Accuracy Deep Learning-After update Alex Net

March 30, 2017

0.0.1 This report shows applyining statistical tests of the results of Multi armed bandit of pruning the parameters

0.0.2 Here, we are showing two kinds of testing ANOVA test and Nonparametric tests

1 Import needed libraries

1.1 Import libraries for manipulating the data and statistic

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import scipy.stats as stats
        from scipy.stats import ttest_1samp, wilcoxon, ttest_ind, mannwhitneyu
        import scipy.special as special
        import emoji
        from math import pi
        from statsmodels.stats.multicomp import pairwise_tukeyhsd, MultiComparison
        from statsmodels.formula.api import ols
        import statsmodels.stats.api as sms
```

1.2 Import libraries for static ploting

```
In [2]: import matplotlib.pyplot as plt
        import matplotlib.gridspec as gridspec
        %matplotlib inline
        from IPython.display import set_matplotlib_formats
        set_matplotlib_formats('png', 'pdf')
        # some nice colors from http://colorbrewer2.org/
        COLOR1 = '#7fc97f'
        COLOR2 = '#beaed4'
        COLOR3 = '#fdc086'
        COLOR4 = '#ffff99'
        COLOR5 = '#386cb0'
```

1.3 Import libraries for interactive ploting Plotly

```
In [3]: import plotly.plotly as py
        from plotly.graph_objs import *
```

```
import plotly.graph_objs as go
#from plotly.tools import FigureFactory as FF
import plotly.figure_factory as FF
import cufflinks as cf
cf.go_offline()
```

```
<IPython.core.display.HTML object>
```

1.4 Import libraries for interactive ploting BOKEH

In [4]:
```
from bokeh.charts import Bar, Area, defaults, Donut
from bokeh.layouts import row, gridplot
from bokeh.charts.attributes import cat, color
from bokeh.charts.operations import blend
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
#defaults.width = 1000
#defaults.height = 800
output_notebook()
```

2 Statring the test and visulize the data on small model

2.1 Load the data for pruning the weights using random expoloration

In [5]:
```
datafile = "./results/results.xlsx"
#datafileLeNet = "LecunPruningWeights.csv"
df_accuracy = pd.read_excel(datafile)

df_accuracy
```

Out[5]:

	Methods	MLP (Reuters)	LeCunn (MNIST)	AlexNet (FC6) (IMAGNET)	\
0	NN	0.786	0.98	0.56	
1	E Greedy	0.790	0.99	0.57	
2	Decay E Gr.	0.790	0.99	0.57	
3	Softmax	0.790	0.99	0.55	
4	Decay SM	0.790	0.99	0.55	
5	UCB1	0.790	0.99	0.58	
6	Tomp. Sampling	0.790	0.99	0.58	
7	Hedge	0.790	0.99	0.53	
8	EXP3	0.790	0.99	0.54	

	AlexNet (FC7) (IMAGNET)	Conv (Cifar10)	Conv (Cifar100)	Conv (IMDB)	\
0	0.56	0.81	0.43	0.8695	
1	0.58	0.82	0.45	0.8900	
2	0.58	0.82	0.45	0.8900	
3	0.57	0.82	0.44	0.8800	

```
4                    0.55            0.82             0.45       0.8900
5                    0.58            0.82             0.45       0.8900
6                    0.58            0.82             0.45       0.8900
7                    0.56            0.82             0.44       0.8900
8                    0.56            0.82             0.44       0.8800

   Conv (SVHN)  Siamese Graph (MNIST)  LSTM (IMDB)  Bi. LSTM_1 (IMDB)  \
0         0.96                  0.986         0.80               0.82
1         0.97                  1.000         0.81               0.84
2         0.97                  0.990         0.81               0.83
3         0.97                  0.990         0.81               0.83
4         0.97                  0.990         0.81               0.83
5         0.97                  1.000         0.81               0.84
6         0.97                  1.000         0.81               0.84
7         0.97                  0.990         0.81               0.83
8         0.97                  0.990         0.81               0.80

   Bi. LSTM_2 (IMDB)  EtoE Mem (bAbI)  Hierar. RNN (MNIST)
0               0.82          0.8589                 0.96
1               0.83          0.8700                 0.97
2               0.83          0.8800                 0.95
3               0.83          0.8700                 0.98
4               0.83          0.8700                 0.97
5               0.83          0.8800                 0.98
6               0.83          0.8700                 0.97
7               0.83          0.8700                 0.98
8               0.80          0.8800                 0.98
```

3 Starting with Accuracy

4 First, All methods

4.1 Visulize the Accuracy of all the models and methods

