0) = -1

Score(bad) = w(bad) * x4 = (0,-1,0) * (1,0,0) = 0

Score(ugly) = w(ugly) * x4 = (1,1,1) * (1,0,0) = +1

So,

w(good) = w(good) = (-1,0,-1)

w(bad) = w(bad) + x4 = (0,-1,0) + (1,0,0) = (1,-1,0)

w(ugly) = w(ugly) – x4 = (1,1,1) – (1,0,0) = (0,1,1)

Initial State:

w(good) = (0,0,0) => (0,1,0) => (-1,0,-1) == (-1,0,-1)

w(bad) = (0,0,0) => (0,-1,0) => (1,-1,0)

w(ugly) = (0,0,0) => (1,1,1) => (0,1,1)

x5: y5 = good

Score:

Score(good) = w(good) * x5 = (-1,0,-1) * (0,0,1) = -1

Score(bad) = w(bad) * x5 = (1,-1,0) * (0,0,1) = 0

Score(ugly) = w(ugly) * x5 = (0,1,1) * (0,0,1) = +1

So,

w(good) = w(good) + x5 = (-1,0,-1) + (0,0,1) = (-1,0,0)

w(bad) = w(bad) = (1,-1,0)

w(ugly) = w(ugly) – x5 = (0,1,1) – (0,0,1) = (0,1,0)

Hence, after weight:

w(good) = (-1,0,0)

w(bad) = (1,-1,0)

w(ugly) = (0,1,0)

Q2. Suppose you are given the following binary classification training data, where each input example has three features and output label takes a value good or bad.

• x_1=(0, 1, 0) and y_1=good

• x_2=(1, 0, 1) and y_2=bad

• x_3=(1, 1, 1) and y_3=good

• x_4=(1, 0, 0) and y_4=bad

• x_5=(0, 0, 1) and y_5=good

Suppose we want to learn a classifier using kernelized perceptron algorithm. Start from the following dual weights: α_1=0; α_2=0; α_3=0; α_4=0; and α_5=0. Please do hand calculations to show how dual weights change after processing examples in the same order (i.e., one single pass over the five training examples). Do this separately for the following kernels: (a) Linear kernel: $K(x, x') = x \cdot x'$; and (b) Polynomial kernel with degree 3: $K(x, x') = (x \cdot x' + 1)^3$, where $x \cdot x'$ stands for dot product between two inputs x and x'.

(a) Linear Kernel:

Using linear combination of inner products formula:

$$f(x) = \sum_{i \in I} \alpha_i \langle \phi(x_i), \phi(x) \rangle = \sum_{i \in I} \alpha_i k(x_i, x)$$

x_1: y_1=good:

f(x) = $\sum_{i=1}^{5} 0 * ((0,1,0) * (0,1,0)) = 0$

$$\alpha 1 = \alpha 1 + y1 = 0 + 1 = 1$$

f(x) = $\sum_{i=1}^{5} 1 * ((0,1,0) * (1,0,1)) = 0 == \alpha 2$

So, $\alpha = [1,0,0,0,0]$

x2: y2=bad

f(x) = $\sum_{i=2}^{5} 0 * ((1,0,1) * (1,0,1)) = 0$

$$\alpha 2 = \alpha 2 + y2 = 0 + 0 = 0$$

f(x) = $\sum_{i=2}^{5} 0 * ((1,0,1) * (0,1,0)) = 0$ # NO EFFECTIVE

So, $\alpha = [1,0,0,0,0]$

x3: y3=good

f(x) = $\sum_{i=3}^{5} 0 * ((1,1,1) * (1,1,1)) = 0$

$$\alpha 3 = \alpha 3 + y3 = 0 + 1 = 1$$

So, $\alpha = [1,0,1,0,0]$

x4: y4=bad

f(x) = $\sum_{i=4}^{5} 0 * ((1,0,0) * (1,0,0)) = 0$

$$\alpha 4 = \alpha 4 + y4 = 0 + 0 = 0$$

f(x) = $\sum_{i=4}^{5} 0 * ((1,0,0) * (0,0,1)) = 0$ # NO EFFECTIVE

So, $\alpha = [1,0,1,0,0]$

x5: y5=good

f(x) = $\sum_{i=5}^{5} 0 * ((0,0,1) * (0,0,1)) = 0$

$$\alpha 5 = \alpha 5 + y5 = 0 + 1 = 1$$

f(x) = $\sum_{i=5}^{5} 1 * ((0,0,1) * (1,0,0)) = 0 == \alpha 4$

Hence, $\alpha = [1,0,1,0,1]$

(b) Polynomial kernel with degree 3:

