**Machine Learning Exam**

**Name: Chitradevi Maruthavanan**

**PSU\_ID: 950828319**

1. (6 points) An online retailer like Amazon wants to determine which products to promote

based on reviews. They only want to promote products that are likely to sell. For each

product, they have past reviews. The reviews have both a numeric score (from 1 to 5) and

text.

(a) To formulate this as a machine learning problem, suggest a target variable that the online

retailer could use.

**To formulate this as a machine learning problem, the online retailer could use a binary target variable indicating whether a product is likely to sell or not based on past reviews. This target variable could be represented as 1 for products that have a high probability of being purchased and 0 for those with a low probability of being purchased.**

(b) For the predictors of the target variable, a data scientist suggests to combine the numeric

score with frequency of occurrence of words that convey judgement like “bad", “good",

and “doesn't work." Describe a possible linear model for this relation.

**probability of product to sell =**  **β0+ β1[numeric score]+ β2[frequency of “bad”]+ ]+ β3[frequency of “good”]+ ]+ β4[frequency of “doesn’t work”]**

**y** ≈𝒚ˆ **=** **β0+ β1x1+ β2 x2 + β3 x3+ β4 x4**

(c) Now, suppose that some reviews have a numeric score from 1 to 5 and others have a score

from 1 to 10. How would this change your features?

**If some reviews have a numeric score from 1 to 5 and others have a score from 1 to 10, it will affect the feature related to the numeric score. The feature related to the frequency of occurrence of words would remain the same. To handle this difference, we would need to normalize the scores to make them comparable. We can do this by using a range of 0 to 1. Then we can divide the 1-5 score range by 5 and the 1-10 score range by 10 so that they will be in a normalized range of 0 to 1**

2. (3 points) In Support Vector Machine optimization problem, parameter C provides a balance between the hinge loss and margin. Discuss the impact of large or small C on model over/under fitting and how an ML engineer can find the optimal C?

**In Support Vector Machine (SVM) optimization, the parameter C balance between the hinge loss and the inverse of margin. A small value of C allows for a larger margin but can result in misclassifying some training examples, leading to underfitting. A large value of C minimizes the hinge loss at the expense of a smaller margin, which can lead to overfitting the training data. If the value of C is too small, the model will underfit and perform poorly on both the training and test data. If the value of C is too large, the model will overfit and perform well on the training data but poorly on the test data. Therefore, it is essential to choose an appropriate value of C that balances the trade-off between underfitting and overfitting.**

**To find the optimal value of C, an ML engineer can use cross-validation. Therefore, to correctly select the parameter C, we need to identify through cross-validation.**

3. (3 points) Consider a dataset with five dimensions where each eigenvalue is equal to two.

Plot the proportion of variance as a function of number of eigenvalues? Is dimensionality reduction a good tool to reduce the data dimension and if so, how many dimensions would you choose for dimensionality reduction?

**Since all eigenvalues are equal to 2 this means that each eigenvalue represents an equal proportion of the variance.**

|  |  |
| --- | --- |
| **Number of Eigen values** | **Proportion of variance** |
| **1** | **0.2** |
| **2** | **0.4** |
| **3** | **0.6** |
| **4** | **0.8** |
| **5** | **1.0** |

**It can be plotted like this:**

**y^ = 0.2x**

**Chart, line chart

Description automatically generated**

**Dimensionality reduction seems like a good tool to use, as we can reduce the dimension of the dataset while still retaining the same variation in data since all eigenvalues are equal. We could use a single dimension to accurately represent our data in this case.**

4. (3 points) Let X be a dataset with number of rows corresponding to different samples and

number of columns corresponding to different features. Let Y be the target variable vector.

Write a few lines of code that gets a 70 to 30 train vs test split of the data and further removes the mean from each column and appropriately scales the data?

**X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.3)**

**X\_scaler = StandardScaler()**

**Y\_scaler = StandardScaler()**

**X\_train1 = X\_scaler.fit\_transform(X\_train)**

**Y\_train1 = Y\_scaler.fit\_transform(Y\_train[:,None])**

**X\_test1 = X\_scaler.transform(X\_test)**

**Y\_test1 = Y\_scaler.transform(Y\_test[:,None])**

5. (6 points) In this problem, we will look at how to exhaustively search over subsets of features.

You are given three python functions:

model = LinearRegression() # Create a linear regression model object

model.fit(X,y) # Fits the model

yhat = model.predict(X) # Predicts targets given features

Given training data Xtr,ytr and test data Xts,yts, write a few lines of python code to:

(a) Find the best model using only one feature of the data (i.e. one column of Xtr and Xts).

**from sklearn.linear\_model import LinearRegression**

**import numpy as np**

**Xtr, ytr = np.load('Xtr.npy'), np.load('ytr.npy')**

**Xts, yts = np.load('Xts.npy'), np.load('yts.npy')**

**model = LinearRegression()**

# Find the best model using only one feature

**best\_feature, best\_score = None, -np.inf**

**for i in range(Xtr.shape[1]):**

# Select a single feature

**Xtr\_i = Xtr[:, i].reshape(-1, 1)**

**Xts\_i = Xts[:, i].reshape(-1, 1)**

# Fit the model on the training data

**model.fit(Xtr\_i, ytr)**

# Predict targets for the test data

**yhat = model.predict(Xts\_i)**

# Compute the R-squared score

**score = model.score(Xts\_i, yts)**

# Update the best feature and score

**if score > best\_score:**

**best\_feature, best\_score = i, score**

**print(f"Best feature: {best\_feature}, R-squared score: {best\_score}")**

(b) Find the best model using only two features of the data (i.e. two columns of Xtr and Xts).

