ML_HW3

Mike Jaron

3/24/2017

mj2776

1.

a)

```
def kernals(x1, x2, b):
    kernal = np.zeros((len(x1),len(x2)))
    for i in range(len(x1)):
        for j in range(len(x2)):
            kernal[i][j] = np.exp((-1./b) * (np.linalg.norm([x1[i]] - x2[j], 2)**2))
    return kernal

def predict(k_XX, k_Xx, var):
    a = np.linalg.inv(var * np.identity(len(k_XX)) + k_XX)
    mu = np.dot(k_Xx, np.dot(a, y_train))
    return mu
```

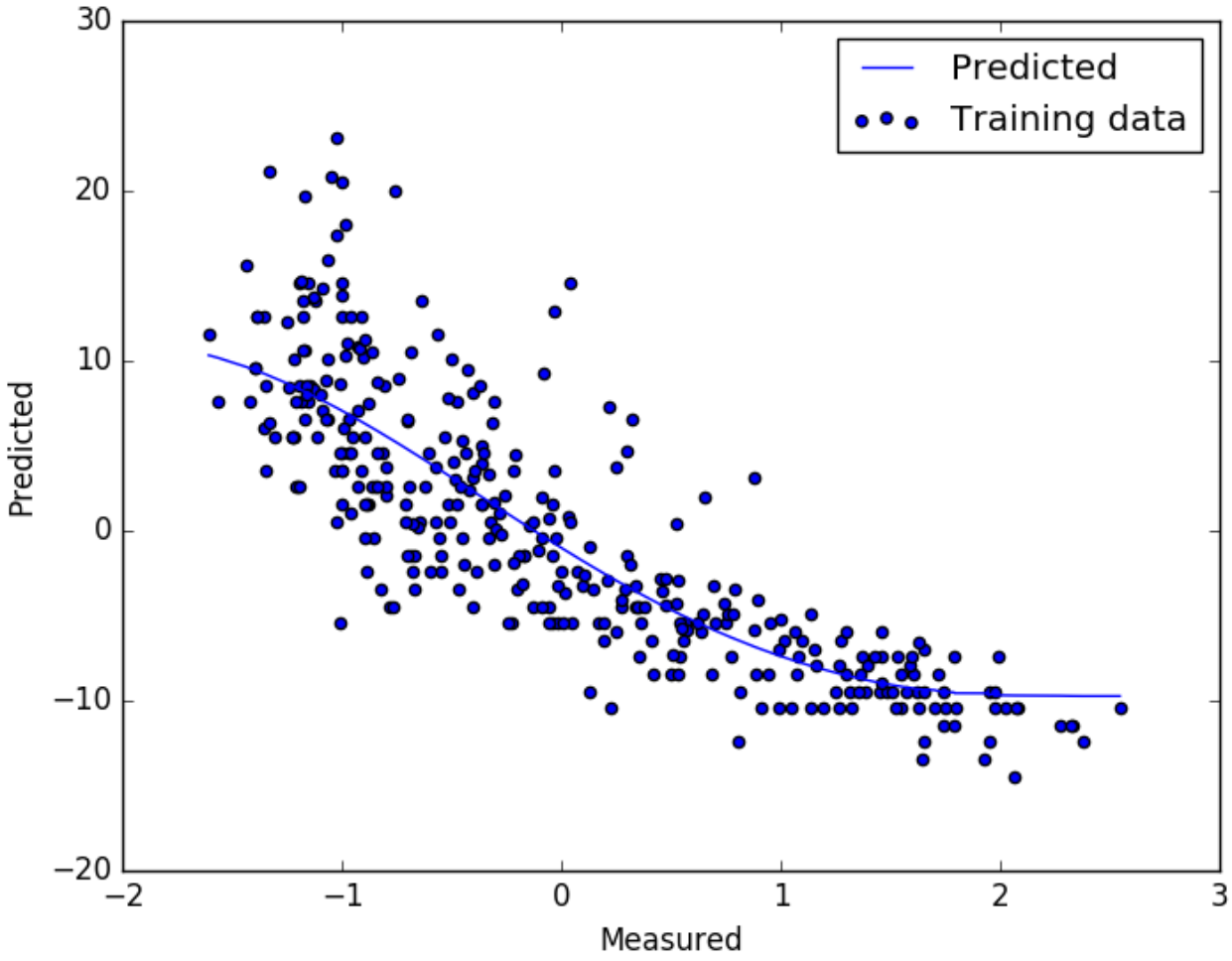
b)

