

HW4

Friday, April 7, 2017 2:37 AM

1a. Let $K(x, y) = g(x)g(y)$ where g is a real valued function

If for any $\{x_1, \dots, x_n\} \subset X^d$

the matrix:

$$\begin{bmatrix} K(x_1, x_1) & \dots & K(x_1, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_1) & \dots & K(x_n, x_n) \end{bmatrix}$$

is symmetric PSD, then K is PSD

K is symmetric if $K(x, x') = K(x', x)$

Using $K(x, y) = g(x)g(y) = g(y)g(x)$, so K is symmetric

K is PSD if for any $a \in \mathbb{R}^n$, $a^T K a \geq 0$

$$\text{Let } K = \begin{bmatrix} g(x_1) & \dots & g(x_n) \end{bmatrix} \begin{bmatrix} g(x_1) \\ \vdots \\ g(x_n) \end{bmatrix}$$

so $K = x^T x$ where $x = [g(x_1) \dots g(x_n)]$

In HW3 we proved $a^T x^T x a$ will always be ≥ 0

$\therefore K$ is a PSD kernel

1c. $\|f\|_{\mathcal{H}}^2 = \langle f, f \rangle_{\mathcal{H}} \geq 0$, and we let.

and we let $f = K(x_f, \cdot)$

we have $\langle K(x_f, \cdot), K(x_f, \cdot) \rangle_{\mathcal{H}}$

and therefore $K(x_f, x_f) \geq 0$

so $K(x, y)$ satisfies the positivity

1d. If $0 < K(x, x) < \infty, \forall x$

Let the inner product be represented by $\langle f, f \rangle_{\mathcal{H}}$

We want to show if $\langle f, f \rangle_{\mathcal{H}} = 0$, then $f(x) = 0, \forall x$

The reproducing property is: $|f(x)| = |\langle f, K(x, \cdot) \rangle| \leq \|f\|_{\mathcal{H}} \sqrt{K(x, x)}$

so $|f(x)| \leq \langle f, f \rangle_{\mathcal{H}} \sqrt{K(x, x)}$

Because $K(x, x) > 0$, $|f(x)| \leq 0$ if $\langle f, f \rangle_{\mathcal{H}} = 0$

Because $|f(x)| \geq 0$ by definition, so if $\langle f, f \rangle_{\mathcal{H}} = 0$ then:

$$|f(x)| \leq 0 \text{ and } |f(x)| \geq 0$$

$$\text{so } |f(x)| = f(x) = 0$$

so $K(x, y)$ satisfies $\langle f, f \rangle_{\mathcal{H}} = 0 \Rightarrow f(x) = 0, \forall x$

Training Error and Validation Error (respectively)

2a.

$$f(w) = \frac{1}{N} \sum_p \max\{0, 1 - y_p w^T x_p\} + \frac{\lambda}{2} w^T w$$

$$\frac{d}{dw} f(w) = \frac{1}{N} \sum_p \max\{0, -x_p^T y_p\} + \lambda w$$

2b. Code shown below

2c and 2d.

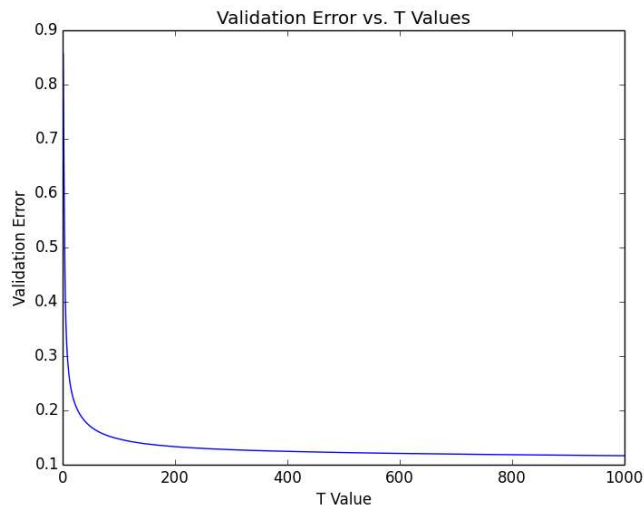
Eta values tried: 1e-01, 1e-02, 1e-03, 1e-04, 1e-05, 1e-06, 1e-07

