IE 345 - K “Introduction to Deep Learning: Fundamentals Concepts”

Prof. Yuzo

Case Studies with Real Data

pg. 106 - 110

In [1]:

```
#import all relevant libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
import tflearn
```

In [2]:

```
#load the dataset
dataset = pd.read_csv('./datasets/Churn_Modelling.csv')
```

In [3]:

```
#get first five rows (observations)
print(dataset.shape)
dataset.head(10)
```

```
(10000, 14)
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Bal
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	838(
2	3	15619304	Onio	502	France	Female	42	8	15966
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12551
5	6	15574012	Chu	645	Spain	Male	44	8	11375
6	7	15592531	Bartlett	822	France	Male	50	7	
7	8	15656148	Obinna	376	Germany	Female	29	4	11504
8	9	15792365	He	501	France	Male	44	4	14205
9	10	15592389	H?	684	France	Male	27	2	13460

In [4]:

```
X = dataset.iloc[:, 3:13].values
y = dataset.iloc[:, 13].values
print(X.shape)
print(y.shape)
X[:3,:]
```

```
(10000, 10)
(10000,)
```

Out[4]:

```
array([[619, 'France', 'Female', 42, 2, 0.0, 1, 1, 1, 101348.88],
       [608, 'Spain', 'Female', 41, 1, 83807.86, 1, 0, 1, 112542.58],
       [502, 'France', 'Female', 42, 8, 159660.8, 3, 1, 0, 113931.57]],
      dtype=object)
```

In [5]:

```
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

ct = ColumnTransformer(
    [('one_hot_encoder', OneHotEncoder(), [1, 2])],    # The column numbers to be trans
formed (here is [1, 2]
    remainder='passthrough'                         # Leave the rest of the columns unt
ouched
    )

X = np.array(ct.fit_transform(X), dtype=np.float)
X[:3,:].shape
```

Out[5]:

```
(3, 13)
```

In [6]:

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state
= 0) # 20% for testing dataset
```

In [7]:

```
y_train = np.reshape(y_train, (-1, 1)) #reshape y_train to [None, 1]
y_test = np.reshape(y_test, (-1, 1)) #reshape y_test to [None, 1]
```

In [8]:

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_test.shape)
print(y_test.shape)
```

```
(2000, 13)
(2000, 1)
```

In [9]:

```
# Using TFLearn wrappers for network building
net = tflearn.input_data(shape=[None,13])
net = tflearn.fully_connected(net, 6, activation='relu')
net = tflearn.dropout(net, 0.5)
net = tflearn.fully_connected(net, 6, activation='relu')
net = tflearn.dropout(net, 0.5)
net = tflearn.fully_connected(net, 1, activation='tanh')
net = tflearn.regression(net)
```

In [10]:

```
#define model
model = tflearn.DNN(net)
#we start training by applying gradient descent algorithm
model.fit(X_train, y_train, n_epoch = 10, batch_size = 16,
          validation_set = (X_test, y_test), show_metric =True, run_id="dense_model")
```

```
Training Step: 4999  | total loss: nan | time: 1.014s
| Adam | epoch: 010 | loss: nan - binary_acc: 0.8033 -- iter: 7984/8000
Training Step: 5000  | total loss: nan | time: 2.051s
| Adam | epoch: 010 | loss: nan - binary_acc: 0.8104 | val_loss: nan - val
_acc: 0.7975 -- iter: 8000/8000
--
```

Alex Rodriguez

alexmidwarr@gmail.com