UC Irvine, Division of Continuing Education

Deep Learning Using TensorFlow

Spring 2019

Homework#3

Date Given: April 22, 2019 Due Date: April 28, 2019

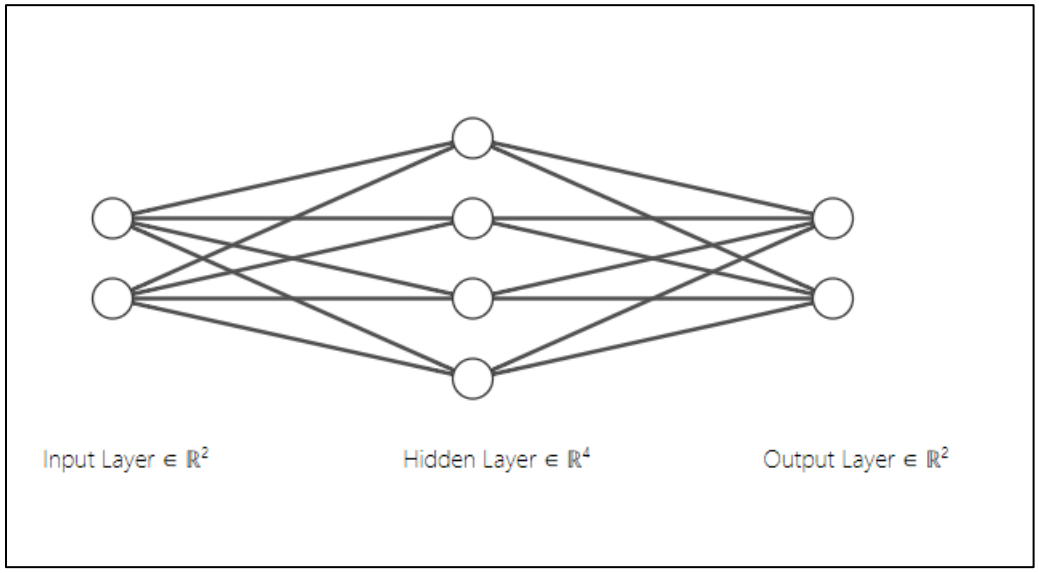
==========

There are 4 problems in this homework assignment.

==========

Problem#1

Compute the output of a Neural Network model using TensorFlow software with the following specifications.



Layer#	Layer	# of neurons	Initial values
0	Input Layer	2	100, 150
1	Hidden layer#1	4	
2	Output layer	2	

Activation function used: $Linear: y = x$.

Compute the values of y_1 and y_2 generated by the output layer.

Weights Value

The following convention is used in identifying the weight values.

- $w_{Neuron\#\ Neuron\#}^{Layer\#}$
- Example: w_{11}^{1}: Layer#1, From Neuron#1 of Input Layer, to Neuron#1 of Hidden Layer#1

==========

Weight values between the Input Layer and the Hidden Layer#1.

$w_{11}^{1} = 5$	$w_{12}^{1} = 10$	$w_{13}^{1} = 15$	$w_{14}^{1} = 20$
$w_{21}^{1} = 25$	$w_{22}^{1} = 30$	$w_{23}^{1} = 35$	$w_{24}^{1} = 40$

Bias values in Hidden layer#1

$b_{1}^{1} = 17$	$b_{2}^{1} = 19$	$b_{3}^{1} = 21$	$b_{4}^{1} = 23$

==========

Weight values between the Hidden Layer#1 and the Output Layer.

$w_{11}^{2} = 30$	$w_{12}^{2} = 35$
$w_{21}^{2} = 40$	$w_{22}^{2} = 45$
$w_{31}^{2} = 70$	$w_{32}^{2} = 75$
$w_{41}^{2} = 80$	$w_{42}^{2} = 85$

Bias values in Output layer.

