Pruning Multiple neurons at one play

March 28, 2017

Compute the performance of MAB methods of pruning Multiple neurons at one time
MAP for choosing multi arms at one time

```
In [1]: import numpy as np
        import time
        import sys
        from numpy import *
        import matplotlib.pyplot as plt
        from sklearn import metrics
        %matplotlib inline
        #plt.rcParams['figure.figsize'] = (15, 6)
```

1 Load Bokeh

```
In [2]: from bokeh.layouts import row, gridplot
        from bokeh.plotting import figure, output_notebook, show
        from bokeh.models import Legend
        TOOLS = 'box_zoom,box_select,crosshair,resize,reset,lasso_select,pan,save,poly_select,ta
        output_notebook()
```

2 Load the data

```
In [3]: X_train = np.load('./cancer/X_train.npy')
        y_train = np.load('./cancer/y_train.npy')
        X_test = np.load('./cancer/X_test.npy')
        y_test = np.load('./cancer/y_test.npy')
        X_deploy = np.load('./cancer/X_deploy.npy')
        y_deploy = np.load('./cancer/y_deploy.npy')
        print('Number of training examples',len(X_train))
        print('Number of validation examples',len(X_test))
        print('Number of testing examples',len(X_deploy))

Number of training examples 364
Number of validation examples 91
Number of testing examples 114
```

```
In [4]: exec(open("core.py").read())   # pyhton 3x
```

2.1 Run Thompson Sampling pruning Algorithm

```
In [5]: algo = Thompson_Sampling([], [])
        Alg_name = 'Thompson_Sampling Algorithm'
        path = './Thompson_Sampling/'
        sys.path.append("./Thompson_Sampling")
        exec(open("mnist_cnnFORTESTING.py").read())
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/sklearn/cross_vali
  "This module will be removed in 0.20.", DeprecationWarning)
Using Theano backend.


Test fraction correct (NN-Score) = 0.13
Test fraction correct (NN-Accuracy) = 0.96
The time for running this method is 0.06955909729003906 seconds
Finsh playing start pruining:
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.97
Test after pruning= 0.97
Test after pruning= 0.97
Test after pruning= 0.97
Test after pruning= 0.97
Test after pruning= 0.97
Test after pruning= 0.97
Test after pruning= 0.97
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.93
Test after pruning= 0.91
Test after pruning= 0.83
Test after pruning= 0.83
Test after pruning= 0.69
Test after pruning= 0.68
Test after pruning= 0.67
Test after pruning= 0.67
Test after pruning= 0.67
Test after pruning= 0.67
Test after pruning= 0.67
Test after pruning= 0.67
Test after pruning= 0.67
```