```
In [6]: p = Bar(df_accuracy, label= 'Methods',
                values = blend('MLP (Reuters)', 'AlexNet (FC6) (IMAGNET)', 'AlexNet (FC7) (IMAGN
                               'Conv (Cifar10)', 'Conv (Cifar100)', 'Conv (IMDB)', 'LeCunn (MNIS
                               'Conv (SVHN)', 'Siamese Graph (MNIST)', 'LSTM (IMDB)', 'Bi. LSTM_
                               'Bi. LSTM_2 (IMDB)', 'EtoE Mem (bAbI)', 'Hierar. RNN (MNIST)',
                                name='Scores', labels_name='Score'),
               group=cat(columns='Score', sort=False),
               title="Compare the performance", legend='bottom_center',
               tools=TOOLS, plot_width=3500, plot_height=1600,
               tooltips=[('Score', '@Score'), ('Model', '@Methods')],
               xlabel='List of Models', ylabel='Score')
        p.title.align = "center"
        #p.yaxis.major_label_orientation = "vertical"
```

```
        p.xaxis.major_label_orientation = pi/2
        show(p)
```

```
In [7]: df=df_accuracy.copy()
        df.set_index('Methods', inplace=True)
        py.iplot([{
            'x': df.index,
            'y': df[col],
            'name': col
        }  for col in df.columns])
```

```
Out[7]: <plotly.tools.PlotlyDisplay object>
```

```
In [8]: df.iplot(subplots=True, shape=(16,1), shared_xaxes=True, fill=True)
```

```
<IPython.core.display.HTML object>
```

```
In [9]: df.iplot(kind='bar', barmode='stack')
```

```
<IPython.core.display.HTML object>
```

```
In [10]: df.iplot(kind='barh',barmode='stack', bargap=.2)
```

```
<IPython.core.display.HTML object>
```

```
In [11]: df.T.iplot(kind='barh',barmode='stack', bargap=.2)
```

```
<IPython.core.display.HTML object>
```

```
In [12]: df.iplot(kind='box')
```

```
<IPython.core.display.HTML object>
```

```
In [13]: df.T.iplot(kind='box')
```

```
<IPython.core.display.HTML object>
```

4.1.1 We will use alpha 0.05 to do ANOVA test. The null hypothesis there is no difference between the all methods and the alternative hypothesis there is a difference. According to p-value we see if there is a difference.

```
In [14]: df.T.columns
```

```
Out[14]: Index(['NN', 'E Greedy', 'Decay E Gr.', 'Softmax', 'Decay SM', 'UCB1',
                'Tomp. Sampling', 'Hedge', 'EXP3'],
               dtype='object', name='Methods')
```

```
In [15]: # Perform the ANOVA
         df1 = df.T
         stats.f_oneway(df1['NN'], df1['UCB1'],df1['E Greedy'], df1['Decay E Gr.'],
                        df1['Softmax'], df1['Decay SM'], df1['Tomp. Sampling'],
                        df1['Hedge'], df1['EXP3'])
```

```
Out[15]: F_onewayResult(statistic=0.011251245202327209, pvalue=0.99999981810355187)
```

4.1.2 One post-hoc test is to perform a separate t-test for each pair of groups. We can perform a t-test between all pairs using by running each pair through the stats.ttest_ind() we covered in the following to do t-tests:

```
In [16]: # Get all models pairs
         interstModel = ['NN', 'UCB1',
                'E Greedy', 'Decay E Gr.', 'Softmax', 'Decay SM', 'Tomp. Sampling',
                'Hedge', 'EXP3']
         lst = list(df1.columns.values)
         #lst.remove('Methods')
         model_pairs = []

         for m1 in range(len(df1.columns)-1):
             for m2  in range(m1+1,len(df1.columns)):
                 model_pairs.append((lst[m1], lst[m2]))

         # Conduct t-test on each pair
         pvalueList = []
         new_model_pairs = []
         for m1, m2 in model_pairs:
             print('\n',m1, m2)
             pvalue = stats.ttest_ind(df1[m1], df1[m2])
             #print(pvalue[1])
             if (m1 in interstModel or m2 in interstModel):
                 new_model_pairs.append((m1,m2))
                 pvalueList.append(pvalue[1])
             print(pvalue)
```

```
 NN E Greedy
Ttest_indResult(statistic=-0.19968378613372589, pvalue=0.84328143046001824)

 NN Decay E Gr.
Ttest_indResult(statistic=-0.16731114092874064, pvalue=0.86841976491610007)

 NN Softmax
Ttest_indResult(statistic=-0.13126540176743365, pvalue=0.89657581869187519)