$b_{1}^{2} = 35$	$b_{2}^{2} = 36$

==========

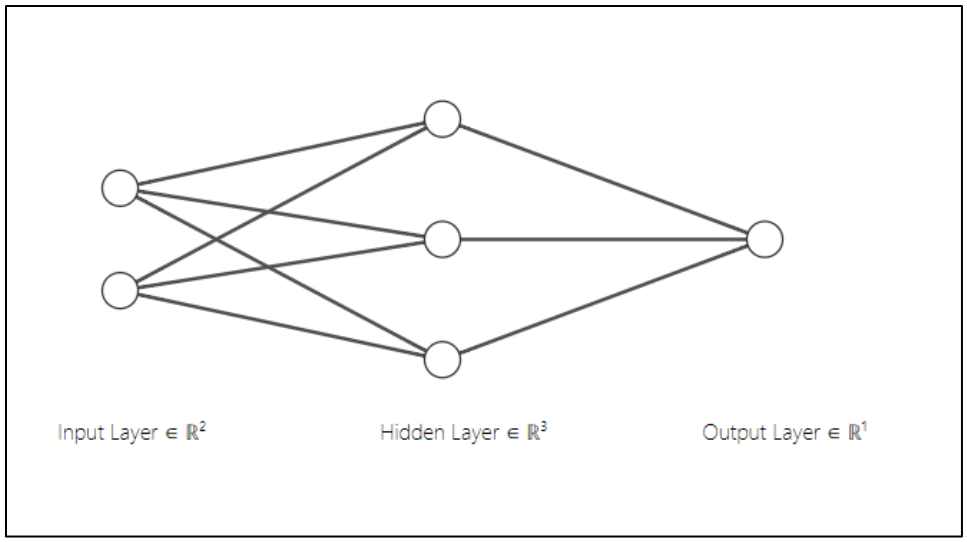
Use only the feed-forward method. Using these weight values, compute the value of the output layer $y_1\ and\ y_2$.

Answer:

- $y_1 = 1{,}464{,}615,\ y_2 = 1{,}587{,}516$
- Make sure that the answer generated by your TensorFlow code matches with the given answer.
- Please include your TensorFlow code with your submission.

Problem#2

Compute the output of a Neural Network model using TensorFlow software with the following specifications.



Training Data (XOR Gate data):

Input 1	Input 2	Output
0	0	0
1	0	1
0	1	1
1	1	0

Layer#	Layer	# of neurons
0	Input Layer	2
1	Hidden layer#1	3
2	Output layer	1

Activation function used: $sigmoid\ f(x) = \frac{1}{1+e^{-x}}$

After running the neural network for 100 epochs, the following weights and biases were computed.

The following convention is used in identifying the weight values.

- $w_{Neuron\#\ Neuron\#}^{Layer\#}$
- Example: w_{11}^{1}: Layer#1, From Neuron#1 of Input Layer, to Neuron#1 of Hidden Layer#1

==========

w_{11}^{1}	w_{12}^{1}	w_{13}^{1}
w_{21}^{1}	w_{22}^{1}	w_{23}^{1}

```
W1 values
[[-4. -6. -5.]
 [ 3.  6.  4.]]
```

Bias values in Hidden layer#1

b_1^1	b_2^1	b_3^1

```
Bias b1 Values
[-2.  3. -2.]
```

==========

w_{11}^{2}
w_{21}^{2}
w_{31}^{2}

```
W2 values
[[ 5.]
 [-9.]
 [ 7.]]
```

Bias values in Output layer

b_1^2

```
Bias b2 Values
[ 4.]
```

==========

Compute the error for all 4 input values.

$Error = (Computed\ Output - True\ Output)^2$

Input 1	Input 2	True Output	Computed output	Error
0	0	0	?	?
1	0	1	?	?
0	1	1	?	?
1	1	0	?	?

Answer:

Input 1	Input 2	True Output	Computed output	Error
0	0	0	0.04137863	0.00171219
1	0	1	0.97319275	0.00071863
0	1	1	0.99201351	6.37839548e-05
1	1	0	0.01791469	0.00032094

Answer:

- Make sure that the answer generated by your TensorFlow code matches with the given answer.
- Please include your TensorFlow code with your submission.

Problem#3

The 'Admission.csv' data file has the following information about 400 students.

	A	B	C	D
1	admit	gre	gpa	rank
2	0	380	3.61	3
3	1	660	3.67	3
4	1	800	4	1
5	1	640	3.19	4
6	0	520	2.93	4
7	1	760	3	2
8	1	560	2.98	1
9	0	400	3.08	2
10	1	540	3.39	3

The feature 'gre' represents the GRE examination results of students. The 'gpa' feature represents the GPA of students. The 'rank' represents the rank of the university (4 is the best, 1 is the lowest). The 'admit' feature is a categorical variable which indicates if that student was admitted into college (0 means not admitted, 1 means admitted).

Build a Neural Network using TensorFlow software to predict if a student will be admitted to college given the 'gre', 'gpa', and 'rank' values.