Using Polynomial kernel with degree 3 of inner products formula:

$$f(x) = \sum_{i \in I} \alpha_i \langle \phi(x_i), \phi(x) \rangle = \sum_{i \in I} \alpha_i k(x_i, x + 1)^3$$

x1: y1=good

f(x) = $\sum_{i=1}^{5} 0 * ((0,1,0) * (0,1,0) + 1)^3 = 0$

$$\alpha 1 = \alpha 1 + y1 = 0 + 1 = 1$$

f(x) = $\sum_{i=1}^{5} 1 * ((0,1,0) * (1,0,1) + 1)^3 = 0 == \alpha 2$

So, $\alpha = [1,0,0,0,0]$

x2: y2=bad

f(x) = $\sum_{i=2}^{5} 0 * ((1,0,1) * (1,0,1) + 1)^3 = 0$

$$\alpha 2 = \alpha 2 + y2 = 0 + 0 = 0$$

f(x) = $\sum_{i=2}^{5} 0 * ((1,0,1) * (0,1,0) + 1)^3 = 0$ # NO EFFECTIVE

So, $\alpha = [1,0,0,0,0]$

x3: y3=good

f(x) = $\sum_{i=3}^{5} 0 * ((1,1,1) * (1,1,1) + 1)^3 = 0$

$$\alpha 3 = \alpha 3 + y3 = 0 + 1 = 1$$

So, $\alpha = [1,0,1,0,0]$

x4: y4=bad

f(x) = $\sum_{i=4}^{5} 0 * ((1,0,0) * (1,0,0) + 1)^3 = 0$

$$\alpha 4 = \alpha 4 + y4 = 0 + 0 = 0$$

f(x) = $\sum_{i=4}^{5} 0 * ((1,0,0) * (0,1,1) + 1)^3 = 0$ # NO EFFECTIVE

So, $\alpha = [1,0,1,0,0]$

x5: y5=good

f(x) = $\sum_{i=5}^{5} 0 * ((0,0,1) * (0,0,1) + 1)^3 = 0$

$$\alpha 5 = \alpha 5 + y5 = 0 + 1 = 1$$

f(x) = $\sum_{i=5}^{5} 1 * ((0,0,1) * (1,0,0) + 1)^3 = 0 == \alpha 4$

Hence, $\alpha = [1,0,1,0,1]$

Q3. Suppose x = ($x_1, x_2, \cdots, x_d$) and z = ($z_1, z_2, \cdots, z_d$) be any two points in a high-dimensional space (i.e., d is very large). Suppose you are given the following property, where the right-hand side quantity represents the standard Euclidean distance.

$$\left(\frac{1}{\sqrt{d}}\sum_{i=1}^{d} x_i - \frac{1}{\sqrt{d}}\sum_{i=1}^{d} z_i\right)^2 \leq \sum_{i=1}^{d}(x_i - z_i)^2 \qquad \textbf{- Equation (1)}$$

We know that the computation of nearest neighbors is very expensive in the high-dimensional space. Discuss how we can make use of the above property to make the nearest neighbors computation efficient?

The calculation of the nearest neighbor in high-dimensional space is very expensive. Looking at the formula mentioned here, it can be seen that the square distance between point x and point z is equal to or smaller than the distance between point x and point z in the high-dimensional space. Therefore, using this equation, I think it is efficient to calculate the nearest neighbor by first averaging the feature values of each point and then sorting these values.