 NN Decay SM
Ttest_indResult(statistic=-0.12022265623396869, pvalue=0.90523098838694027)
```

NN UCB1
Ttest_indResult(statistic=-0.23309063080076725, pvalue=0.81751634102407456)

NN Tomp. Sampling
Ttest_indResult(statistic=-0.21147373203890799, pvalue=0.83416652721598683)

NN Hedge
Ttest_indResult(statistic=-0.10812103173466558, pvalue=0.9147297601430312)

NN EXP3
Ttest_indResult(statistic=-0.054076009804563251, pvalue=0.95728798422932648)

E Greedy Decay E Gr.
Ttest_indResult(statistic=0.033738677950430188, pvalue=0.97334322000595375)

E Greedy Softmax
Ttest_indResult(statistic=0.066205525585309552, pvalue=0.9477206633969284)

E Greedy Decay SM
Ttest_indResult(statistic=0.07719605079468396, pvalue=0.93905858331809999)

E Greedy UCB1
Ttest_indResult(statistic=-0.033546019682289444, pvalue=0.97349537899566618)

E Greedy Tomp. Sampling
Ttest_indResult(statistic=-0.011215497341757091, pvalue=0.99113713237775203)

E Greedy Hedge
Ttest_indResult(statistic=0.087299948232824248, pvalue=0.93110199360641799)

E Greedy EXP3
Ttest_indResult(statistic=0.14248175542110866, pvalue=0.88779790110777768)

Decay E Gr. Softmax
Ttest_indResult(statistic=0.033295253702898826, pvalue=0.97369343222727434)

Decay E Gr. Decay SM
Ttest_indResult(statistic=0.044368245372145766, pvalue=0.96494988152790473)

Decay E Gr. UCB1
Ttest_indResult(statistic=-0.067492788715877092, pvalue=0.94670575965335602)

Decay E Gr. Tomp. Sampling
Ttest_indResult(statistic=-0.045131577854210854, pvalue=0.96434728398523051)

Decay E Gr. Hedge
Ttest_indResult(statistic=0.054872722595799207, pvalue=0.95665935027007631)

Decay E Gr. EXP3
Ttest_indResult(statistic=0.11023006979384041, pvalue=0.91307337113402931)

Softmax Decay SM
Ttest_indResult(statistic=0.010888532168631967, pvalue=0.99139550062010784)

Softmax UCB1
Ttest_indResult(statistic=-0.099330152869861674, pvalue=0.92163804759337586)

Softmax Tomp. Sampling
Ttest_indResult(statistic=-0.077482172864622251, pvalue=0.93883317605466066)

Softmax Hedge
Ttest_indResult(statistic=0.021554604594747763, pvalue=0.98296781678679501)

Softmax EXP3
Ttest_indResult(statistic=0.075762313145837909, pvalue=0.94018816078827971)

Decay SM UCB1
Ttest_indResult(statistic=-0.11030432519630097, pvalue=0.91301505992213872)

Decay SM Tomp. Sampling
Ttest_indResult(statistic=-0.088500606927967476, pvalue=0.93015696996361497)

Decay SM Hedge
Ttest_indResult(statistic=0.010771481395783135, pvalue=0.99148799450428315)

Decay SM EXP3
Ttest_indResult(statistic=0.064903752392712052, pvalue=0.94874709848791583)

UCB1 Tomp. Sampling
Ttest_indResult(statistic=0.022436096892870443, pvalue=0.98227139140037867)

UCB1 Hedge
Ttest_indResult(statistic=0.12006327696938568, pvalue=0.90535599787413368)

UCB1 EXP3
Ttest_indResult(statistic=0.17540025251119062, pvalue=0.86212401777617176)

Tomp. Sampling Hedge
Ttest_indResult(statistic=0.098513855257048616, pvalue=0.92227985906414456)

Tomp. Sampling EXP3
Ttest_indResult(statistic=0.15391694278000945, pvalue=0.87886370431181882)

Hedge EXP3
Ttest_indResult(statistic=0.053541047183599443, pvalue=0.95771010405416179)

```
In [17]: for pair, p in zip(new_model_pairs, pvalueList):
             if p < 0.05:
                 print('The pvalue between',pair, 'is', p, '< 0.05 then',
                       emoji.emojize('REJECT the NULL Hypothesis :thumbs_up_sign:'))
             else:
                 print('The pvalue between',pair, 'is', p, '> 0.05 then',
                       emoji.emojize('FAIL to REJECT the NULL Hypothesis :thumbs_down_sign:'))
```