Thompson_Sampling Algorithm
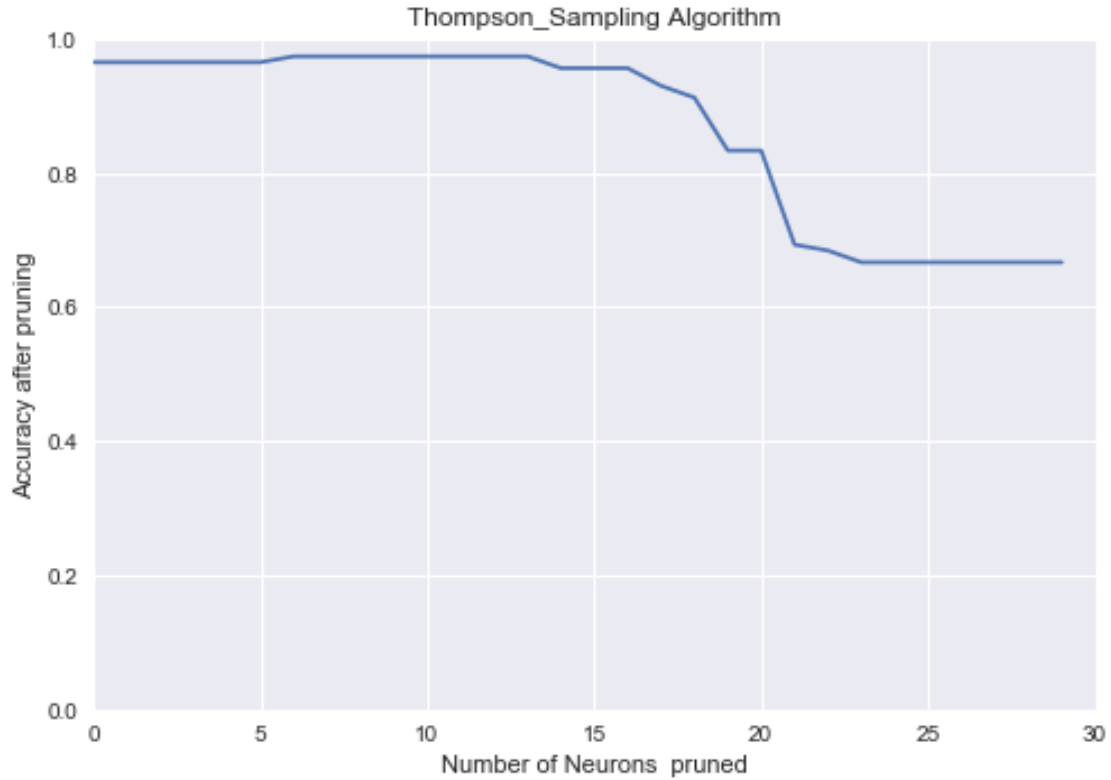
2.2 Run UCB1 pruning Algorithm

```
In [6]: algo = UCB1([], [])
        Alg_name = 'UCB1 Algorithm'
        path = './UCB1/'
        sys.path.append("./UCB1")
        exec(open("mnist_cnnFORTESTING.py").read())
```

```
Test fraction correct (NN-Score) = 0.13
Test fraction correct (NN-Accuracy) = 0.96
The time for running this method is 0.06415200233459473 seconds
Finsh playing start pruining:
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.96
Test after pruning= 0.97
Test after pruning= 0.97
```

