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 [7]: 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 [8]: 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 [9]: X_train = np.load('X_train.npy')
        y_train = np.load('y_train.npy')
        X_test = np.load('X_test.npy')
        y_test = np.load('y_test.npy')
        X_deploy = np.load('X_deploy.npy')
        y_deploy = np.load('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 2944
Number of validation examples 736
Number of testing examples 921
```

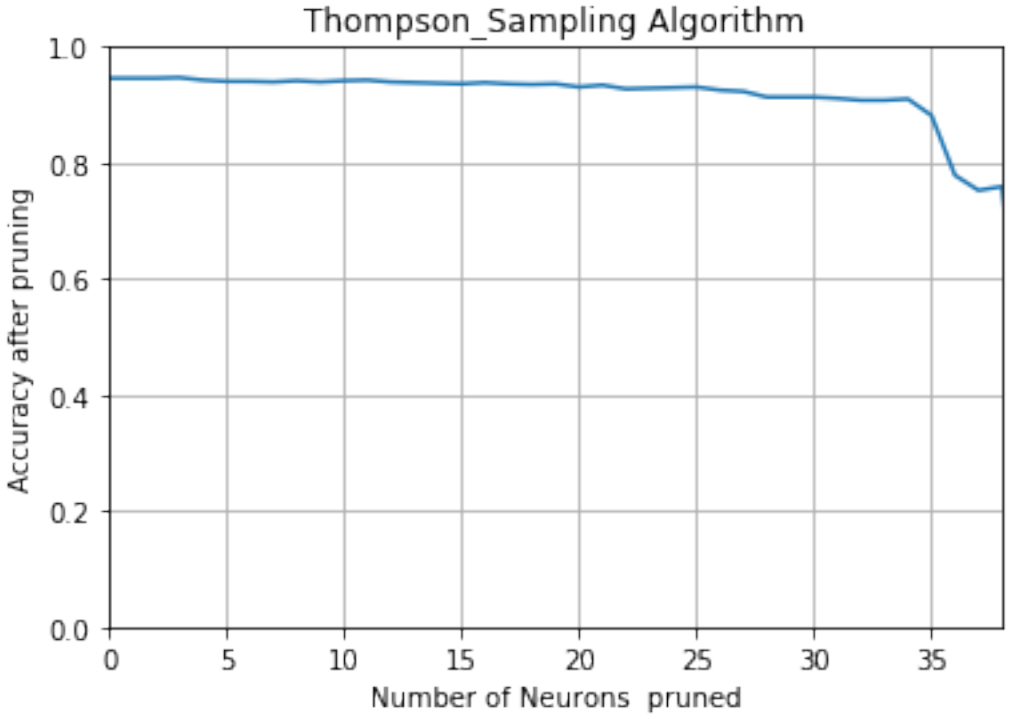
```
In [10]: exec(open("core.py").read())  # pyhton 3x
```

2.1 Run Thompson Sampling pruning Algorithm

```
In [11]: algo = Thompson_Sampling([], [])
         Alg_name = 'Thompson_Sampling Algorithm'
         path = './Thompson_Sampling/'
         sys.path.append("./Thompson_Sampling")
         exec(open("mnist_cnnFORTESTING.py").read())
```

```
Using Theano backend.


736 test samples
Test score: 0.338247084349
Test accuracy: 0.944625407166
The time for running this method is 1.5775988101959229 seconds
Finsh playing start pruining:
Test accuracy after pruning: 0.945711183496
Test accuracy after pruning: 0.945711183496
Test accuracy after pruning: 0.945711183496
Test accuracy after pruning: 0.946796959826
Test accuracy after pruning: 0.942453854506
Test accuracy after pruning: 0.940282301846
Test accuracy after pruning: 0.940282301846
Test accuracy after pruning: 0.939196524933
Test accuracy after pruning: 0.941368077593
Test accuracy after pruning: 0.939196524933
Test accuracy after pruning: 0.941368078176
Test accuracy after pruning: 0.942453854506
Test accuracy after pruning: 0.939196524933
Test accuracy after pruning: 0.938110748603
Test accuracy after pruning: 0.937024972273
Test accuracy after pruning: 0.935939196979
Test accuracy after pruning: 0.938110749639
Test accuracy after pruning: 0.935939195943
Test accuracy after pruning: 0.934853420648
Test accuracy after pruning: 0.935939196979
Test accuracy after pruning: 0.930510315328
Test accuracy after pruning: 0.933767644318
Test accuracy after pruning: 0.927252986338
Test accuracy after pruning: 0.928338762668
Test accuracy after pruning: 0.929424537963
Test accuracy after pruning: 0.930510315328
Test accuracy after pruning: 0.925081433095
Test accuracy after pruning: 0.922909880435
Test accuracy after pruning: 0.913137892882
Test accuracy after pruning: 0.913137892882
```