```
The pvalue between ('NN', 'E Greedy') is 0.84328143046 > 0.05 then FAIL to REJECT the NULL Hypot
The pvalue between ('NN', 'Decay E Gr.') is 0.868419764916 > 0.05 then FAIL to REJECT the NULL H
The pvalue between ('NN', 'Softmax') is 0.896575818692 > 0.05 then FAIL to REJECT the NULL Hypot
The pvalue between ('NN', 'Decay SM') is 0.905230988387 > 0.05 then FAIL to REJECT the NULL Hypo
The pvalue between ('NN', 'UCB1') is 0.817516341024 > 0.05 then FAIL to REJECT the NULL Hypothes
The pvalue between ('NN', 'Tomp. Sampling') is 0.834166527216 > 0.05 then FAIL to REJECT the NUL
The pvalue between ('NN', 'Hedge') is 0.914729760143 > 0.05 then FAIL to REJECT the NULL Hypothe
The pvalue between ('NN', 'EXP3') is 0.957287984229 > 0.05 then FAIL to REJECT the NULL Hypothes
The pvalue between ('E Greedy', 'Decay E Gr.') is 0.973343220006 > 0.05 then FAIL to REJECT the
The pvalue between ('E Greedy', 'Softmax') is 0.947720663397 > 0.05 then FAIL to REJECT the NULL
The pvalue between ('E Greedy', 'Decay SM') is 0.939058583318 > 0.05 then FAIL to REJECT the NUL
The pvalue between ('E Greedy', 'UCB1') is 0.973495378996 > 0.05 then FAIL to REJECT the NULL Hy
The pvalue between ('E Greedy', 'Tomp. Sampling') is 0.991137132378 > 0.05 then FAIL to REJECT t
The pvalue between ('E Greedy', 'Hedge') is 0.931101993606 > 0.05 then FAIL to REJECT the NULL H
The pvalue between ('E Greedy', 'EXP3') is 0.887797901108 > 0.05 then FAIL to REJECT the NULL Hy
The pvalue between ('Decay E Gr.', 'Softmax') is 0.973693432227 > 0.05 then FAIL to REJECT the N
The pvalue between ('Decay E Gr.', 'Decay SM') is 0.964949881528 > 0.05 then FAIL to REJECT the
The pvalue between ('Decay E Gr.', 'UCB1') is 0.946705759653 > 0.05 then FAIL to REJECT the NULL
The pvalue between ('Decay E Gr.', 'Tomp. Sampling') is 0.964347283985 > 0.05 then FAIL to REJEC
The pvalue between ('Decay E Gr.', 'Hedge') is 0.95665935027 > 0.05 then FAIL to REJECT the NULL
The pvalue between ('Decay E Gr.', 'EXP3') is 0.913073371134 > 0.05 then FAIL to REJECT the NULL
The pvalue between ('Softmax', 'Decay SM') is 0.99139550062 > 0.05 then FAIL to REJECT the NULL
The pvalue between ('Softmax', 'UCB1') is 0.921638047593 > 0.05 then FAIL to REJECT the NULL Hyp
The pvalue between ('Softmax', 'Tomp. Sampling') is 0.938833176055 > 0.05 then FAIL to REJECT th
The pvalue between ('Softmax', 'Hedge') is 0.982967816787 > 0.05 then FAIL to REJECT the NULL Hy
The pvalue between ('Softmax', 'EXP3') is 0.940188160788 > 0.05 then FAIL to REJECT the NULL Hyp
The pvalue between ('Decay SM', 'UCB1') is 0.913015059922 > 0.05 then FAIL to REJECT the NULL Hy
The pvalue between ('Decay SM', 'Tomp. Sampling') is 0.930156969964 > 0.05 then FAIL to REJECT t
The pvalue between ('Decay SM', 'Hedge') is 0.991487994504 > 0.05 then FAIL to REJECT the NULL H
The pvalue between ('Decay SM', 'EXP3') is 0.948747098488 > 0.05 then FAIL to REJECT the NULL Hy
The pvalue between ('UCB1', 'Tomp. Sampling') is 0.9822713914 > 0.05 then FAIL to REJECT the NUL
The pvalue between ('UCB1', 'Hedge') is 0.905355997874 > 0.05 then FAIL to REJECT the NULL Hypot
The pvalue between ('UCB1', 'EXP3') is 0.862124017776 > 0.05 then FAIL to REJECT the NULL Hypoth
The pvalue between ('Tomp. Sampling', 'Hedge') is 0.922279859064 > 0.05 then FAIL to REJECT the
The pvalue between ('Tomp. Sampling', 'EXP3') is 0.878863704312 > 0.05 then FAIL to REJECT the N
The pvalue between ('Hedge', 'EXP3') is 0.957710104054 > 0.05 then FAIL to REJECT the NULL Hypot
```

```
In [18]: matrix_twosample = []
```