	b	5	7	9	11	13	15
var							
0.1		1.966276	1.920163	1.897649	1.890507	1.895849	1.909603
0.2		1.933135	1.904877	1.902519	1.914981	1.935586	1.959549
0.3		1.923420	1.908080	1.917648	1.938849	1.964597	1.990804
0.4		1.922198	1.915902	1.932514	1.957936	1.985502	2.011915
0.5		1.924769	1.924804	1.945699	1.973216	2.001314	2.027370
0.6		1.929213	1.933701	1.957235	1.985764	2.013878	2.039465
0.7		1.934634	1.942254	1.967403	1.996375	2.024310	2.049463
0.8		1.940583	1.950380	1.976492	2.005603	2.033307	2.058105
0.9		1.946820	1.958093	1.984741	2.013835	2.041317	2.065845
1.0		1.953213	1.965438	1.992341	2.021345	2.048642	2.072976

c)

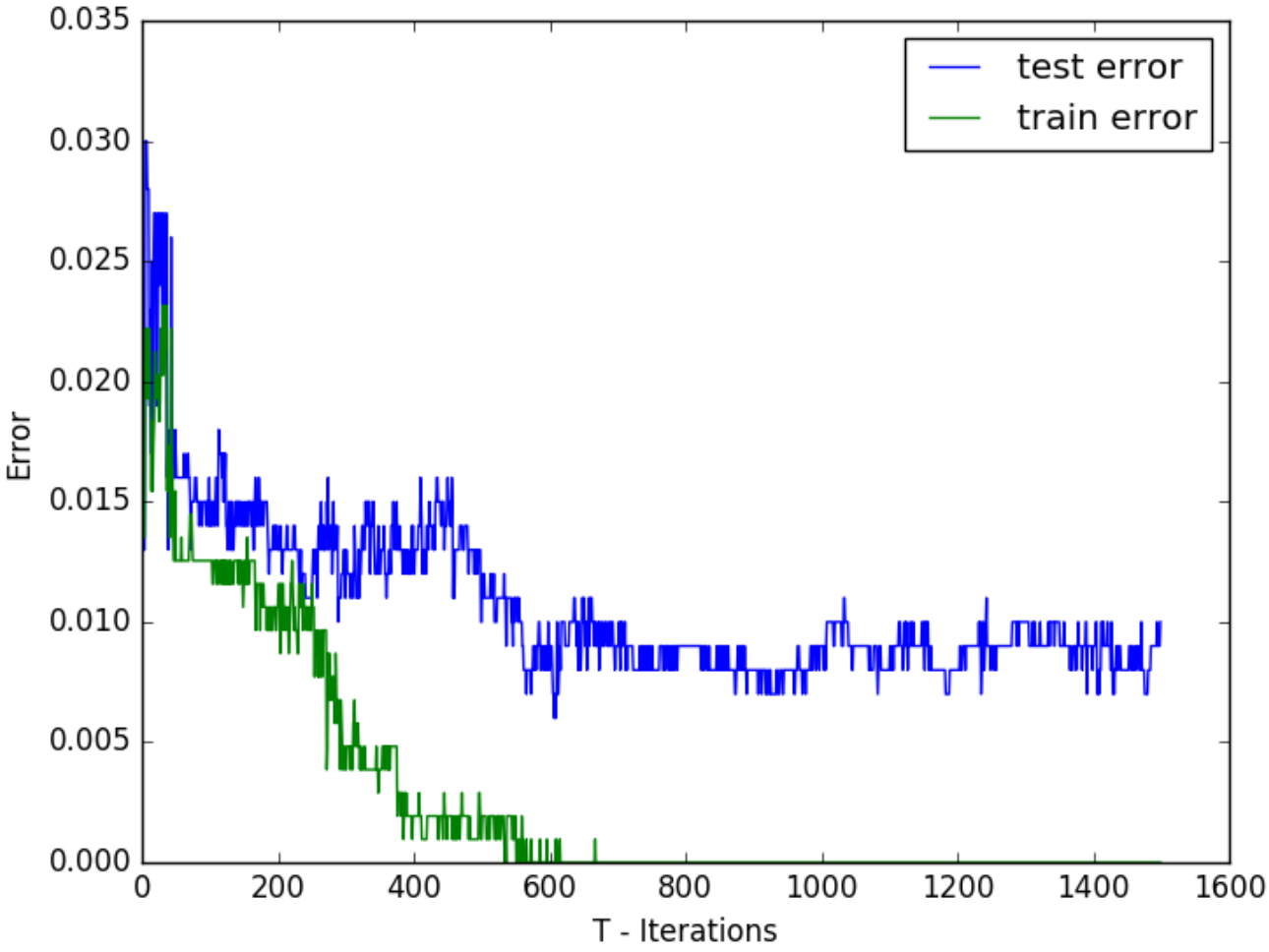
The best values of this algorithm was at b = 11 & var = .1, with those parameters the RMSE was ~1.891. Some possible downsides of this approach vs. HW1, is that now we need to tune 2 different hyper-parameters vs in HW1, we only tuned 1, lambda. Also this approach is a little more complex and thus takes more time to run and to understand the importance of the features.