```
Test accuracy after pruning: 0.913137893464
Test accuracy after pruning: 0.910966340804
Test accuracy after pruning: 0.907709011814
Test accuracy after pruning: 0.907709011814
Test accuracy after pruning: 0.909880564474
Test accuracy after pruning: 0.88165037931
Test accuracy after pruning: 0.778501628859
Test accuracy after pruning: 0.752442996937
Test accuracy after pruning: 0.7589576555
Test accuracy after pruning: 0.375678610061
```

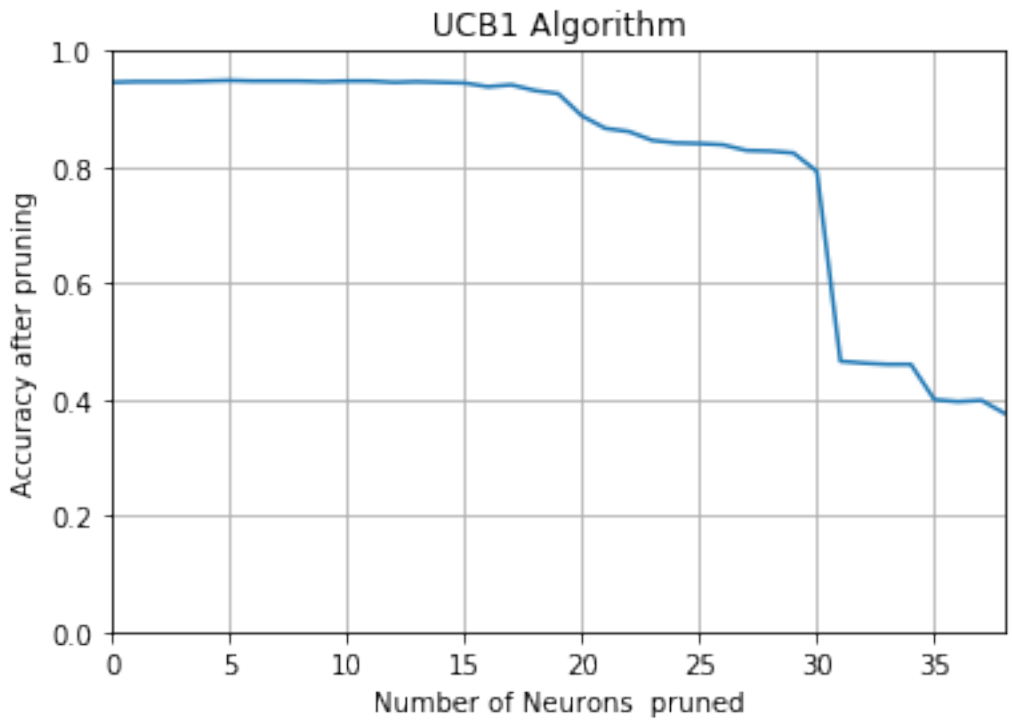


2.2 Run UCB1 pruning Algorithm

```
In [12]: algo = UCB1([], [])
         Alg_name = 'UCB1 Algorithm'
         path = './UCB1/'
         sys.path.append("./UCB1")
         exec(open("mnist_cnnFORTESTING.py").read())
```

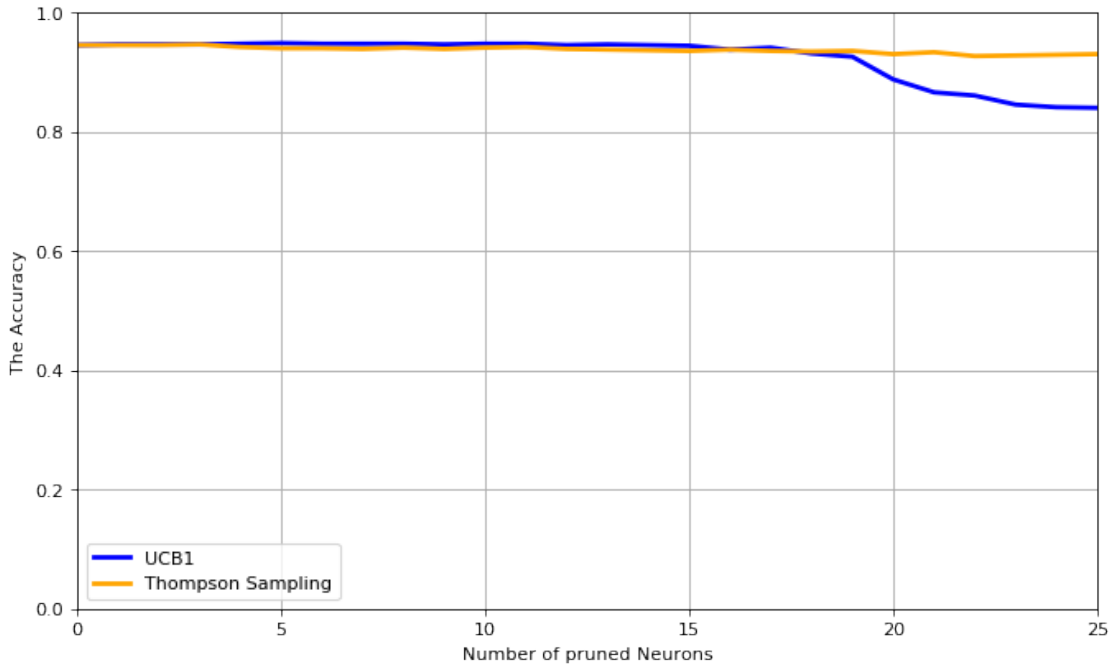
```
736 test samples
Test score: 0.338247084349
Test accuracy: 0.944625407166
The time for running this method is 1.5783360004425049 seconds
```