Q4. We know that we can convert any decision tree into a set of if-then rules, where there is one rule per leaf node. Suppose you are given a set of rules R = {r_1,r_2,„r_k}, where r_i corresponds to the i^{th} rule. Is it possible to convert the rule set R into an equivalent decision tree? Explain your construction or give a counterexample.

Yes, a given rule set R can be reconstructed into an equivalent decision tree.

Counterexample:

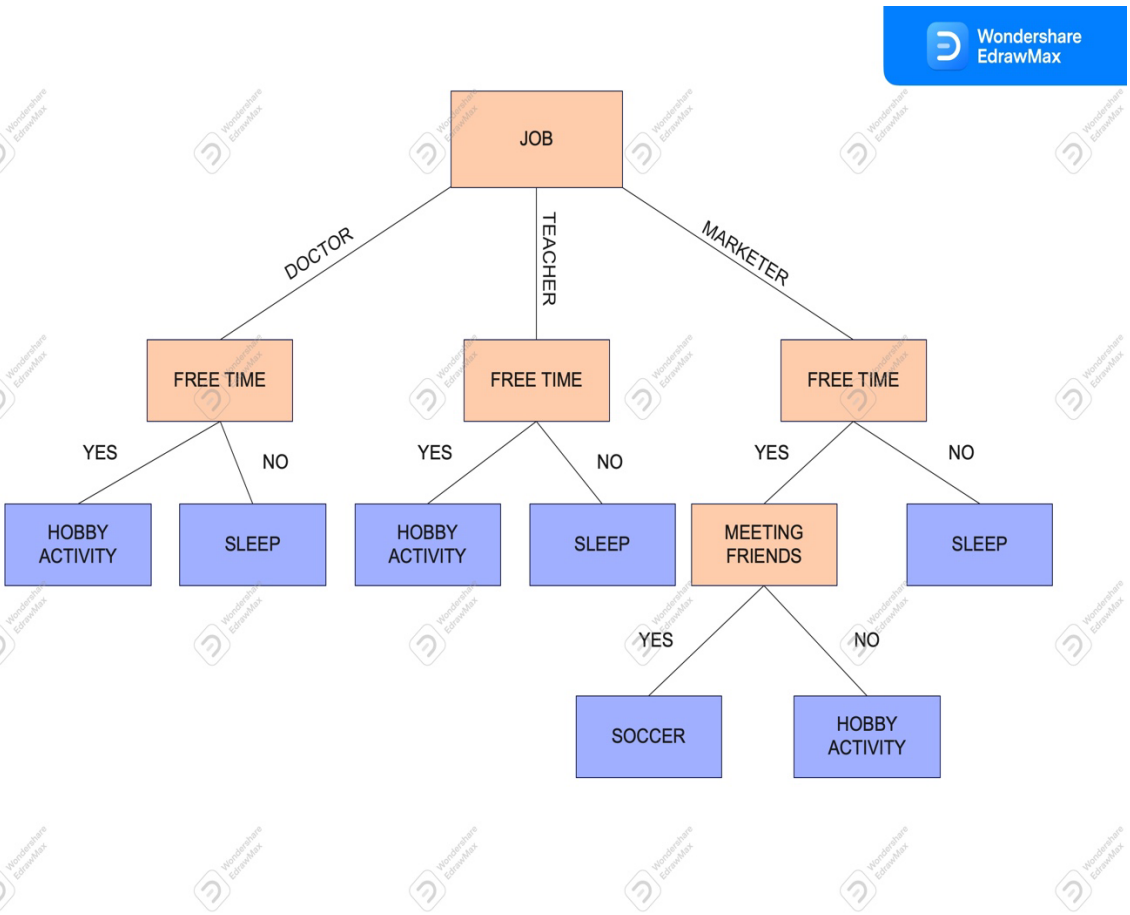
Job: Doctor, Teacher, Marketer

Free Time: Yes, No

Meeting Friends: Yes, No --- Convert the rule

Decision: (1) Hobby Activity (2) Sleep (3) Soccer

Making example decision tree using EdrawMax tool:



Q5. Please read the following two papers and write a brief summary of the main points in at most THREE pages.

