

PROGRAMMING ASSIGNMENT #2

T81-559: Applications of Deep Neural Networks, Washington University

August 24, 2016

Listing 1 shows a sample submission skeleton that you can use as a starting point for this assignment.

Listing 1: Sample Submission Skeleton

```
1 # Programming Assignment #2 by Jeff Heaton
2 # T81-558: Application of Deep Learning
3 import os
4 import sklearn
5 import pandas as pd
6 import numpy as np
7 import tensorflow.contrib.learn as skflow
8 from sklearn.cross_validation import KFold
9 from scipy.stats import zscore
10 from sklearn import metrics
11 from sklearn import preprocessing
12 from sklearn.cross_validation import KFold
13 from sklearn.cross_validation import train_test_split
14
15 path = "./data/"
16
17 # These four functions will help you, they were covered in class.
18 # Encode a text field to dummy variables
19 def encode_text_dummy(df,name):
20     dummies = pd.get_dummies(df[name])
21     for x in dummies.columns:
22         dummy_name = "{}-{}".format(name,x)
23         df[dummy_name] = dummies[x]
24     df.drop(name, axis=1, inplace=True)
25
26 # Encode a text field to a single index value
27 def encode_text_index(df,name):
28     le = preprocessing.LabelEncoder()
29     df[name] = le.fit_transform(df[name])
30     return le.classes_
31
32 # Encode a numeric field to Z-Scores
33 def encode_numeric_zscore(df,name,mean=None,sd=None):
34     if mean is None:
35         mean = df[name].mean()
36
```

```

37     if sd is None:
38         sd = df[name].std()
39
40     df[name] = (df[name]-mean)/sd
41
42 # Encode a numeric field to fill missing values with the median.
43 def missing_median(df, name):
44     med = df[name].median()
45     df[name] = df[name].fillna(med)
46
47 # Convert a dataframe to x/y suitable for training.
48 def to_xy(df,target):
49     result = []
50     for x in df.columns:
51         if x != target:
52             result.append(x)
53     return df.as_matrix(result),df[target]
54
55 # Encode the toy dataset
56 def question1():
57     print()
58     print("***Question 1***")
59
60     path = "./data/"
61
62     filename_read = os.path.join(path,"toy1.csv")
63     filename_write = os.path.join(path,"submit-jheaton-prog2q1.csv")
64     df = encode_toy_dataset(filename_read) # You just have to implement ←
        encode_toy_dataset above
65     df.to_csv(filename_write,index=False)
66     print("Wrote {} lines.".format(len(df)))
67
68
69 # Model the toy dataset, no cross validation
70 def question2():
71     print()
72     print("***Question 2***")
73
74 def question3():
75     print()
76     print("***Question 3***")
77
78     # Z-Score encode these using the mean/sd from the dataset (you got ←
        this in question 2)
79     testDF = pd.DataFrame([
80         {'length':1, 'width':2, 'height': 3},
81         {'length':3, 'width':2, 'height': 5},

```

```

82         {'length':4, 'width':1, 'height': 3}
83     ])
84
85
86 def question4():
87     print()
88     print("***Question 4***")
89
90
91 def question5():
92     print()
93     print("***Question 5***")
94
95
96 question1()
97 question2()
98 question3()
99 question4()
100 question5()

```

Listing 2 shows what the output from this assignment would look like. Your numbers might differ from mine slightly. Every question, except 2, also generates an output CSV file. For your submission please include your Jupyter notebook and any generated CSV files that the questions specified. Name your output CSV files something such as **submit-jheaton-prog2q1.csv**. Submit a ZIP file that contains your Jupyter notebook and 4 CSV files to Blackboard. This will be 5 files total.