d)

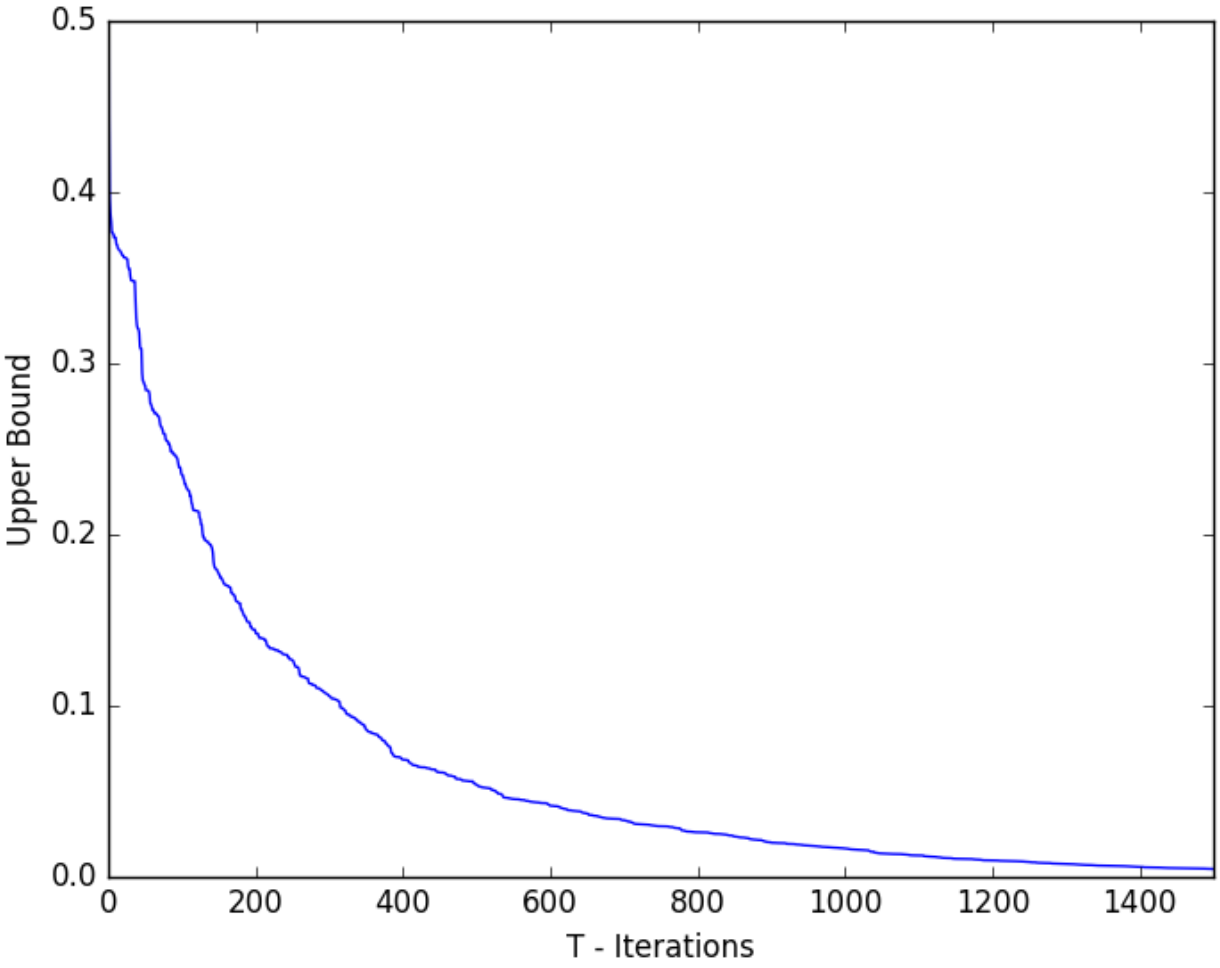


2.