Article 1: Hidden Technical Debt in Machine Learning Systems

This article refers to the technology debt that exists in machine learning systems. Machine learning is used in many societies as a toolkit that can quickly build complex prediction systems. But even with such a powerful system, there are problems. The system is relatively fast and inexpensive to develop and deploy, but over time it is difficult and costly to maintain, resulting in technical debt. A major factor in these technology liabilities is data-dependent liabilities. Unstable data dependencies are convenient to use signals as input functions generated by other systems, but they compromise the suitability of the model by destabilizing some input signals while varying behavior qualitatively or quantitatively over time. To alleviate this unstable data dependence, it is to make a copy of a given signal.

There is something we need to check to prevent this unstable data dependency. It is a legacy function, bundle function, feature, and correlated characteristics. By using these functions, safer results can be obtained by identifying and eliminating unnecessary functions.

Another area where debt can accumulate is the construction of machine learning systems. The configuration of the system has a wide range of options and the number of lines in this configuration can demonstrate the possibility of mistakes by exceeding the number of lines in the code. In addition, misconfiguration of these systems can be costly, leading to severe time loss, waste of computing resources, and production problems. Therefore, you should be familiar with the principles for a good configuration system.

These ML systems often interact directly with the outside world. However, many ML systems are designed to adapt over time, so invariants are not always clear. Therefore, it is

very important to conduct comprehensive real-time monitoring of ML system behavior and long-term system reliability.

In conclusion, ML systems result in technical debt by not providing strict metrics that can be tracked over time. Therefore, it is necessary to determine whether the impact of new changes to the system can be accurately measured and consider how easy it can be to test them altogether. As there are many problems in using the ML system, it is most important for engineers and researchers to develop insight.

Article 2: The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction

This article talks about the requirements for ML system testing and monitorin g. Currently, the reliability problem is becoming increasingly important as machine lea rning systems play a central role in the actual production environment. Therefore, testi ng and monitoring of machine learning systems is a key consideration for ensuring th e production readiness of ML systems and reducing technical debt.

The characteristics of the test on the data should be identified before the mod el is submerged. Functional expectations must first be identified in the schema. This i s useful to compare with data to avoid fixed bias. And you need to understand the v alue each feature provides in additional predictive capabilities. At this time, there is a method of calculating the correlation coefficient or removing one of the functions in dividually. Because the cost of using these features is daunting, it is important to bud get enough time while developing new features for proper processing.

The machine learning infrastructure must then be tested. First, deterministic tra ining that dramatically simplifies reasoning about the entire system should be reproduc ed. And we need to test the model specification code. This test is necessary because there may be bugs in the model file. And by conducting a pipeline integration test, it

is necessary to verify that the data and code are successfully moved by vomiting ea ch step. Finally, before attempting to service a model, it is necessary to train the mo del and ensure that the quality of the model is sufficient. These models are tested thr ough the canary process before entering the service environment. At this time, there i s no guarantee that it will be carried out well, so you should always take some risks. And you have to practice rollback in case the model doesn't work properly.

These model systems may not work well over time, so it is important to ensu re that they work properly periodically. At this time, an initial test is performed to en sure that it is safe and easy. First, it is necessary to check the change in dependency and ensure that the data invariant is maintained when entering training and services. I t is also necessary to ensure that training and service functions generate the same val ue. Problems can arise if these two values are not the same. In addition, the model s hould be updated frequently. This is because maintenance costs are incurred even for models that are not frequently updated. In addition, it is necessary to ensure that the model is numerically stable and to know how to respond quickly to performance chan ges caused by infrastructure changes. Finally, three methods can be used to ensure tha t the service prediction quality of the model does not degrade. The first is to measur e the predicted average of a particular piece of data. The second is to enable labels t o be immediately predictable in some tasks. The third is to periodically add new data to training data while manually annotating labels for recorded service inputs from pe ople.

In conclusion, there are many difficulties in getting machine learning models t o actually use. Therefore, people in ML-related fields should follow sufficient procedu res using a roadmap. Although this process may not be easy, it can improve the sust ainability of the model and minimize the cost of maintenance.