Listing 2: Expected Output

```

1  ***Question 1***
2  Wrote 10000 lines.
3
4  ***Question 2***
5  Step #99, avg. train loss: 1428641.75000
6  ...Lines removed for space...
7  Step #10000, epoch #42, avg. train loss: 31899.88477
8  Out of sample (RMSE): 249.13940526005305
9
10 ***Question 3***
11 length: (5.5259, 2.8610396265648745)
12 width: (5.5336999999999996, 2.8597695953020104)
13 height: (5.5338000000000003, 2.8721215970510063)
14     height    length    width
15 0 -0.882205 -1.581907 -1.235659
16 1 -0.185856 -0.882861 -1.235659
17 2 -0.882205 -0.533338 -1.585338

```

18
19 ***Question 4***
20 Fold #1
21 Step #50, epoch #12, avg. train loss: 0.07184, avg. val loss: 0.05284
22 ...Lines removed for space...
23 Step #250, epoch #62, avg. train loss: 0.01217, avg. val loss: 0.01708
24 Stopping. Best step:
25 step 97 with loss 0.013468504883348942
26 Fold score (RMSE): 0.17531210760832266
27 Fold #2
28 Step #50, epoch #12, avg. train loss: 0.07019, avg. val loss: 0.04958
29 ...Lines removed for space...
30 Step #550, epoch #137, avg. train loss: 0.01109, avg. val loss: 0.01040
31 Fold score (RMSE): 0.14272974831031332
32 Fold #3
33 Stopping. Best step:
34 step 362 with loss 0.009098603390157223
35 Step #50, epoch #12, avg. train loss: 0.06759, avg. val loss: 0.06043
36 ...Lines removed for space...
37 Step #1300, epoch #325, avg. train loss: 0.00812, avg. val loss: 0.01476
38 Fold score (RMSE): 0.161749386296102
39 Fold #4
40 Stopping. Best step:
41 step 1106 with loss 0.011062948033213615
42 Step #50, epoch #12, avg. train loss: 0.06714, avg. val loss: 0.07087
43 ...
44 Step #600, epoch #150, avg. train loss: 0.00964, avg. val loss: 0.02152
45 Stopping. Best step:
46 step 438 with loss 0.020158855244517326
47 Fold score (RMSE): 0.21273141903164636
48 Fold #5
49 Step #50, epoch #12, avg. train loss: 0.06887, avg. val loss: 0.05595
50 ...
51 Step #950, epoch #237, avg. train loss: 0.00902, avg. val loss: 0.01661
52 Stopping. Best step:
53 step 786 with loss 0.011288278736174107
54 Fold score (RMSE): 0.18455804191270428
55 Final, out of sample score (RMSE): 0.17696627225086287
56
57 ***Question 5***
58 Fold #1
59 Step #50, epoch #3, avg. train loss: 0.53294, avg. val loss: 0.37356
60 ...Lines removed for space...
61 Step #500, epoch #38, avg. train loss: 0.09219, avg. val loss: 0.03402
62 Fold score: 0.9875
63 Fold #2
64 Step #50, epoch #3, avg. train loss: 0.53294, avg. val loss: 0.44299

```

65 ...Lines removed for space...
66 Step #500, epoch #38, avg. train loss: 0.09219, avg. val loss: 0.03660
67 Fold score: 1.0
68 Fold #3
69 Step #50, epoch #3, avg. train loss: 0.53294, avg. val loss: 0.44690
70 ...Lines removed for space...
71 Step #500, epoch #38, avg. train loss: 0.09219, avg. val loss: 0.06254
72 Fold score: 0.9625
73 Fold #4
74 Step #50, epoch #3, avg. train loss: 0.53294, avg. val loss: 0.64565
75 ...Lines removed for space...
76 Step #500, epoch #38, avg. train loss: 0.09219, avg. val loss: 0.18395
77 Fold score: 0.9367088607594937
78 Fold #5
79 Step #50, epoch #3, avg. train loss: 0.53294, avg. val loss: 0.50485
80 ...Lines removed for space...
81 Step #500, epoch #38, avg. train loss: 0.09219, avg. val loss: 0.11250
82 Fold score: 0.9620253164556962
83 Final, out of sample score: 0.9698492462311558

```

Question 1

Encode the **toy1.csv** dataset. Generate dummy variables for the shape and metal. Encode high, width and length as z-scores. Include, but do not encode the weight. If this encoding is performed in a function, named **encode_toy_dataset**, you will have an easier time reusing the code from question 1 in question 2.

Write the output to a CSV file that you will submit with this assignment. The CSV file will look similar to Listing 3.

Listing 3: Question 2 Output Sample

```

1 height,length,width,weight,metal-bronze,metal-gold,metal-platinum,metal-↵
  silver,metal-tin,shape-box,shape-cylinder,shape-sphere
2 -0.18585564084337075, -0.18381430131795315, -0.1866234261937586, ↵
  729.63,1.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0
3 -0.5340303145851667, -0.18381430131795315, 0.16305509393694917, ↵
  2530.8,0.0,0.0,1.0,0.0,0.0,1.0,0.0,0.0
4 -1.2303796620687586, -1.2323841890415879, -1.5853375067165896, ↵
  58.37,1.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0
5 -0.8822049883269626, -0.8828608931337095, -0.8859804664551741, ↵
  74.15,0.0,0.0,0.0,1.0,0.0,0.0,0.0,1.0
6 ...

```

Question 2

Use the encoded dataset from question 1 and train a neural network. Use 25% of the data as validation and 75% as training, make sure you shuffle the data. Report the RMSE error for the validation set. No CSV file need be generated for this question.

Question 3

Using the **toy1.csv** dataset calculate and report the mean and standard deviation for height, width and length. Calculate the z-scores for the dataframe given by Listing 4. Make sure that you use the mean and standard deviations you reported for this question. Write the results to a CSV file.

Listing 4: Question 3 Input Data

```
1 testDF = pd.DataFrame([
2     {'length':1, 'width':2, 'height': 3},
3     {'length':3, 'width':2, 'height': 5},
4     {'length':4, 'width':1, 'height': 3}
5 ])
6 ...
```

Your resulting CSV file should look almost exactly like Listing 5.

Listing 5: Question 3 Output Sample

```
1 height,length,width
2 -0.8822049883269626,-1.5819074849494659,-1.2356589865858818
3 -0.18585564084337075,-0.8828608931337095,-1.2356589865858818
4 -0.8822049883269626,-0.5333375972258314,-1.5853375067165896
```

Question 4

Usually the **iris.csv** dataset is used to classify the species. Not this time! Use the fields species, sepal-l, sepal-w, and petal-l to predict petal-w. Use a 5-fold crossvalidation and report ONLY out-of-sample predictions to a CSV file. Your generated CSV file should look similar to Listing 6. Encode each of the inputs in a way that makes sense (e.g. dummies, z-scores).

Listing 6: Question 4 Output Sample

```
1 sepal_l,sepal_w,petal_l,petal_w,species-Iris-setosa,species-Iris-↵
   versicolor,species-Iris-virginica,0,0
2 0.30995914214417364, -0.5903951331558184, 0.5336208818725668, ↵
   1.2,0.0,1.0,0.0,1.2,1.444551944732666
```

```
3 -0.1730940663922016, 1.7038864723719687, -1.1658086782311483, ↵  
    0.3,1.0,0.0,0.0,0.3,0.\  
4 ...
```

Question 5

Usually the **auto-mpg.csv** dataset is used to regress the mpg. Not this time! Use the fields to predict how many cylinders the car has. Treat this as a classification problem, where there is a class for each number of cylinders. Use a 5-fold crossvalidation and report ONLY out-of-sample predictions to a CSV file. Your generated CSV file should look similar to Listing 7. Encode each of the inputs in a way that makes sense (e.g. dummies, z-scores). Report the final out of sample accuracy score.

Listing 7: Question 4 Output Sample

```
1 mpg,cylinders,displacement,horsepower,weight,acceleration,year,origin,name↵  
    ,ideal,predict  
2 -0.7055506566787514, 8, 1.0892327311042995, 0.6722714619460141, ↵  
    0.6300768256149949, -1.2938698102195594, 70, -0.7142457922976494,↵  
    chevrolet chevelle malibu,8,8  
3 -1.0893794720944747, 8, 1.5016242793620063, 1.5879594901955474, ↵  
    0.8532590135498572, -1.4751810504376373, 70, -0.7142457922976494,buick↵  
    skylark 320,8,8  
4 -0.7055506566787514, 8, 1.1947282434492943, 1.19552176380289, ↵  
    0.5497784722839334, -1.6564922906557151, 70, -0.7142457922976494,↵  
    plymouth satellite,8,8  
5 ...
```