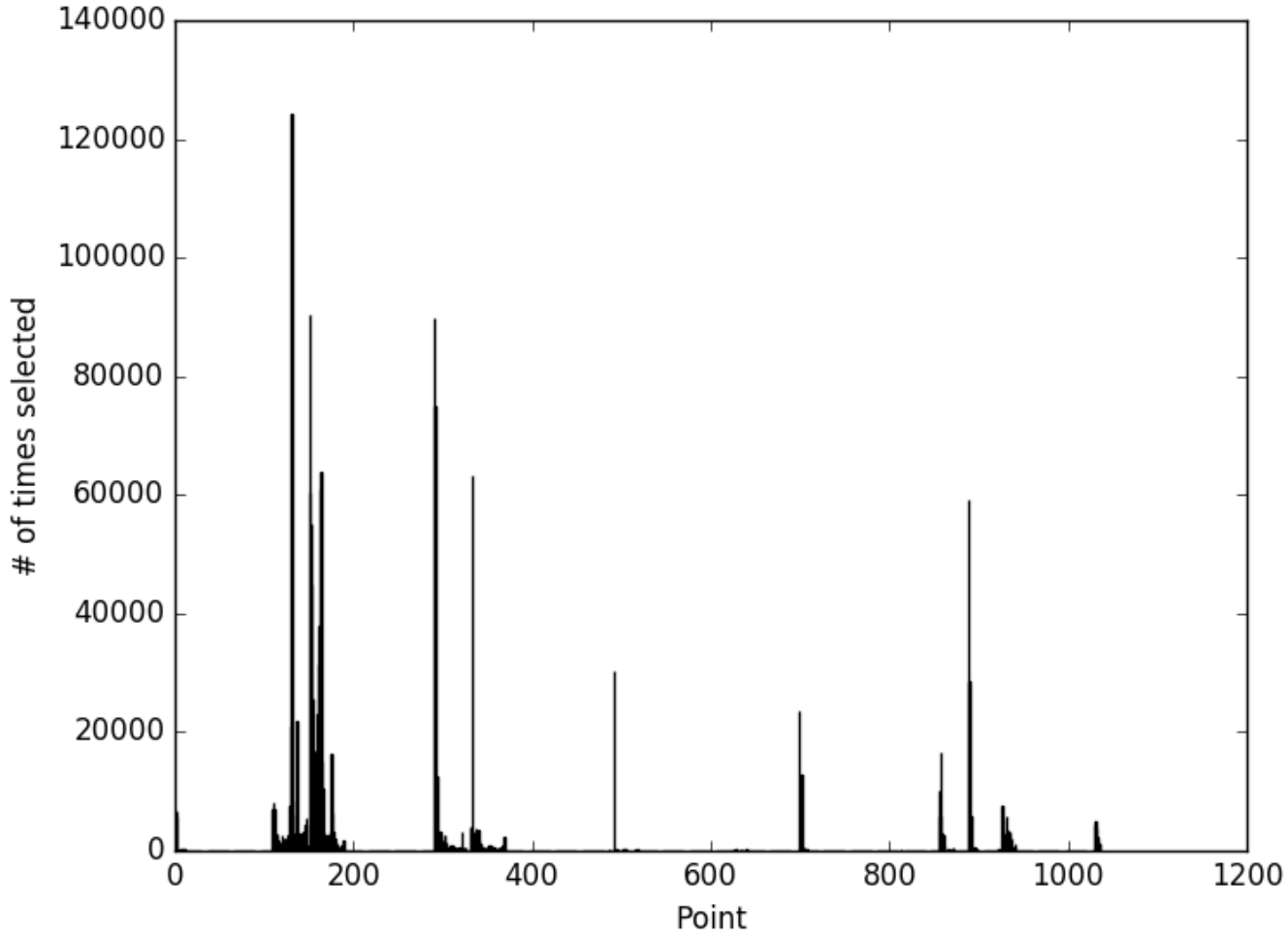
a)


test error
train error
Error
T - Iterations
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000
1200
1400
1600