Lambda values tried: 1e-01, 1e-02, 1e-03, 1e-04, 1e-05, 1e-06, 1e-07

Best eta value: 1e-06

Best lambda value: 1e-07

Best T: 1000



2e. Test error: 0.107528

3a.

$$f(a) = \frac{1}{N} \sum_p \max\{0, 1 - y_p((Ka)_p + b)\} + \frac{\lambda}{2} a^T Ka$$

$$\frac{d}{da} f(a) = \frac{1}{N} \sum_p \max\{0, -K_p^T y_p\} + \lambda Ka$$

3b. Code included below

3c and 3d.

Eta values tried: 1e-08, 1e-09, 1e-10, 1e-11, 1e-12, 1e-13, 1e-14

Lambda values tried: 1e-08, 1e-09, 1e-10, 1e-11, 1e-12, 1e-13, 1e-14

Out of all of the combinations attempted, the best results were:

Best eta value: 1e-14

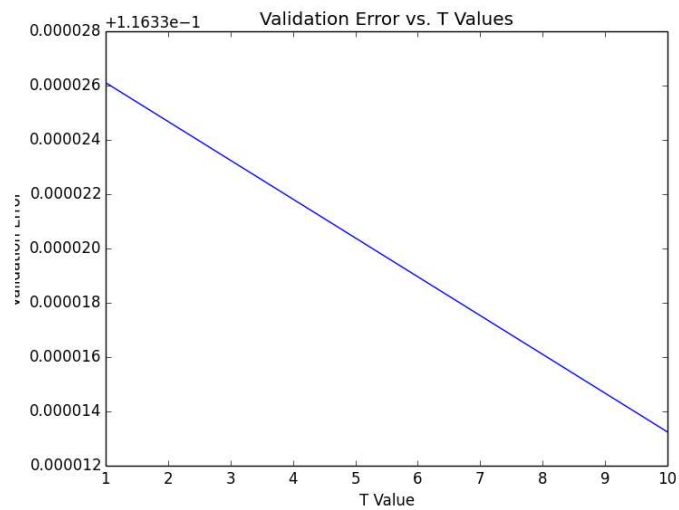
Best lambda value: 1e-14

Best Training error: 0.116

Best Validation error: 0.116

		eta						
		1e-01	1e-02	1e-03	1e-04	1e-05	1e-06	1e-07
	1e-01	438.860	47.414	2.540	0.242	0.085	0.104	0.147
	1e-02	501.446	26.544	2.626	0.225	0.089	0.103	0.147
lambda	1e-03	210.441	35.640	2.913	0.233	0.087	0.104	0.147
	1e-04	285.370	37.176	2.843	0.234	0.084	0.104	0.147
	1e-05	427.453	39.242	2.889	0.234	0.085	0.104	0.147
	1e-06	460.390	24.646	2.639	0.234	0.085	0.104	0.147
	1e-07	311.453	24.646	2.639	0.234	0.085	0.104	0.147

		eta						
		1e-01	1e-02	1e-03	1e-04	1e-05	1e-06	1e-07
	1e-01	502.709	63.591	4.442	0.446	0.118	0.117	0.148
	1e-02	661.250	45.695	4.600	0.427	0.119	0.117	0.148
lambda	1e-03	394.044	55.187	4.919	0.438	0.120	0.117	0.148
	1e-04	484.868	56.881	4.844	0.440	0.118	0.117	0.148
	1e-05	620.746	58.969	4.894	0.440	0.118	0.117	0.148
	1e-06	648.228	44.747	4.628	0.440	0.118	0.117	0.148
	1e-07	511.779	44.747	4.628	0.440	0.118	0.117	0.148



CODE SECTION

LINEAR SVM

Setup code was the same as HW3, so did not include in this submission.

