Overview Data Kernels Discussion Leaderboard Rules Team My Submissions **Submit Predictions**

add submission details

output3.csv 4 hours ago by Junhao Pan add submission details	0.80528
output3.csv 4 hours ago by Junhao Pan add submission details	0.80425
output2.csv 4 hours ago by Junhao Pan add submission details	0.80671
output3.csv 4 hours ago by Junhao Pan add submission details	0.80630
output3.csv 4 hours ago by Junhao Pan add submission details	0.75900
output3.csv 4 hours ago by Junhao Pan	0.78153

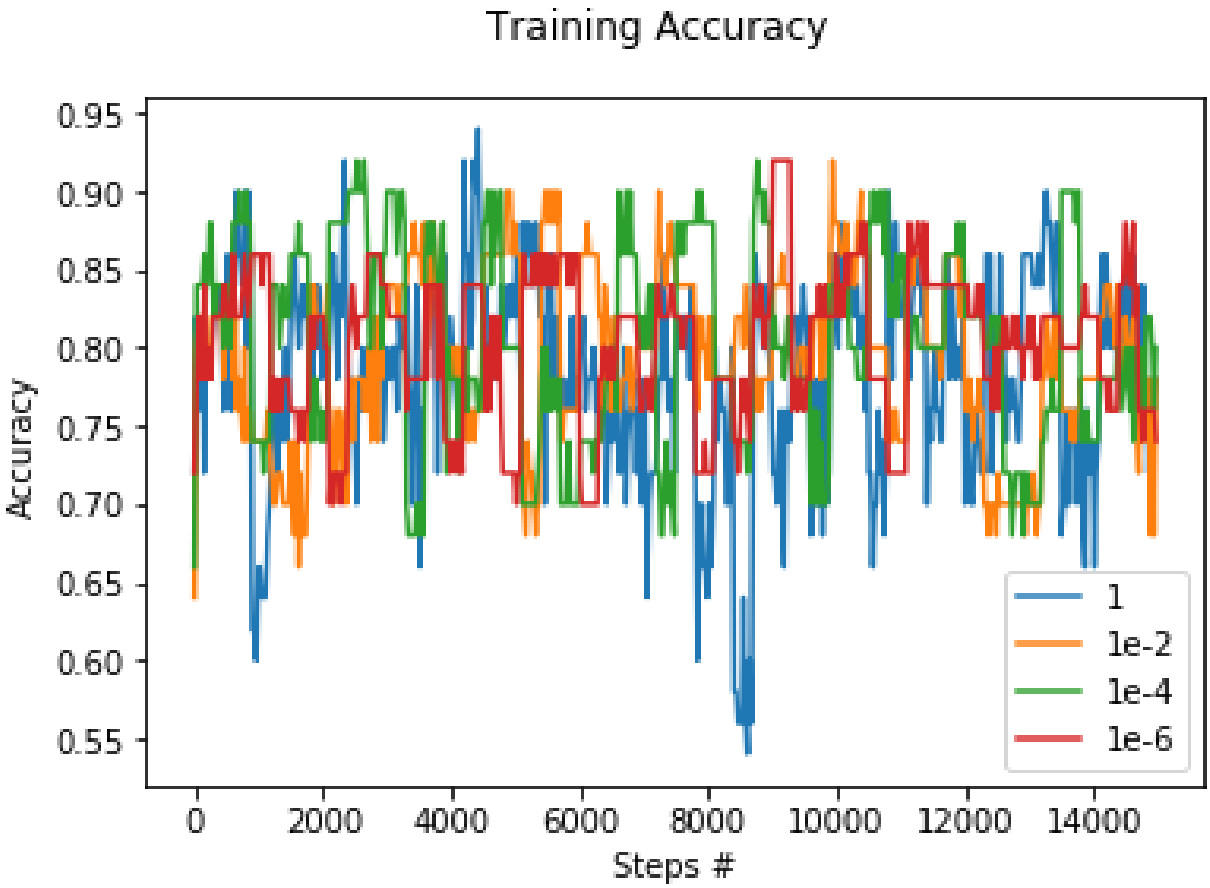
Overview Data Kernels Discussion Leaderboard Rules Team My Submissions **Submit Predictions**

