MNIST Decision Tree

July 15, 2019

In [1]:
```
import time
import mnist

from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

import numpy as np
```

In [2]:
```
# getting data
train = mnist.train_images()
label_train = mnist.train_labels()
test = mnist.test_images()
label_test = mnist.test_labels()
```

In [3]:
```
# reshaping the data for the decision tree
nsamples, nx, ny = train.shape
train = train.reshape((nsamples,nx*ny))

nsamples, nx, ny = test.shape
test = test.reshape((nsamples,nx*ny))
```

In [4]:
```
from sklearn.tree._tree import TREE_LEAF

def prune_index(inner_tree, index, threshold):
    if inner_tree.value[index].min() < threshold:
        # turn node into a leaf by "unlinking" its children
        inner_tree.children_left[index] = TREE_LEAF
        inner_tree.children_right[index] = TREE_LEAF
    # if there are shildren, visit them as well
    if inner_tree.children_left[index] != TREE_LEAF:
        prune_index(inner_tree, inner_tree.children_left[index], threshold)
        prune_index(inner_tree, inner_tree.children_right[index], threshold)
```

In [5]:
```
# building the decision tree of depth 10
start = time.time()

dt = DecisionTreeClassifier(max_depth = 10, random_state = 1)
dt.fit(train, label_train)
prune_index(dt.tree_, 0, 1)

end = time.time()
```

In [6]:
```
# predicting
pred = dt.predict(test)
```

In [7]:
```
# function to calculate the metrics
def metrics(cm, cls, size):
    cm = np.array(cm)
    tp = cm[cls][cls]
    fp = sum(cm[x, cls] for x in range(10))-cm[cls][cls]
    fn = sum(cm[cls, x] for x in range(10))-cm[cls][cls]
    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 [8]:
```
# Rows: Actual
# Cols: Predicted
cm = confusion_matrix(label_test, pred)
print("Confusion Matrix:\n ")
print(cm)
```

```
Confusion Matrix:

[[ 839    1    3    4    1   55   18    4   24   31]
 [   1 1043   43   15    3    9    3    1   17    0]
 [  12   62  760    8   19   20   52   21   57   21]
 [  16   16   53  714    8   63   12   12   93   23]
 [   4    9    5    1  705   23   20   28   42  145]
 [  29    9    7   60   13  630   45   47   33   19]
 [  27   20    9    1   19   34  766    3   26   53]
 [   2   37   50    2    9   10    1  830   28   59]
 [   6   25   28   25    8   46   62    5  712   57]
 [   6   12    4   25   29   37    9   37   60  790]]
```

In [9]:
```
# Class 0
precision0, recall0, f0, acc0 = metrics(cm, 0, len(test))
print("        Precision Recall F-measure Accuracy")
```

```
        print("Class 0: ", round(precision0, 3), "  ", round(recall0, 3), \
              " ", round(f0, 3), "   ", round(acc0,3))
```

```
        Precision Recall F-measure Accuracy
Class 0:  0.891    0.856   0.873     0.976
```

```
In [10]: # Class 1
         precision1, recall1, f1, acc1 = metrics(cm, 1, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 1: ", round(precision1, 3), "  ", round(recall1, 3), \
               " ", round(f1, 3), "   ", round(acc1,3))
```

```
        Precision Recall F-measure Accuracy
Class 1:  0.845    0.919   0.881     0.972
```

```
In [11]: # Class 2
         precision2, recall2, f2, acc2 = metrics(cm, 2, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 2: ", round(precision2, 3), "  ", round(recall2, 3), \
               " ", round(f2, 3), "   ", round(acc2,3))
```

```
        Precision Recall F-measure Accuracy
Class 2:  0.79    0.736   0.762     0.953
```

```
In [12]: # Class 3
         precision3, recall3, f3, acc3 = metrics(cm, 3, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 3: ", round(precision3, 3), "  ", round(recall3, 3), \
               " ", round(f3, 3), "   ", round(acc3,3))
```

```
        Precision Recall F-measure Accuracy
Class 3:  0.835    0.707   0.766     0.956
```

```
In [13]: # Class 4
         precision4, recall4, f4, acc4 = metrics(cm, 4, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 4: ", round(precision4, 3), "  ", round(recall4, 3), \
               " ", round(f4, 3), "   ", round(acc4,3))
```

```
        Precision Recall F-measure Accuracy
Class 4:  0.866    0.718   0.785     0.961
```

```
In [14]: # Class 5
         precision5, recall5, f5, acc5 = metrics(cm, 5, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 5: ", round(precision5, 3), "  ", round(recall5, 3), \
               " ", round(f5, 3), "   ", round(acc5,3))
```

```
        Precision Recall F-measure Accuracy
Class 5:  0.68    0.706   0.693     0.944
```

```
In [15]: # Class 6
         precision6, recall6, f6, acc6 = metrics(cm, 6, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 6: ", round(precision6, 3), "  ", round(recall6, 3), \
               " ", round(f6, 3), "   ", round(acc6,3))
```

```
        Precision Recall F-measure Accuracy
Class 6:  0.775    0.8   0.787     0.959
```

```
In [16]: # Class 7
         precision7, recall7, f7, acc7 = metrics(cm, 7, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 7: ", round(precision0, 3), "  ", round(recall7, 3), \
               " ", round(f7, 3), "   ", round(acc7,3))
```

```
        Precision Recall F-measure Accuracy
Class 7:  0.891    0.807   0.823     0.964
```

```
In [17]: # Class 8
         precision8, recall8, f8, acc8 = metrics(cm, 8, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 8: ", round(precision8, 3), "  ", round(recall8, 3), \
               " ", round(f8, 3), "   ", round(acc8,3))
```

```
        Precision Recall F-measure Accuracy
Class 8:  0.652    0.731   0.689     0.936
```

```
In [18]: # Class 9
         precision9, recall9, f9, acc9 = metrics(cm, 9, len(test))
         print("        Precision Recall F-measure Accuracy")
         print("Class 9: ", round(precision9, 3), "  ", round(recall9, 3), \
               " ", round(f9, 3), "   ", round(acc9,3))
```

```
        Precision Recall F-measure Accuracy
Class 9:  0.659    0.783   0.716     0.937
```

```
In [19]: # number of instances classified correctly
         acc_score = accuracy_score(pred, label_test)
         print("Accuracy_score: ", round(acc_score, 5))
```

```
Accuracy_score:  0.7789
```

In [20]:
```
# training time
print("Training Time: %s secs" % round(end - start, 5))
```

```
Training Time: 10.52419 secs
```

In [21]:
```
# Decision tree visualization
#dot_data = StringIO()
#export_graphviz(dt, out_file=dot_data)
#graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
#Image(graph.create_png())
```