```
Finsh playing start pruining:
Test accuracy after pruning: 0.945711183496
Test accuracy after pruning: 0.946796959826
Test accuracy after pruning: 0.946796959826
Test accuracy after pruning: 0.946796960279
Test accuracy after pruning: 0.947882736609
Test accuracy after pruning: 0.948968511904
Test accuracy after pruning: 0.947882736609
Test accuracy after pruning: 0.947882736609
Test accuracy after pruning: 0.947882736609
Test accuracy after pruning: 0.946796960279
Test accuracy after pruning: 0.947882736609
Test accuracy after pruning: 0.947882736609
Test accuracy after pruning: 0.945711183949
Test accuracy after pruning: 0.946796960279
Test accuracy after pruning: 0.945711183949
Test accuracy after pruning: 0.944625407037
Test accuracy after pruning: 0.938110748474
Test accuracy after pruning: 0.941368077464
Test accuracy after pruning: 0.931596090493
Test accuracy after pruning: 0.926167208843
Test accuracy after pruning: 0.888165038326
Test accuracy after pruning: 0.866449511142
Test accuracy after pruning: 0.861020629491
Test accuracy after pruning: 0.845819761906
Test accuracy after pruning: 0.841476656586
Test accuracy after pruning: 0.840390880255
Test accuracy after pruning: 0.838219327595
Test accuracy after pruning: 0.828447340625
Test accuracy after pruning: 0.827361564295
Test accuracy after pruning: 0.824104235304
Test accuracy after pruning: 0.792616721732
Test accuracy after pruning: 0.465798045408
Test accuracy after pruning: 0.462540716418
Test accuracy after pruning: 0.460369163985
Test accuracy after pruning: 0.460369163985
Test accuracy after pruning: 0.399565689322
Test accuracy after pruning: 0.396308360332
Test accuracy after pruning: 0.398479912992
Test accuracy after pruning: 0.375678610061
Test accuracy after pruning: 0.375678610061
```



3 Compare the accuracy

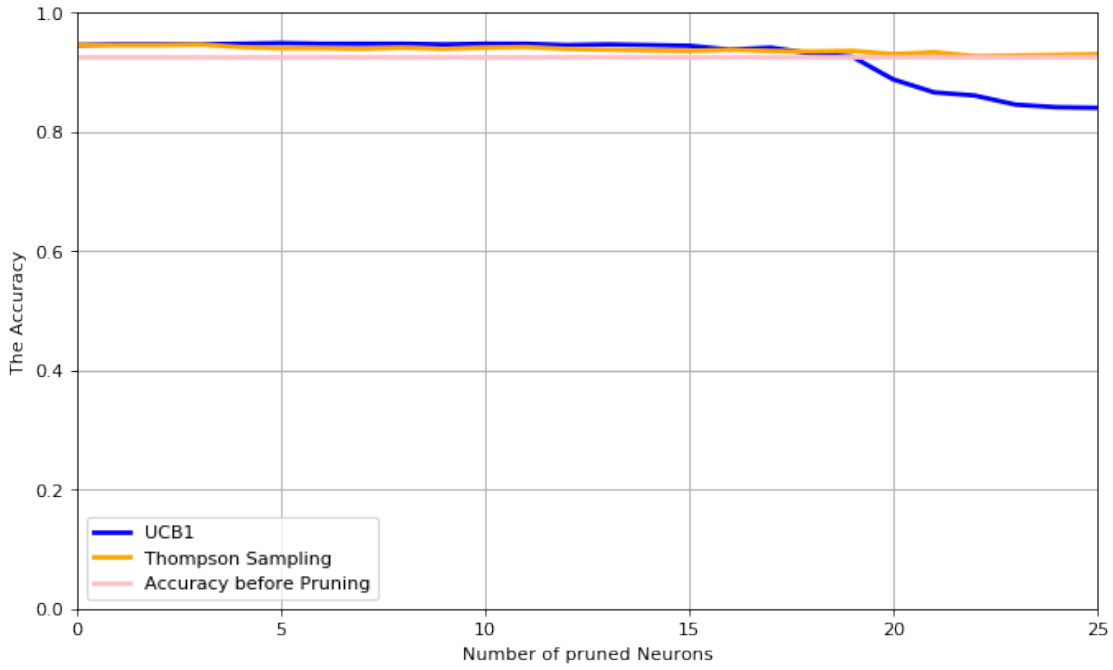
```
In [13]: ucb1 = np.load('./UCB1/AccuracyAftrerPrune.npy')
         ThompsonSampling = np.load('./Thompson_Sampling/AccuracyAftrerPrune.npy')
         Accuracy = np.load('AccuracyBeforePruning.npy')

In [14]: 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="T
         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 [15]: 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 [16]: 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 [17]:
```
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)
```