Connect-4 Neural Network

June 25, 2019

In [50]:
```
from keras.models import Sequential
from keras.layers import Dense
import numpy
import time

from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

from sklearn.model_selection import train_test_split
```

In [51]:
```
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
```

In [52]:
```
# load connect 4 dataset
dataset = numpy.genfromtxt("connect-4.csv", dtype='str', delimiter=",")
```

In [53]:
```
# split into input (X) and output (Y) variables
preX = dataset[:,0:42]
preY = dataset[:,42]

X = numpy.zeros(preX.shape)
Y = numpy.zeros(preY.shape)
```

In [54]:
```
# converting predictors into numerical data
for i, row in enumerate(preX):
    for j, col in enumerate(row):
        if col == 'x':
            X[i,j] = 1.0
        if col == 'o':
            X[i,j] = -1.0
        if col == 'b':
            X[i,j] = 0.0
```

In [55]:
```
# converting categorical variable into numerical values
encoder = LabelEncoder()
```

```
        encoder.fit(preY)

        # code: 0 - draw; 1 - loss; 2 -win
        encoded_Y = encoder.transform(preY)
        dummy_y = np_utils.to_categorical(encoded_Y)
```

In [56]:
```
        # splitting the dataset into 80% training and 20% test data set
        train, test, label_train, label_test = train_test_split(X, dummy_y,test_size = 0.2)
```

In [57]:
```
        # training the neural network model
        start = time.time()

        model = Sequential()
        model.add(Dense(8, input_dim=42, activation='relu'))
        model.add(Dense(3, activation='softmax'))
        model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        model.fit(train, label_train, epochs=5, batch_size=5)

        end = time.time()
```

```
Epoch 1/5
54045/54045 [==============================] - 43s 797us/step - loss: 0.6293 - acc: 0.7459
Epoch 2/5
54045/54045 [==============================] - 44s 808us/step - loss: 0.5729 - acc: 0.7708
Epoch 3/5
54045/54045 [==============================] - 44s 808us/step - loss: 0.5640 - acc: 0.7742
Epoch 4/5
54045/54045 [==============================] - 35s 644us/step - loss: 0.5597 - acc: 0.7751
Epoch 5/5
54045/54045 [==============================] - 27s 498us/step - loss: 0.5576 - acc: 0.7764
```

In [58]:
```
        # predicting the test data
        pred = model.predict(test)
        pred = numpy.argmax(pred, axis=1)
        label_test = numpy.argmax(label_test, axis=1)
```

In [59]:
```
        # the function to calculate the metrics
        def metrics(cm, cl, size):
            cm = numpy.array(cm)
            tp = cm[cl][cl]
            fp = sum(cm[x, cl] for x in range(3))-cm[cl][cl]
            fn = sum(cm[cl, x] for x in range(3))-cm[cl][cl]
            tn = size - tp - fp - fn
            precision = tp/(tp+fp)
            recall = tp/(tp+fn)
            fmeasure = 2*(precision*recall)/(precision + recall)
            accuracy = (tp + tn)/size

            return precision, recall, fmeasure, accuracy
```

```
In [60]: # getting the confusion matrix
         cm = confusion_matrix(label_test, pred)
         print("Confusion matrix: ")
         print(cm)

Confusion matrix:
[[  12  376  857]
 [   8 2329 1053]
 [   8  705 8164]]
```

```
In [67]: # metrics for class 0 (draw)
         precision0, recall0, f0, acc0 = metrics(cm, 0, len(test))
         print("                Precision Recall F-measure Accuracy")
         print("Class 0 (draw): ", round(precision0, 3), "  ", round(recall0, 3), \
               " ", round(f0, 3), "   ", round(acc0,3))

                Precision Recall F-measure Accuracy
Class 0 (draw):  0.429    0.01   0.019     0.908
```

```
In [68]: # metrics for class 1 (lose)
         precision1, recall1, f1, acc1 = metrics(cm, 1, len(test))
         print("                Precision Recall F-measure Accuracy")
         print("Class 1 (loss): ", round(precision1, 3), "  ", round(recall1, 3), \
               " ", round(f1, 3), "   ", round(acc1,3))

                Precision Recall F-measure Accuracy
Class 1 (loss):  0.683    0.687   0.685     0.841
```

```
In [69]: # metrics for class 2 (win)
         precision2, recall2, f2, acc2 = metrics(cm, 2, len(test))
         print("                Precision Recall F-measure Accuracy")
         print("Class 2 (win): ", round(precision2, 3), "  ", round(recall2, 3), \
               " ", round(f2, 3), "   ", round(acc2,3))

                Precision Recall F-measure Accuracy
Class 2 (win):  0.81    0.92   0.862     0.806
```

```
In [64]: # average metrics
         avg_p = (precision0 + precision1 + precision2)/3.0
         avg_r = (recall0 + recall1 + recall2) / 3.0
         avg_f = (f0 + f1 + f2) / 3.0
         avg_a = (acc0 + acc1 + acc2)/ 3.0
         print("        Precision Recall F-measure Accuracy")
         print("Average: ", round(avg_p, 3), "  ", round(avg_r, 3), \
               " ", round(avg_f, 3), "   ", round(avg_a,3))
```

```
Average:  0.64 0.54 0.52 0.85
```

```
In [65]: # training time
         print("Training time: ", round(end - start, 5), " sec")
```

```
Training time:  192.61302  sec
```

```
In [66]: # accuracy score - number of instances correctly classified
         print("Accuracy score: ", round(accuracy_score(label_test, pred), 5))
```

```
Accuracy score:  0.77746
```