# Find the best model using only two features

**best\_features, best\_score = None, -np.inf**

**for i in range(Xtr.shape[1]):**

**for j in range(i+1, Xtr.shape[1]):**

**Xtr\_ij = Xtr[:, [i,j]]**

**Xts\_ij = Xts[:, [i,j]]**

**model.fit(Xtr\_ij, ytr)**

# Predict targets for the test data

**yhat = model.predict(Xts\_ij)**

# Compute the R-squared score

**score = model.score(Xts\_ij, yts)**

# Update the best features and score

**if score > best\_score:**

**best\_features, best\_score = (i,j), score**

**print(f"Best features: {best\_features}, R-squared score: {best\_score}")**

6. Answer:

**Compute the gradient ∂f (x)/∂x of the function**

**We have the below function which is of degree 2**

**f (x) = x1 + x2x3.**

**Taking partial derivatives, with respect to x1,x2 and x3 we get the below**

**∂f (x)/∂x1 = 1**

**∂f (x)/∂x2 = x3**

**∂f (x)/∂x3 = x2**

**For vector input W, and scalar function f(w)the gradient is**

**∇f (w)= [∂f (w)/∂w1……. ∂f (w)/∂wN]**

**Therefore, we get the below for our question**

**the gradient of the function f (x) is:**

**∇f (x) = [∂f (x)/∂x1, ∂f (x)/∂x2, ∂f (x)/∂x3] = [1, x3, x2]**

**Since gradient is same size as the argument, and we have 3 arguments in our input function we get the below**

**Size of gradient vector = 3**

7. a) Answer

**import matplotlib.pyplot as plt**

**# Define the data**

**income = [3, 5, 7, 8, 10]**

**num\_websites = [0, 1, 1, 2, 1]**

**donate = [0, 1, 0, 1, 1]**

**# Create the scatter plot**

**plt.scatter(income, num\_websites, c=donate, cmap='coolwarm')**

**plt.xlabel('Income (tens of thousands $)')**

**plt.ylabel('Number of websites with similar political views')**

**plt.title('Donations')**

**plt.colorbar(label='Donated')**

**plt.show()**

Chart, scatter chart

Description automatically generated

7. b) Answer

**Zi = wTxi + b**

**Z1 = 3w1 + 0 w2 + b < 0**

**Z2 = 5 w1 + w2+ b > 0**

**Z3 = 7 w1 + w2 + b < 0**

**Z4 = 8 w1+ 2 w2+ b > 0**

**Z5 = 10 w1+ w2 + b > 0**

**Consider,**

**W = [1, 1] and b = - 4**

**Zi = [1,1]Txi + b**

**Zi = x1 + x2 - 4**

**Case 1:**

**Zi = x1 + x2 – 4**

**Z1 = 3 + 0 – 4**

**Z1 = – 1 < 0 (true)**

**Case 2:**

**Z2 = 5 w1 + w2+ b**

**Z2 = 5 + 1 -4**

**Z2 = 2 > 0 (true)**

**Case 3:**

**Z3 = 7 w1 + w2 + b**

**Z3 = 7 + 1 -4**

**Z3 = 4 > 0 (false)**

**Case 4:**

**Z4 = 8 w1+ 2 w2+ b**

**Z4 = 8 + 2 -4**

**Z4 = 6 > 0 (true)**

**Case 5:**

**Z5 = 10 w1+ w2+ b**

**Z5 = 10 + 1 -4**

**Z5 = 7 > 0 (true)**

**Hence w1 = 1, w2 = 1 and b = -4 works for all the points, but we got one error in the case 3.**

7. c) Answer

**Using the logistic model, we have:**

**P(yi=1|xi) = 1 / (1 + e-zi)**

**Using w = [1, 1] and b = -4 from the previous part, we can calculate Zi for each sample:**

**Z1 = wT x1 + b = (1)(3) + (1)(0) - 4 = -1**

**z2 = wT x2 + b = (1)(5) + (1)(1) - 4 = 2**

**z3 = wT x3 + b = (1)(7) + (1)(1) - 4 = 4**

**z4 = wT x4 + b = (1)(8) + (1)(2) - 4 = 6**

**z5 = wT x5 + b = (1)(10) + (1)(1) - 4 = 7**

**Now we can calculate the probability of each sample:**

**P(y1=1|x1) = 1 / (1 + e^1) = 0.2689**

**P(y2=1|x2) = 1 / (1 + e^-2) = 0.8808**

**P(y3=1|x3) = 1 / (1 + e^-4) = 0.9820**

**P(y4=1|x4) = 1 / (1 + e^-6) = 0.9975**

**P(y5=1|x5) = 1 / (1 + e^-7) = 0.9991**

**Therefore, sample 1 (x1 = 3 and x2 = 0) has the smallest probability and is thus the least likely to donate according to this logistic model.**