Homework 6 - MATH 4322

Phu Nguyen

Fall 2021

Problem 1

You are given:

- n data samples $\mathbf{x}_i = (x_{1,i}, \cdots, x_{p,i}), i = 1, \ldots, n$
- n corresponding to true responses (or labels) $y_i, i = 1, \ldots, n$.

and asked to train a single linear neuron "network" to approximate function $f(.)$ such that $f(\mathbf{x}_i) = y_i, i = 1, \ldots, n$. Provided the train steps for your "network" by answering the following questions.

a) What is the formula to calculate an output $\hat{y}_i$ from an input $\mathbf{x}_i$? What are the model parameters in that formula?

$b + \sum_{i=1}^{n} \hat{y}_i x_i$

b) What criteria do we need to optimize in order to estimate the model parameters?

- the weight

c) What is the name of the method used to optimize this criteria in case you do not have access to an analytical solution?

Problem 2

Presume that for a single linear neuron model with input variables $x_1, \ldots, x_5$, you are given the following parameter values:

- weights: $w_1 = 0.2, w_2 = -0.54, w_3 = -0.21, w_4 = -0.1, w_5 = 0.33$,
- bias: $b = 0.14$.

a) Draw a mathematical model of this linear neuron that taks an arbitrary input vector $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$.

$y = f(x_1, x_2, x_3) = b + \sum_{i=1}^{5} x_i w_i$

b) Calculate the linear neuron output for the case of $x_1 = 4, x_2 = -3, x_3 = 7, x_4 = 5, x_5 = -1$. Show your work.

$y = 0.14 + \sum_{i=1}^{5} (4 * 0.2) + (-3 * -0.54) + (7 * -0.21) + (5 * -0.1) + (-1 * 0.33) = 0.26$

Problem 3

You are given an artificial neural network (ANN) of linear neurons with

- Input layer of two neurons: x_1, x_2
- Fully-connected hidden layer of three neurons: h_1, h_2, h_3
- One output neuron, y.

The following weight matrices are provided:

1) Between input & hidden layer:

		Hidden h_1	h_2	h_3
	1 (bias)	-0.3	0.5	0.5
Input	x_1	0.6	-0.4	0.5
	x_2	-0.7	-0.3	0.2

2) Between hidden & output layer:

		Output y
	1 (bias)	0.2
Hidden	h_1	-0.3
	h_2	0.5
	h_3	-0.7

a) Draw this ANN as was done in lecture slides.
b) Calculate the output of this ANN for the case of $x_1 = 10, x_2 - -5$. Show work.

Problem 4

- We want to predict the 'medv' value based on the input of the other thirteen variables.
- We will run a regression neural network for the `Boston` dataset.
- We will split the data into training/testing by a 70/30 split.

a) Type and run the following in `R`.

```
library(neuralnet)
library(MASS)

data = Boston #renaming the Boston dataset to "data"
summary(data)
```

What is the mean of `age`? What is the mean of `ptratio`?

b) Normalizing data

- It is recommended to **normalize** (or scale, or standardize, either works) features in order for all the variables to be on the same scale.

- With normalization, data units are eliminated, allowing you to easily compare data from different locations.
- This avoids unnecessary results or difficult training processes resulting in algorithm convergence problems.
- There are different methods for scaling the data.
- The **z-normalization**

$$x_{scale} = \frac{x - \bar{x}}{s}$$

- The **min-max scale**

$$x_{scale} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- And so forth
- The function in `R` is scale(x,center = ,scale =)
- For this example we will use the min-max method to get all the scaled data in the range [0, 1].
- In order to scale we need to find the minimum and maximum value for each of the columns in the data set. To do this we use the apply function.
- The `apply` function returns a vector or an array or a list of values obtained by **applying** a function to margins of an array or matrix.

- Type and run the following:

```
max_data = apply(data,2,max)
#2=columns, we are getting the maximum value from each column
min_data = apply(data,2,min)
data_scaled = scale(data, center = min_data, scale = max_data - min_data)
head(data_scaled)
```

What is the scaled value of the first observation for `medv`?

c) Now we can split the data into training and testing data sets. We will use the 70/30 split

```
set.seed(10)
index = sample(1:nrow(data), round(0.7*nrow(data)))
train_data = as.data.frame(data_scaled[index,])
test_data = as.data.frame(data_scaled[-index,])
dim(train_data)
```

How many observations do we have in the training data set?

d) Type and run the following

```
set.seed(1)
net_data = neuralnet(medv ~ ., data = train_data,
                     hidden = 10, linear.output = TRUE)

plot(net_data)
```

Apply the test dataset to determine the MSE

```
predict_net = predict(net_data,test_data)
predict_net_start = predict_net*(max(data$medv) - min(data$medv)) + min(data$medv)
test_data_start = test_data$medv*(max(data$medv) - min(data$medv)) + min(data$medv)
sum((predict_net_start - test_data_start)^2)/nrow(test_data)
```

What is the test MSE for this model?

e) Let us compare this test MSE to the linear regression model. Type and run the following:

```
lm.boston = lm(medv ~ ., data = data, subset = index )
summary(lm.boston)
test = data[-index,]
predict_lm = predict(lm.boston,test)
sum((predict_lm - test$medv)^2)/nrow(test)
```

What is the training MSE for the linear model?