3

```
Test after pruning= 0.97
Test after pruning= 0.97
Test after pruning= 0.97
Test after pruning= 0.97
Test after pruning= 0.96
Test after pruning= 0.92
Test after pruning= 0.88
Test after pruning= 0.88
Test after pruning= 0.84
Test after pruning= 0.82
Test after pruning= 0.82
Test after pruning= 0.70
Test after pruning= 0.70
Test after pruning= 0.70
Test after pruning= 0.67
Test after pruning= 0.67
Test after pruning= 0.67
Test after pruning= 0.67
Test after pruning= 0.67
Test after pruning= 0.67
```
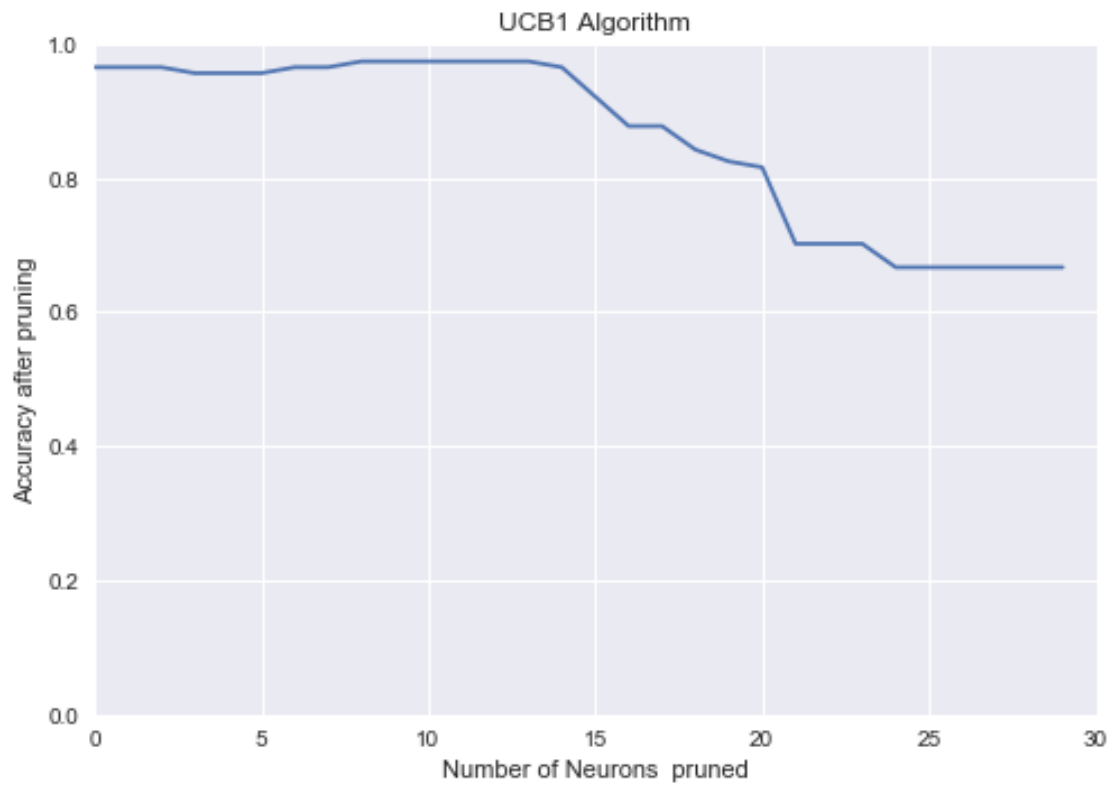
3   Compare the accuracy

```
In [7]: ucb1 = np.load('./UCB1/AccuracyAftrerPrune.npy')
        ThompsonSampling = np.load('./Thompson_Sampling/AccuracyAftrerPrune.npy')
        Accuracy = np.load('AccuracyBeforePruning.npy')
```
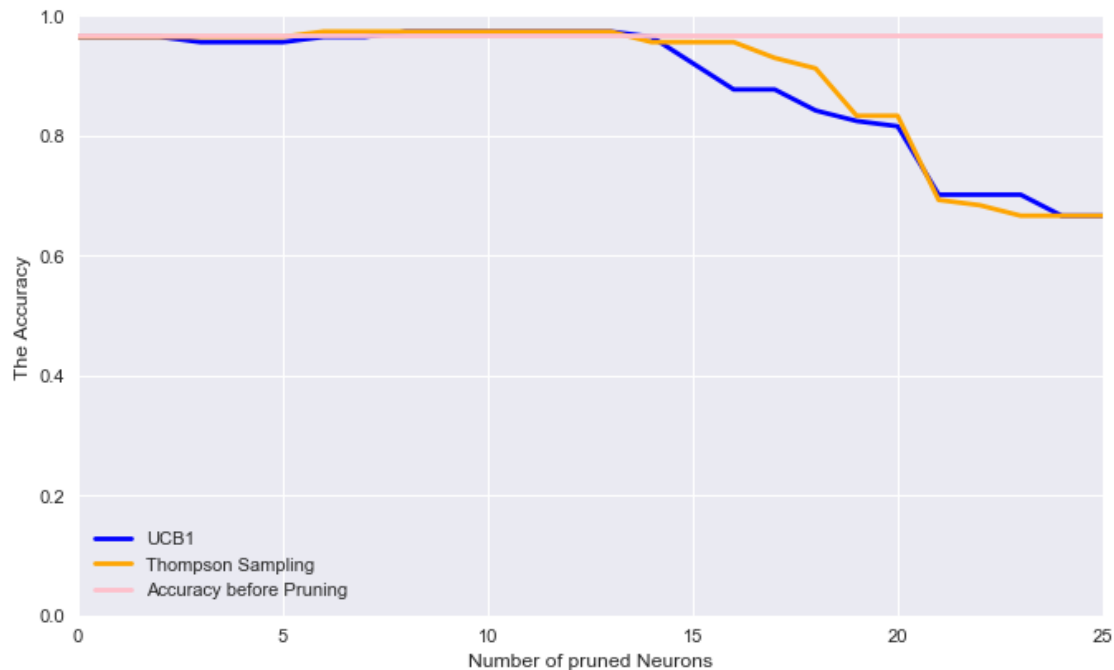
```
In [8]: fig = plt.figure(figsize=(10, 6), dpi=80)
        ax = fig.add_subplot(111)
        N = len(ucb1)
        ind = np.arange(N)                       # the x locations for the groups
        plt.plot(ind , ucb1 , color="blue", linewidth=2.5, linestyle="-", label="UCB1")
        plt.plot(ind , ThompsonSampling, color="orange", linewidth=2.5, linestyle="-", label="Th
        plt.legend(loc = 3)
        plt.axis([0, 25, 0, 1])
        plt.xlabel('Number of pruned Neurons')
        plt.ylabel('The Accuracy')
        plt.grid(True)
        plt.show()
```
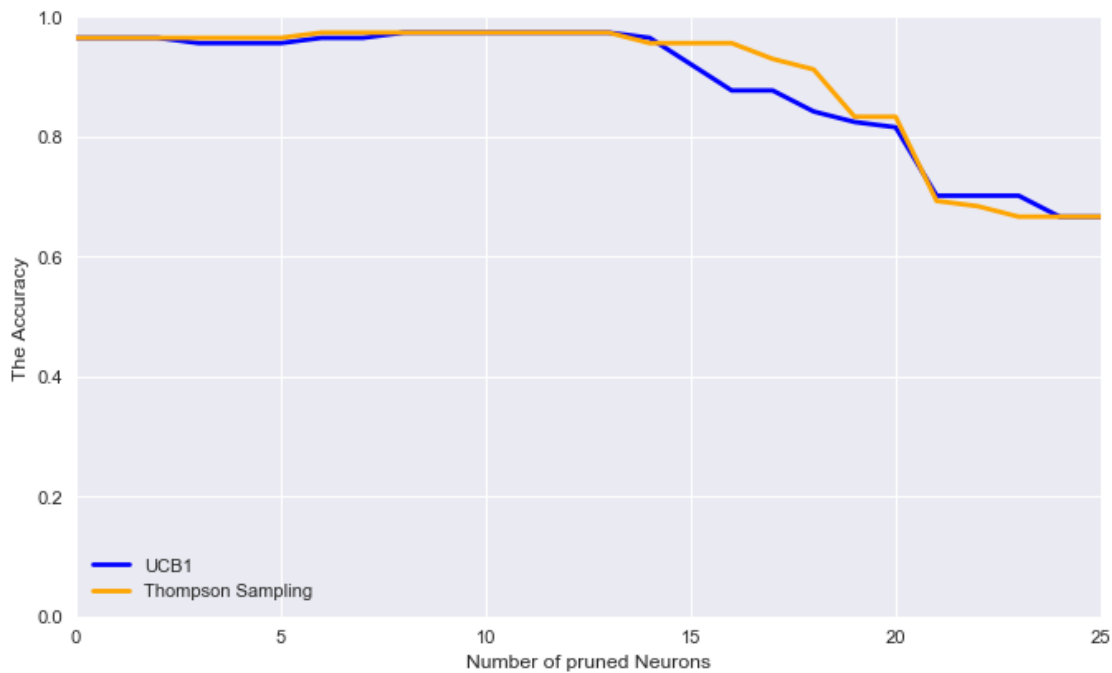


```
In [9]: p1 = figure(title="The Performance over the number of neurons' pruned", tools=TOOLS)
        p1.line(ind, ucb1, legend="ucb1", line_color="blue", line_width=2)
        p1.line(ind, ThompsonSampling, legend="Thompson Sampling", line_color="red", line_width=
        p1.title.align = "center"
        show(p1)
```