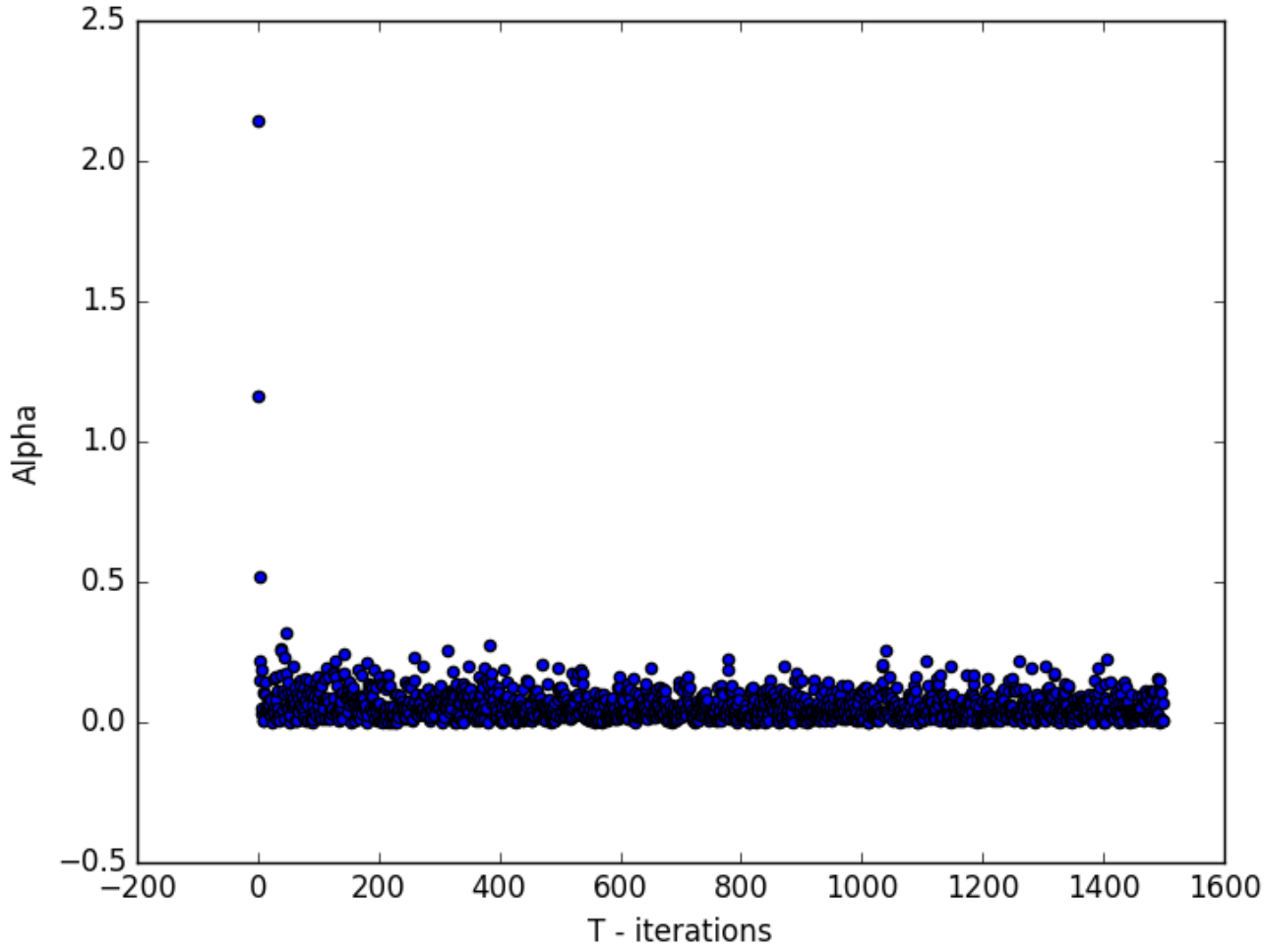
b)