Neural Network Specification

- Number of input node = 3 (gre, gpa, rank)
- Number of hidden nodes = 5 (vary this number from 1 to 10 to get highest accuracy of prediction)
- Number of output node = 2 (categorical: 0,1 means not admitted, 1,0 means admitted)
- Cost Function: Cross Entropy Cost Function
- Optimization Function: Gradient Descent
    - Feed "Learning Rate" as a parameter to the optimization function

Before building the Neural Network, scale the predictor variables values between 0 and 1. Otherwise Neural Network may not converge. Since there are 400 observations, split the dataset into training (70%) and testing (30%).

Train the Neural Network model using the training data. Predict the outcome using the testing dataset. Measure the accuracy of your predictions. In the end, print the accuracy, values of weights and bias of all the nodes of your Neural Network.

Problem#4

The 'Advertising.csv' contains the following information.

	A	B	C	D	E
1		TV	Radio	Newspaper	Sales
2	1	230.1	37.8	69.2	22.1
3	2	44.5	39.3	45.1	10.4
4	3	17.2	45.9	69.3	9.3
5	4	151.5	41.3	58.5	18.5
6	5	180.8	10.8	58.4	12.9
7	6	8.7	48.9	75	7.2
8	7	57.5	32.8	23.5	11.8
9	8	120.2	19.6	11.6	13.2
10	9	8.6	2.1	1	4.8
11	10	199.8	2.6	21.2	10.6

A company spends money on advertising their products on 3 media channels - TV, Radio and Newspapers. The above table shows the relationship between amount of money spent in advertising and sales. There are 200 observations in the 'Advertising.csv' file.

Build a Neural Network using TensorFlow software to predict the sales given the amount of money spent on all 3 the media channels.

The only difference between problem#1 and problem#2 is the type of the response variable. In problem#1 the response variable is categorical (with 2 levels). In problem#2 the response variable is numerical.

Neural Network Specification:

- Number of input node = 3 (TV, Radio, Newspaper)
- Number of hidden nodes = 5 (vary this number from 1 to 10 to get highest accuracy of prediction)
- Number of output node = 1 (Sales - numerical)
- Cost Function: (Observed output value – Computed output value)^2
- Optimization Function: Gradient Descent
    - Feed "Learning Rate" as a parameter to the optimization function

Before building the Neural Network, scale the predictor variables values between 0 and 1. Otherwise Neural Network may not converge. Since there are 200 observations, split the dataset into training (70%) and testing (30%).

Train the Neural Network model using the training data. Predict the outcome using the testing dataset. Measure the accuracy of your predictions by computing RMSE (Root Mean Square Error). In the end, print the RMSE, values of weights and bias of all the nodes of your Neural Network.

How to scale a variable value between 0 and 1.

```
################################################
# 1. Load the libraries
#
import pandas as pd
from sklearn import preprocessing
################################################
# 2. Read the Dataset

data = pd.read_csv('Admissions.csv')
data.head()
Out[10]:
   admit  gre   gpa  rank
0      0  380  3.61     3
1      1  660  3.67     3
2      1  800  4.00     1
3      1  640  3.19     4
4      0  520  2.93     4

features = data[['gre','gpa','rank']]

type(features)
Out[12]: pandas.core.frame.DataFrame

features[0:5]
Out[13]:
   gre   gpa  rank
0  380  3.61     3
1  660  3.67     3
2  800  4.00     1
3  640  3.19     4
4  520  2.93     4

featuresScale = preprocessing.minmax_scale(data[['gre','gpa','rank']])

type(featuresScale)
Out[15]: numpy.ndarray

featuresScale[0:5]
Out[16]:
array([[ 0.27586207,  0.77586207,  0.66666667],
       [ 0.75862069,  0.81034483,  0.66666667],
       [ 1.        ,  1.        ,  0.        ],
       [ 0.72413793,  0.53448276,  1.        ],
       [ 0.51724138,  0.38505747,  1.        ]])
```

```
#########################################################
# Second Method
#
featuresMin = features.min(axis=0)
featuresMax = features.max(axis=0)
featuresScale2 = (features - featuresMin)/(featuresMax - featuresMin)

featuresScale2[0:5]
Out[21]:
        gre       gpa      rank
0  0.275862  0.775862  0.666667
1  0.758621  0.810345  0.666667
2  1.000000  1.000000  0.000000
3  0.724138  0.534483  1.000000
4  0.517241  0.385057  1.000000
```