linsvm.py - contains the loss function and gradient for linear SVM

gradientdescent.py - contains the gradient descent algorithm for linear SVM

```
1. import gradientdescent as gd
2. import numpy as np
3.
4. def loss(X, Y, w, l, N):
5.     loss = 0
6.     for i in range(N):
7.         h = float(w.T * X[i].T * Y[i])
8.         if h < 1:
9.             loss += 1 - h
10.
11.     loss *= 1.0 / N
12.     loss += (l / 2) * w.T * w
13.
14.     return loss
15.
16. def gradient(X, Y, w, eta, l, N):
17.     gsum = 0
18.     for i in range(N):
19.         h = float(w.T * X[i].T * Y[i])
20.         if h < 1:
21.             gsum += -X[i].T * Y[i]
22.
23. lambda_values = [1e-5, 1e-6, 1e-7, 1e-8, 1e-9, 1e-10, 1e-11]
24. eta_values = [1e-5, 1e-6, 1e-7, 1e-8, 1e-9, 1e-10, 1e-11]
25.
26. lambda_values = [1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7]
27. eta_values = [1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7]
28.
29. gd.run_experiment(loss, gradient, eta_values, lambda_values)
30.
```

```
1. import numpy as np
2. import os
3. import os.path
4. import pickle
5. import matplotlib.pyplot as plt
6.
7. import setup as s
8.
9. (training_X, training_Y, training_N, val_X, val_Y, val_N, test_X, test_Y,
   test_N) = s.run()
10.
11. def train_increment(gradient, w, eta, l):
12.     return gradient(training_X, training_Y, w, eta, l, training_N)
13.
14. def train(gradient, T, eta, l):
15.     w = np.zeros((785,1))
16.     for t in range(0, T):
17.         w = train_increment(gradient, w, eta, l)
18.
19.     return w
20.
21. def train_err(loss, w, l):
22.     return float(loss(training_X, training_Y, w, l, training_N))
23.
24. def val_err(loss, w, l):
25.     return float(loss(val_X, val_Y, w, l, val_N))
26.
27. def test_err(loss, w, l):
28.     return float(loss(test_X, test_Y, w, l, test_N))
29.
30. def run_experiment(loss, gradient, eta_values, lambda_values):
31.     T = 1000
32.     best_combo = (0, 0)
33.     lowest_err = float("inf")
34.
35.     for l in lambda_values:
36.         for eta in eta_values:
37.             w = train(gradient, T, eta, l)
38.
39.             tr_err = train_err(loss, w, l)
40.             v_err = val_err(loss, w, l)
41.
42.             print "eta: " + str(eta)
43.             print "lambda: " + str(l)
44.             print "training error: " + str(tr_err)
45.             print "validation error: " + str(v_err)
46.             print
47.
48.             if v_err < lowest_err:
49.                 best_combo = (eta, l)
50.                 lowest_err = v_err
```

```
51.
52.     best_T = 0
53.     lowest_err = float("inf")
54.
55.     max_T = 1000
56.
57.     T_values = range(1, max_T + 1)
58.     err_values = []
59.
60.     w = np.zeros((785,1))
61.
62.     best_eta = best_combo[0]
63.     best_lambda = best_combo[1]
64.
65.     for T in T_values:
66.         w = train_increment(gradient, w, best_eta, best_lambda)
67.
68.         err = val_err(loss, w, best_lambda)
69.
70.         err_values.append(err)
71.
72.         if err < lowest_err:
73.             best_T = T
74.             lowest_err = err
75.
76.
77.     w = train(gradient, best_T, best_eta, best_lambda)
78.
79.     print "best T: " + str(best_T)
80.     print "best eta: " + str(best_eta)
81.     print "best lambda: " + str(best_lambda)
82.     print "test error: " + str(test_err(loss, w, best_lambda))
83.
84.     plt.plot(T_values, err_values, '-')
85.
86.     plt.title('Validation Error vs. T Values')
87.     plt.xlabel('T Value')
88.     plt.ylabel('Validation Error')
89.
90.     plt.show()
91.
```