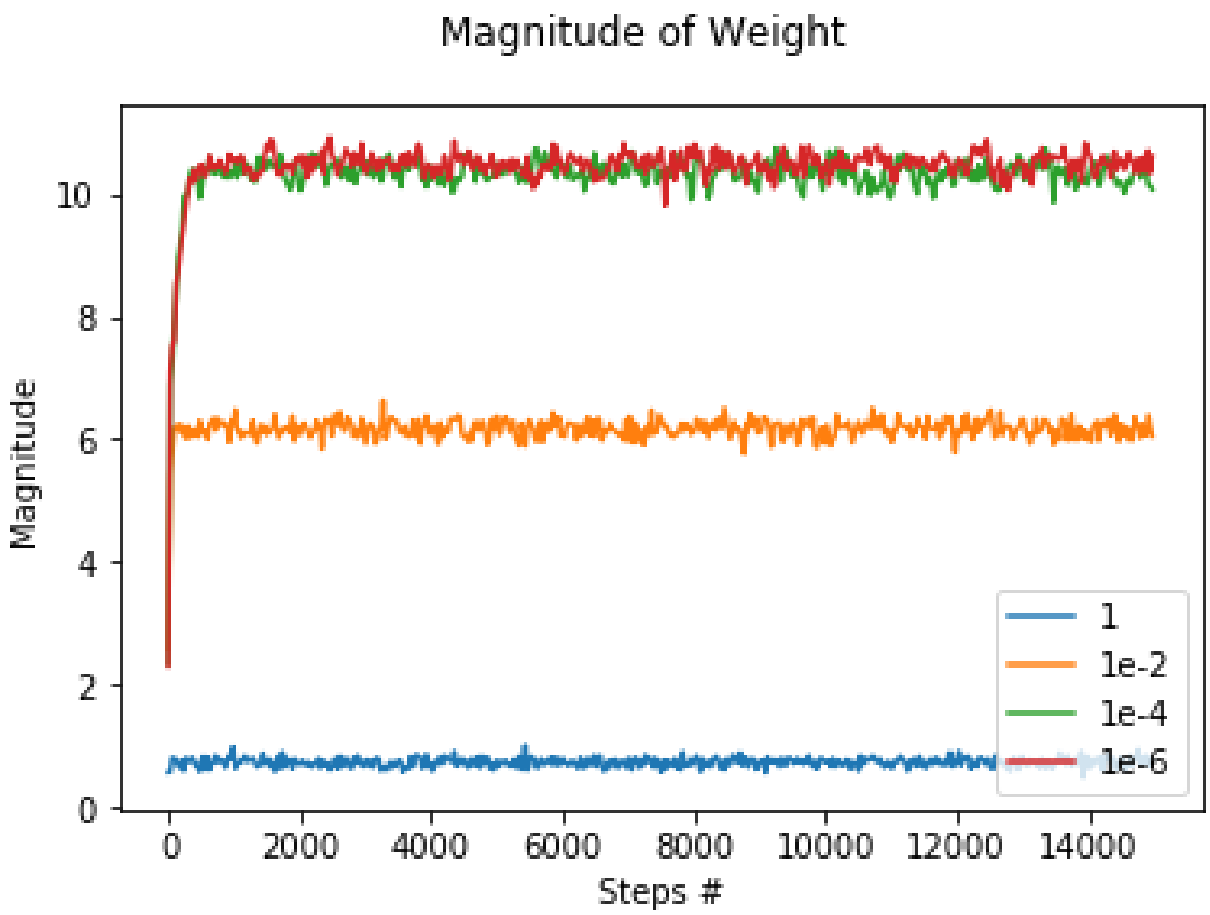
53	new	**Pan Zhang**	0.80692	9	9m
54	new	**Shuyue Lai**	0.80692	22	1d
55	new	**YuxuanRen**	0.80692	16	5h
56	new	**Mathias**	0.80692	1	6h
57	new	**JoJo**	0.80671	18	18h
58	new	**yanxu**	0.80671	16	7h
59	new	**Junhao Pan**	0.80671	20	3h

Your Best Entry ↑

Your submission scored 0.80589, which is not an improvement of your best score. Keep trying!