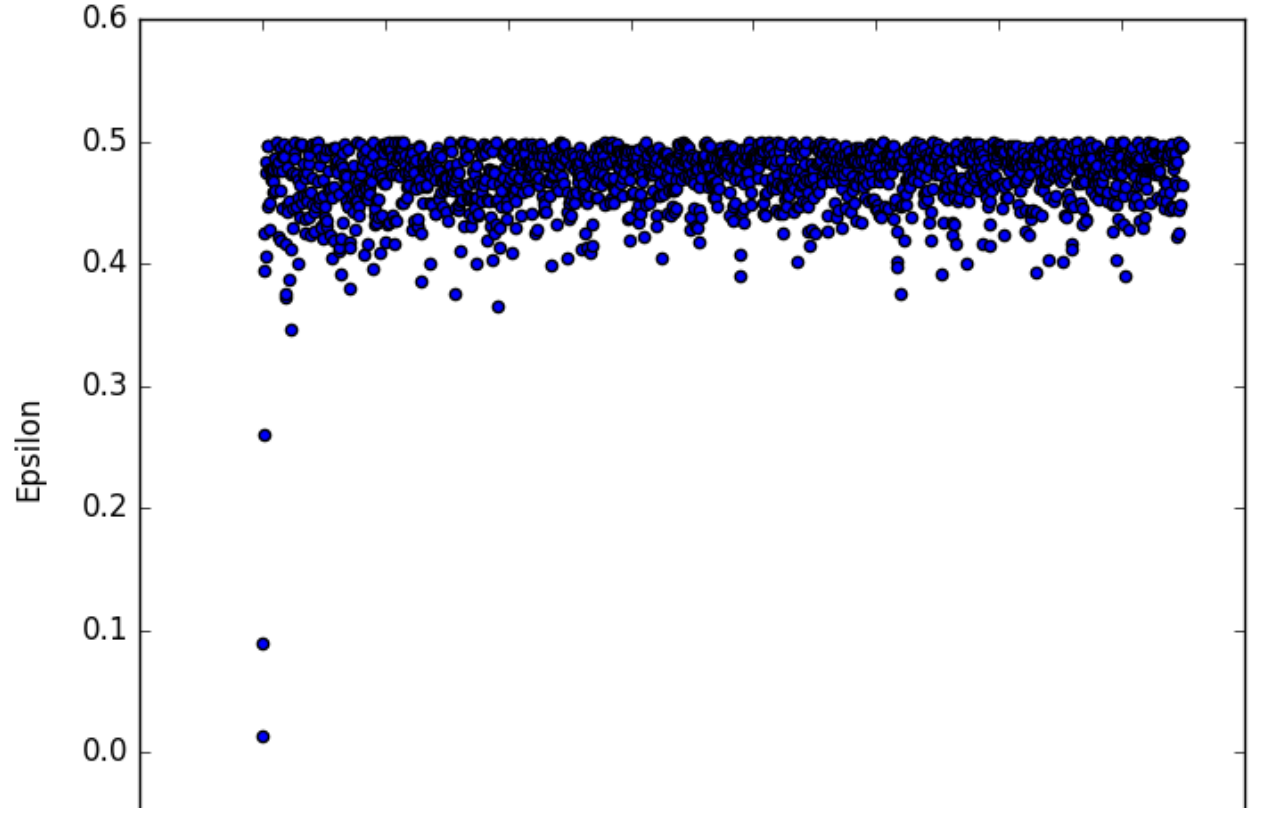

0.5
0.4
0.3
0.2
0.1
0.0
Upper Bound
0
200
400
600
800
1000
1200
1400
T - Iterations

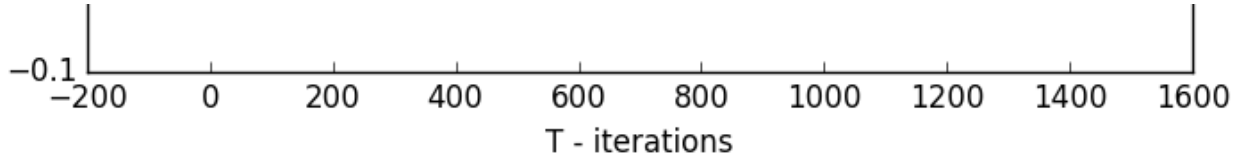
c)


140000
120000
100000
80000
60000
40000
20000
0
# of times selected
0
200
400
600
800
1000
1200
Point

d)


Alpha
T - iterations
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−200
0
200
400
600
800
1000
1200
1400
1600


Epsilon
0.6
0.5
0.4
0.3
0.2
0.1
0.0


−0.1
−200
0
200
400
600
800
1000
1200
1400
1600
T - iterations