KERNEL SVM

kernsvm.py - contains the loss function and gradient for kernel SVM

gradientdescent.py - contains the algorithm for gradient descent for kernel SVM

```

1. import gradientdescent as gd
2. import numpy as np
3.
4. def loss(K, Y, w, l, N):
5.     loss = 0
6.     for i in range(N):
7.         h = float(Y[i] * (K*w)[i])
8.         if h < 1:
9.             loss += 1 - h
10.
11.     loss *= 1.0 / N
12.     loss += (l / 2) * w.T * K * w
13.
14.     return loss
15.
16. def gradient(K, Y, w, eta, l, N):
17.     gsum = 0
18.     for i in range(N):
19.         h = float(Y[i] * (K*w)[i])
20.         if h < 1:
21.             gsum += K[i].T * Y[i]
22.
23.     return w - eta * ((1.0 / N) * gsum + l * K * w)
24.
25. lambda_values = [1e-08, 1e-09, 1e-10, 1e-11, 1e-12, 1e-13, 1e-14]
26. eta_values = [1e-08, 1e-09, 1e-10, 1e-11, 1e-12, 1e-13, 1e-14]
27.
28. gd.run_experiment(loss, gradient, eta_values, lambda_values)
29.

```



```

1. import numpy as np
2. import os
3. import os.path
4. import pickle
5. import matplotlib.pyplot as plt
6.
7. import setup as s
8.
9. (training_X, training_Y, training_N, val_X, val_Y, val_N, test_X, test_Y,
    test_N) = s.run()
10.
11. training_K = np.power((training_X * training_X.T + 1), 3)
12. val_K = np.power((training_X * val_X.T + 1), 3)
13. test_K = np.power((training_X * test_X.T + 1), 3)
14.
15. training_KY = training_K * training_Y
16. val_KY = val_K * val_Y
17. test_KY = test_K * test_Y
18.
19. print 'kernel matrices built'
20.
21. def train_increment(gradient, w, eta, l):
22.     return gradient(training_K, training_Y, w, eta, l, training_N)
23.
24. def train(gradient, T, eta, l):
25.     w = np.zeros((training_N,1))
26.     for t in range(0, T):
27.         w = train_increment(gradient, w, eta, l)
28.
29.     return w
30.
31. def train_err(loss, w, l):
32.     return float(loss(training_K, training_Y, w, l, training_N))
33.
34. def val_err(loss, w, l):
35.     return float(loss(val_K, val_Y, w, l, val_N))
36.
37. def test_err(loss, w, l):
38.     return float(loss(test_K, test_Y, w, l, test_N))
39.
40. def run_experiment(loss, gradient, eta_values, lambda_values):
41.     T = 10
42.     best_combo = (0, 0)
43.     lowest_err = float("inf")
44.
45.     for l in lambda_values:
46.         for eta in eta_values:
47.             w = train(gradient, T, eta, l)
48.
49.             tr_err = train_err(loss, w, l)

```

1/3

```

50.         v_err = val_err(loss, w, l)
51.
52.         print "eta: " + str(eta)
53.         print "lambda: " + str(l)
54.         print "training error: " + str(tr_err)
55.         print "validation error: " + str(v_err)
56.         print
57.
58.         if v_err < lowest_err:
59.             best_combo = (eta, l)
60.             lowest_err = v_err
61.
62.     best_T = 0
63.     lowest_err = float("inf")
64.
65.     max_T = 10
66.
67.     T_values = range(1, max_T + 1)
68.     err_values = []
69.
70.     w = np.zeros((training_N, 1))
71.
72.     best_eta = best_combo[0]
73.     best_lambda = best_combo[1]
74.
75.     for T in T_values:
76.         w = train_increment(gradient, w, best_eta, best_lambda)
77.
78.         err = val_err(loss, w, best_lambda)
79.
80.         err_values.append(err)
81.
82.         if err < lowest_err:
83.             best_T = T
84.             lowest_err = err
85.
86.
87.     w = train(gradient, best_T, best_eta, best_lambda)
88.
89.     print "best T: " + str(best_T)
90.     print "best eta: " + str(best_eta)
91.     print "best lambda: " + str(best_lambda)
92.     # print "test error: " + str(test_err(loss, w, best_lambda))
93.
94.     plt.plot(T_values, err_values, '-')
95.
96.     plt.title('Validation Error vs. T Values')
97.     plt.xlabel('T Value')
98.     plt.ylabel('Validation Error')
99.

```

```
100. plt.show()  
101.  
102.
```