60	new	**Robert Jin**	0.80671	35	1h
61	new	**robertjin**	0.80671	1	1h
62	new	**Adam Dama**	0.80651	3	2h
63	new	**Hassan**	0.80630	3	3h
64	new	**Gauraang Naik**	0.80630	9	1h


Training Accuracy
Accuracy
Steps #
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
2000
4000
6000
8000
10000
12000
14000
1
1e-2
1e-4
1e-6


Magnitude of Weight
Magnitude
Steps #
0
2
4
6
8
10
0
2000
4000
6000
8000
10000
12000
14000
1
1e-2
1e-4
1e-6

The model is not overly sensitive to the choice of the lambda when it is around 1e-2 or less. In theory, smaller lambda would take longer time to stabilize themselves to establish a consistent accuracy. However, since we run 50 epochs on the dataset, it is enough for small lambda at even 1e-7 to stabilize.

My best accuracy is generated with a lambda of 1e-5. I have experienced with other numbers as well, but accuracy fluctuates only slightly.

In fact, there are other parameters which have more effects on the outcome. For example, the way to calculate the learning rate is quite a major impact on the result. I have changed the constants in the formula to control the update of the weight, but apparently the one documented by professor gave a consistent and good result. It uses the season's count as a variable to gradually decrease the step length.

jupyter P1p Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Notebook saved | Not Trusted | Python 3

Run | Code

In [1]:
```
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as plt
import argparse
from sklearn import preprocessing

train_file = open('train.data')
test_file = open('test.data')

epochs = 50
seasons = 300
steps = 145

d_train = []
d_label = []
train_file.seek(0)
for line in train_file:
    line = line.rstrip('\n')
    lc = line.split(', ')
    line_l = [lc[0], lc[2], lc[4], lc[10], lc[11], lc[12]]
    d_train.append(line_l)
    if (lc[14] == '<=50K'):
        d_label.append(-1)
    else:
        d_label.append(1)
d_train = preprocessing.scale(d_train)
d_train = np.c_[np.ones(len(d_train)), d_train, d_label]

d_test = []
test_file.seek(0)
for line in test_file:
    line = line.rstrip('\n')
    lc = line.split(',')
    line_l = [lc[0], lc[2], lc[4], lc[10], lc[11], lc[12]]
    d_test.append(line_l)
d_test = preprocessing.scale(d_test)
d_test = np.c_[np.ones(len(d_test)), d_test]

d_train = np.array(d_train, dtype = float)
d_test = np.array(d_test, dtype = float)

print(d_train.shape)
print(d_test.shape)
```

```
(43958, 8)
(4884, 7)
```

In [2]:
```
lam = [1, 0.01, 0.0001, 0.000001]

accuracy_plot = []
magnitude_plot = []
weights = []

for l in lam:
    print('Start training... lam = %.5f' % l)
    weight = np.random.uniform(size = (len(d_train[0]) - 1))
    step_accu_history = []
    mag_history = []

    for i in range(epochs):
        if (i%10 == 0):
            print('Training epoch:', i, '~', i + 10, '...')
        np.random.shuffle(d_train)
        e_train = d_train[50:, 0:7]
        e_train_label = d_train[50:, 7:8].flatten()
        e_val = d_train[0:50, 0:7]
        e_val_label = d_train[0:50, 7:8].flatten()

        for j in range(seasons):
            for k in range(steps):
                step_len = 1/(0.01*i + 50)
                row_idx = j*steps + k
                row = e_train[row_idx]
                label = e_train_label[row_idx]
                pred = row.dot(weight)
                if (label*pred >= 1):
                    weight = (1 - step_len*l)*weight
                else:
                    weight = weight - step_len*(l*weight - label*row)
#                     bias = bias + step_len*label
            if (j%30 == 0):
                preds_side = e_val.dot(weight)
                err_count = 0
                for n in range(len(e_val_label)):
                    if (preds_side[n]*e_val_label[n] < 0):
                        err_count += 1
                step_accu_history.append(1 - err_count/len(e_val_label))
                mag_history.append(la.norm(weight, 2))

    accuracy_plot.append(step_accu_history)
    magnitude_plot.append(mag_history)
    weights.append(weight)
```