Compression Normal

March 28, 2017

Compute the performance of MAB methods

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

0.1 Load BOKEH libariry

```
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()
```

1 Compare the accuracy of the models

1.1 Load the pruned algorithm from normal prune

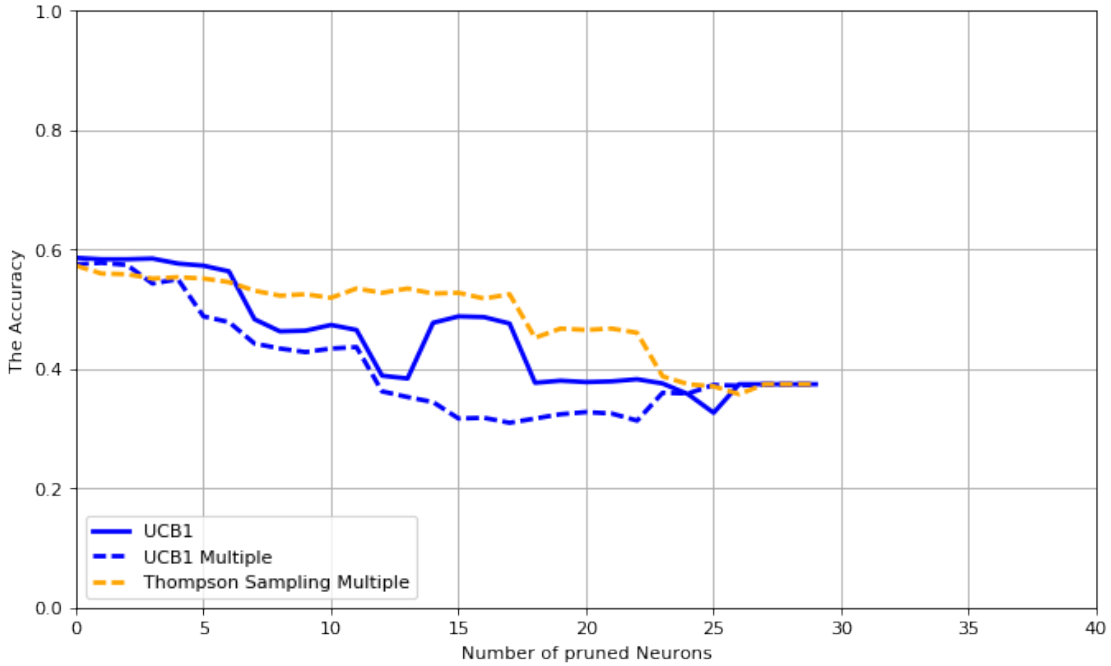
```
In [3]: ucb1 = np.load('/Users/salemameen/Desktop/banditsbook/python_abalone/UCB1/AccuracyAftrer
        Accuracy = np.load('/Users/salemameen/Desktop/banditsbook/python_abalone/AccuracyBeforeP
```

1.2 Load the pruned algorithm from Multiple prune

```
In [4]: ucb1_Multiple = np.load('/Users/salemameen/Desktop/banditsbook_Prune_many_one_play/pytho
        ThompsonSampling_Multiple = np.load('/Users/salemameen/Desktop/banditsbook_Prune_many_on
```

```
In [5]: fig = plt.figure(figsize=(10, 6), dpi=80)
        ax = fig.add_subplot(111)
        N = len(ucb1)
        ## necessary variables
        ind = np.arange(N)                # the x locations for the groups
        ### Normal algoritms
        plt.plot(ind , ucb1 , color="blue", linewidth=2.5, linestyle="-", label="UCB1")
        ### Absuluate once algoritms
```

```
plt.plot(ind , ucb1_Multiple , color="blue", linewidth=2.5, linestyle="--", label="UCB1
plt.plot(ind , ThompsonSampling_Multiple, color="orange", linewidth=2.5, linestyle="--",
##########################
plt.legend(loc = 3)
plt.axis([0, 40, 0, 1])
plt.xlabel('Number of pruned Neurons')
plt.ylabel('The Accuracy')
plt.grid(True)
plt.show()
```