4 Comparing All algorithms with the model before pruning

```
In [10]: fig = plt.figure(figsize=(10, 6), dpi=80)
         ax = fig.add_subplot(111)
         N = len(ucb1)
         Acc = [Accuracy for col in range(N)]
         ind = np.arange(N)                    # the x locations for the groups
         plt.plot(ind , ucb1 , color="blue", linewidth=2.5, linestyle="-", label="UCB1")
         plt.plot(ind , ThompsonSampling, color="orange", linewidth=2.5, linestyle="-", label="T
         plt.plot(ind , Acc, color="pink", linewidth=2.5, linestyle="-", label="Accuracy before
         plt.legend(loc = 3)
         plt.axis([0, 25, 0, 1])
         plt.xlabel('Number of pruned Neurons')
         plt.ylabel('The Accuracy')
         plt.grid(True)
         plt.show()
```



```
In [11]: p1 = figure(title="The Performance over the number of neurons' pruned", tools=TOOLS)
         p1.line(ind, ucb1, legend="ucb1", line_color="green", line_width=2)
         p1.line(ind, ThompsonSampling, legend="Thompson Sampling", line_color="red", line_width
         p1.line(ind, Acc, legend="Accuracy", line_dash=(4, 4), line_color="blue", line_width=2)
         p1.title.align = "center"
         show(p1)
```