wine_neural_network

February 17, 2019

In [1]:
```
import tensorflow as tf
import pandas as pd
from sklearn.utils import shuffle
from sklearn import preprocessing
```

In [2]:
```
flags = tf.app.flags
FLAGS = flags.FLAGS

columns = ['Wine'] + ['col' + str(i) for i in range(1,14)]

df_train = pd.read_csv("train_wine.csv", names=columns)
df_test = pd.read_csv("test_wine.csv", names=columns)
```

In [3]:
```
df_train.shape, df_test.shape
```

Out[3]:
```
((151, 14), (28, 14))
```

In [4]:
```
X_train = df_train[df_train.columns[1:14]].values
X_test = df_test[df_test.columns[1:14]].values
```

In [5]:
```
y_train = df_train['Wine'].values-1
y_test = df_test['Wine'].values-1
```

In [6]:
```
sess = tf.InteractiveSession()
```

In [7]:
```
Y_train = tf.one_hot(indices = y_train, depth=3, on_value=1., off_value=0., axis=1 , nam
Y_test = tf.one_hot(indices = y_test, depth=3, on_value=1., off_value=0., axis=1 , name
```

In [8]:
```
X_train, Y_train = shuffle (X_train, Y_train)
X_test, Y_test = shuffle (X_test, Y_test)

scaler = preprocessing.StandardScaler()

sc = scaler.fit(X_train)

X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

In [9]:
```
# Create the model
x = tf.placeholder(tf.float32, [None, 13])
W = tf.Variable(tf.zeros([13, 3]))
b = tf.Variable(tf.zeros([3]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

```
WARNING:tensorflow:From /home/aniruddha-tapas/miniconda3/lib/python3.7/site-packages/tensorflow/
Instructions for updating:
Colocations handled automatically by placer.
```

In [10]:
```
# Define loss and optimizer
y_ = tf.placeholder(tf.float32, [None, 3])
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
train_step = tf.train.GradientDescentOptimizer(0.1).minimize(cross_entropy)
```

```
WARNING:tensorflow:From /home/aniruddha-tapas/miniconda3/lib/python3.7/site-packages/tensorflow/
Instructions for updating:
Use tf.cast instead.
```

In [11]:
```
# Train
tf.initialize_all_variables().run()

for i in range(100):
    X_train,Y_train =shuffle (X_train, Y_train, random_state=1)

    #batch_xs, batch_ys = mnist.train.next_batch(100)
    batch_xs, batch_ys = X_train , Y_train

    train_step.run({x: batch_xs, y_: batch_ys})

    cost = sess.run (cross_entropy,
                     feed_dict={x: batch_xs, y_: batch_ys})
    # Test trained model
    correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    print("Cost:{0} | Accuracy:{1}".format(cost, accuracy.eval({x: X_test, y_: Y_test})
```

```
WARNING:tensorflow:From /home/aniruddha-tapas/miniconda3/lib/python3.7/site-packages/tensorflow/
Instructions for updating:
Use `tf.global_variables_initializer` instead.
Cost:0.9262384176254272 | Accuracy:0.9285714030265808
Cost:0.7961614727973938 | Accuracy:0.9642857313156128
Cost:0.6973766088485718 | Accuracy:0.9642857313156128
Cost:0.6210889220237732 | Accuracy:0.9642857313156128
Cost:0.560943603515625 | Accuracy:0.9642857313156128
Cost:0.5125309824943542 | Accuracy:0.9642857313156128
Cost:0.47281116247177124 | Accuracy:0.9642857313156128
```

Cost:0.4396665096282959 | Accuracy:0.9642857313156128
Cost:0.4115954339504242 | Accuracy:0.9642857313156128
Cost:0.3875121772289276 | Accuracy:0.9642857313156128
Cost:0.3666156530380249 | Accuracy:0.9642857313156128
Cost:0.34830382466316223 | Accuracy:0.9642857313156128
Cost:0.3321162462234497 | Accuracy:0.9642857313156128
Cost:0.31769537925720215 | Accuracy:0.9642857313156128
Cost:0.30475953221321106 | Accuracy:0.9642857313156128
Cost:0.2930837869644165 | Accuracy:0.9642857313156128
Cost:0.2824867069721222 | Accuracy:0.9642857313156128
Cost:0.2728201150894165 | Accuracy:0.9642857313156128
Cost:0.263962060213089 | Accuracy:0.9642857313156128
Cost:0.25581103563308716 | Accuracy:0.9642857313156128
Cost:0.24828200042247772 | Accuracy:0.9642857313156128
Cost:0.24130327999591827 | Accuracy:0.9642857313156128
Cost:0.23481371998786926 | Accuracy:0.9642857313156128
Cost:0.22876113653182983 | Accuracy:0.9642857313156128
Cost:0.22310061752796173 | Accuracy:0.9642857313156128
Cost:0.217793270945549 | Accuracy:0.9642857313156128
Cost:0.21280525624752045 | Accuracy:0.9642857313156128
Cost:0.20810696482658386 | Accuracy:0.9642857313156128
Cost:0.20367248356342316 | Accuracy:0.9642857313156128
Cost:0.1994788497686386 | Accuracy:0.9642857313156128
Cost:0.19550581276416779 | Accuracy:0.9642857313156128
Cost:0.1917353868484497 | Accuracy:0.9642857313156128
Cost:0.18815144896507263 | Accuracy:0.9642857313156128
Cost:0.18473967909812927 | Accuracy:0.9642857313156128
Cost:0.1814870983362198 | Accuracy:0.9642857313156128
Cost:0.17838217318058014 | Accuracy:0.9642857313156128
Cost:0.17541439831256866 | Accuracy:0.9642857313156128
Cost:0.1725742369890213 | Accuracy:0.9642857313156128
Cost:0.1698531210422516 | Accuracy:0.9642857313156128
Cost:0.16724321246147156 | Accuracy:0.9642857313156128
Cost:0.1647372841835022 | Accuracy:0.9642857313156128
Cost:0.16232891380786896 | Accuracy:0.9642857313156128
Cost:0.16001208126544952 | Accuracy:0.9642857313156128
Cost:0.1577812284231186 | Accuracy:0.9642857313156128
Cost:0.155631422996521 | Accuracy:0.9642857313156128
Cost:0.1535579413175583 | Accuracy:0.9642857313156128
Cost:0.15155649185180664 | Accuracy:0.9642857313156128
Cost:0.14962312579154968 | Accuracy:0.9642857313156128
Cost:0.14775417745113373 | Accuracy:0.9642857313156128
Cost:0.14594624936580658 | Accuracy:0.9642857313156128
Cost:0.14419615268707275 | Accuracy:0.9642857313156128
Cost:0.14250095188617706 | Accuracy:0.9642857313156128
Cost:0.14085790514945984 | Accuracy:0.9642857313156128
Cost:0.13926446437835693 | Accuracy:0.9642857313156128
Cost:0.13771824538707733 | Accuracy:0.9642857313156128

```
Cost:0.13621702790260315 | Accuracy:1.0
Cost:0.13475869596004486 | Accuracy:1.0
Cost:0.13334134221076965 | Accuracy:1.0
Cost:0.13196305930614471 | Accuracy:1.0
Cost:0.13062217831611633 | Accuracy:1.0
Cost:0.129317045211792 | Accuracy:1.0
Cost:0.1280461698770523 | Accuracy:1.0
Cost:0.1268080621957779 | Accuracy:1.0
Cost:0.12560144066810608 | Accuracy:1.0
Cost:0.12442495673894882 | Accuracy:1.0
Cost:0.12327741086483002 | Accuracy:1.0
Cost:0.12215770035982132 | Accuracy:1.0
Cost:0.12106470763683319 | Accuracy:1.0
Cost:0.11999742686748505 | Accuracy:1.0
Cost:0.1189548671245575 | Accuracy:1.0
Cost:0.11793611198663712 | Accuracy:1.0
Cost:0.11694031208753586 | Accuracy:1.0
Cost:0.11596661806106567 | Accuracy:1.0
Cost:0.1150142103433609 | Accuracy:1.0
Cost:0.11408239603042603 | Accuracy:1.0
Cost:0.11317042261362076 | Accuracy:1.0
Cost:0.11227761954069138 | Accuracy:1.0
Cost:0.11140332370996475 | Accuracy:1.0
Cost:0.11054694652557373 | Accuracy:1.0
Cost:0.10970786213874817 | Accuracy:1.0
Cost:0.10888554155826569 | Accuracy:1.0
Cost:0.10807940363883972 | Accuracy:1.0
Cost:0.10728895664215088 | Accuracy:1.0
Cost:0.10651372373104095 | Accuracy:1.0
Cost:0.10575319081544876 | Accuracy:1.0
Cost:0.10500692576169968 | Accuracy:1.0
Cost:0.10427451133728027 | Accuracy:1.0
Cost:0.10355552285909653 | Accuracy:1.0
Cost:0.10284954309463501 | Accuracy:1.0
Cost:0.10215621441602707 | Accuracy:1.0
Cost:0.10147515684366226 | Accuracy:1.0
Cost:0.10080602765083313 | Accuracy:1.0
Cost:0.10014849156141281 | Accuracy:1.0
Cost:0.09950221329927444 | Accuracy:1.0
Cost:0.09886687994003296 | Accuracy:1.0
Cost:0.09824220091104507 | Accuracy:1.0
Cost:0.09762787818908691 | Accuracy:1.0
Cost:0.0970236286520958 | Accuracy:1.0
Cost:0.09642919898033142 | Accuracy:1.0
Cost:0.0958443284034729 | Accuracy:1.0
```