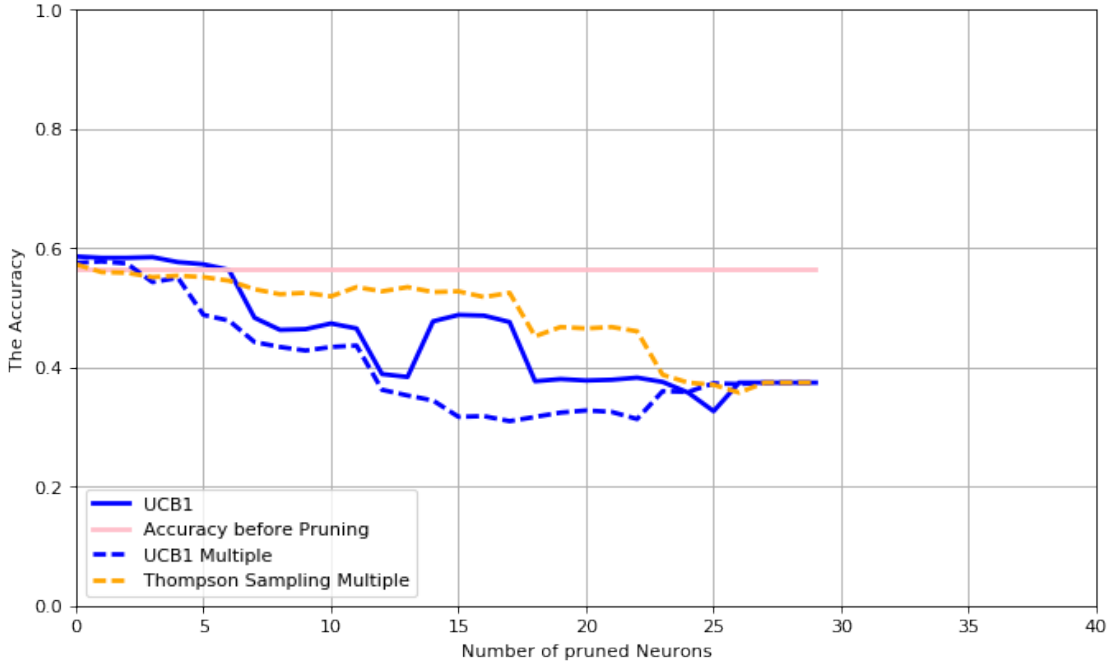


In [6]:
```
p1 = figure(title="The Performance over the number of neurons' pruned", tools=TOOLS)
p1.line(ind, ucb1, legend="ucb1", line_color="orange", line_width=2)
##### ================================================================================
# Multiple pruned
p1.circle(ind, ucb1_Multiple, legend="ucb1 Multiple", line_color="orange", line_width=2)
p1.line(ind, ucb1_Multiple, legend="ucb1 Multiple", line_color="orange", line_width=2)
p1.circle(ind, ThompsonSampling_Multiple, legend="Thompson Sampling Multiple", line_colo
p1.line(ind, ThompsonSampling_Multiple, legend="Thompson Sampling Multiple", line_color=
p1.title.align = "center"
show(p1)
```

1.3 Comparing All algorithms with the model before pruning

In [7]:
```
fig = plt.figure(figsize=(10, 6), dpi=80)
ax = fig.add_subplot(111)
```

```
N = len(ucb1)
Acc = [Accuracy for col in range(N)]
## necessary variables
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 , Acc, color="pink", linewidth=2.5, linestyle="-", label="Accuracy before P
### Absuluate once algoritms
plt.plot(ind , ucb1_Multiple , color="blue", linewidth=2.5, linestyle="--", label="UCB1
plt.plot(ind , ThompsonSampling_Multiple, color="orange", linewidth=2.5, linestyle="--",
plt.legend(loc = 3)
plt.axis([0, 40, 0, 1])
plt.xlabel('Number of pruned Neurons')
plt.ylabel('The Accuracy')
plt.grid(True)
plt.show()
```



In [8]:
```
p1 = figure(title="The Performance over the number of neurons' pruned", tools=TOOLS)
# Normal pruned
#p1.circle(ind, ucb1, legend="ucb1", color="orange")
p1.line(ind, ucb1, legend="ucb1", line_color="orange", line_width=2)

##### ================================================================================
# Multiple pruned
# Multiple pruned
p1.circle(ind, ucb1_Multiple, legend="ucb1 Multiple", line_color="red", line_width=2)
```

```
p1.line(ind, ucb1_Multiple, legend="ucb1 Multiple", line_color="red", line_width=2)

p1.circle(ind, ThompsonSampling_Multiple, legend="Thompson Sampling Multiple", line_colo
p1.line(ind, ThompsonSampling_Multiple, legend="Thompson Sampling Multiple", line_color=
p1.line(ind, Acc, legend="Accuracy", line_dash=(4, 4), line_color="orange", line_width=2
p1.title.align = "center"
show(p1)
```

In [ ]: