UMJI **VE445** Introduction to Machine Learning

LAB 1 SVM

Wu Guangzheng

1 Basic SVM

I ran the linear SVM on the data set 1, and the result was 100% correct. I use the same code for basic SVM and the Kernel SVM with linear kernel, since the dual problem of these two are exactly the same.

```
Work until i  0  out of total num  18731
Work until i  5000  out of total num  18731
Work until i  10000  out of total num  18731
Work until i  15000  out of total num  18731
     pcost       dcost       gap    pres   dres
 0: -5.5440e+03 -1.0628e+04  1e+05  4e+02  2e+00
 1: -1.7603e+04 -1.2291e+04  4e+04  2e+02  7e-01
 2: -2.4213e+04 -1.6064e+04  4e+04  2e+02  7e-01
 3: -3.1236e+04 -1.9153e+04  4e+04  1e+02  6e-01
 4: -2.8588e+04 -1.1299e+04  4e+04  9e+01  4e-01
 5: -4.5277e+03 -4.1418e+02  1e+04  2e+01  8e-02
 6: -9.1395e+01 -3.2574e+01  2e+02  3e-01  1e-03
 7: -2.1668e+01 -2.9119e+01  7e+00  7e-14  1e-13
 8: -2.6213e+01 -2.7453e+01  1e+00  7e-14  8e-14
 9: -2.6380e+01 -2.7462e+01  1e+00  3e-15  8e-14
10: -2.7092e+01 -2.7396e+01  3e-01  2e-13  1e-13
11: -2.7330e+01 -2.7381e+01  5e-02  3e-13  1e-13
12: -2.7378e+01 -2.7379e+01  8e-04  2e-13  1e-13
13: -2.7379e+01 -2.7379e+01  8e-06  4e-14  1e-13
Optimal solution found.
1.0
16181
```

2 Kernel SVM

I applied four different kernels for kernel SVM, linear, Gaussian, Laplace and Polynominal. The results are shown below. Here are linear and Gaussian.

```
Work until i  0  out of total num  16181
Work until i  5000  out of total num  16181
Work until i  10000  out of total num  16181
Work until i  15000  out of total num  16181
     pcost       dcost       gap    pres   dres
 0: -1.6023e+04 -3.4211e+04  2e+04  8e-13  2e+00
 1: -3.7427e+04 -3.7852e+04  4e+02  2e-12  1e+00
 2: -3.3380e+06 -3.3384e+06  4e+02  2e-10  1e+00
 3: -2.3570e+10 -2.3571e+10  5e+05  4e-07  1e+00
Terminated (singular KKT matrix).
0.5390715109573241
16181
```

```
     pcost       dcost       gap    pres   dres
 0: -2.3561e+03 -4.4102e+03  5e+04  2e+02  2e+00
 1: -2.7584e+03 -2.0696e+03  2e+04  8e+01  6e-01
 2: -1.7177e+03 -8.5841e+02  1e+04  4e+01  3e-01
 3: -2.9261e+02 -2.0791e+02  2e+03  7e+00  5e-02
 4: -9.9095e+00 -1.4397e+02  1e+02  9e-15  3e-14
 5: -5.1141e+01 -1.1801e+02  7e+01  2e-14  2e-14
 6: -6.4249e+01 -1.2066e+02  6e+01  2e-14  2e-14
 7: -9.5867e+01 -1.1578e+02  2e+01  9e-14  2e-14
 8: -1.0722e+02 -1.1303e+02  6e+00  1e-14  3e-14
 9: -1.1104e+02 -1.1277e+02  2e+00  6e-14  4e-14
10: -1.1235e+02 -1.1270e+02  3e-01  6e-14  4e-14
11: -1.1266e+02 -1.1269e+02  3e-02  2e-13  5e-14
12: -1.1268e+02 -1.1268e+02  5e-03  1e-13  5e-14
13: -1.1268e+02 -1.1268e+02  2e-04  3e-13  5e-14
14: -1.1268e+02 -1.1268e+02  4e-06  2e-13  5e-14
Optimal solution found.
1.0
```

And Laplace and Polynominal as follow.

```
Work until i  0  out of total num  16181
Work until i  5000  out of total num  16181
Work until i  10000  out of total num  16181
Work until i  15000  out of total num  16181
     pcost       dcost       gap    pres   dres
 0: -4.0222e+02 -9.6660e+02  3e+04  1e+02  1e+00
 1: -3.5165e+02 -6.7215e+02  4e+03  2e+01  2e-01
 2: -2.7707e+02 -3.5462e+02  2e+03  9e+00  9e-02
 3: -1.8019e+02 -1.9887e+02  7e+02  3e+00  3e-02
 4: -7.2820e+01 -1.2977e+02  8e+01  9e-02  9e-04
 5: -9.6858e+01 -1.1838e+02  2e+01  4e-03  4e-05
 6: -1.1003e+02 -1.1463e+02  5e+00  3e-04  3e-06
 7: -1.1366e+02 -1.1402e+02  4e-01  1e-05  1e-07
 8: -1.1397e+02 -1.1398e+02  1e-02  5e-07  5e-09
 9: -1.1398e+02 -1.1398e+02  4e-04  1e-08  1e-10
10: -1.1398e+02 -1.1398e+02  9e-06  2e-10  2e-12
Optimal solution found.
1.0
16181
```

```
Work until i  0  out of total num  16181
Work until i  5000  out of total num  16181
Work until i  10000  out of total num  16181
Work until i  15000  out of total num  16181
     pcost       dcost       gap    pres   dres
 0: -2.5020e+02 -6.3444e+02  2e+04  1e+02  1e+00
 1: -1.8333e+02 -4.7652e+02  2e+03  1e+01  1e-01
 2: -1.2992e+02 -2.3280e+02  5e+02  2e+00  2e-02
 3: -8.7553e+01 -1.5503e+02  2e+02  5e-01  5e-03
 4: -9.6571e+01 -1.2319e+02  5e+01  1e-01  1e-03
 5: -1.0848e+02 -1.1681e+02  1e+01  1e-02  1e-04
 6: -1.1387e+02 -1.1544e+02  2e+00  1e-03  1e-05
 7: -1.1514e+02 -1.1525e+02  1e-01  8e-05  8e-07
 8: -1.1523e+02 -1.1524e+02  3e-03  2e-06  2e-08
 9: -1.1524e+02 -1.1524e+02  8e-05  4e-08  3e-10
Optimal solution found.
1.0
```

The error rate of linear kernel is very high, because the data is not linearly separatable.

3 Soft Margin SVM

By applying the Soft Margin SVM from the manual onto our model, and train via the data set 3, we get the following result, with $C = 0.1$.

```
Work until i  0  out of total num  21000
Work until i  5000  out of total num  21000
Work until i  10000  out of total num  21000
Work until i  15000  out of total num  21000
Work until i  20000  out of total num  21000
     pcost       dcost       gap    pres   dres
 0: -8.7373e+03 -8.4409e+03  6e+05  5e+01  2e-09
 1: -1.7487e+03 -8.2183e+03  6e+04  5e+00  2e-09
 2: -9.7349e+02 -7.8116e+03  2e+04  1e+00  4e-10
 3: -8.1166e+02 -6.2175e+03  1e+04  5e-01  2e-10
 4: -7.1014e+02 -3.7343e+03  5e+03  2e-01  1e-10
 5: -6.1726e+02 -2.0109e+03  2e+03  1e-01  5e-11
 6: -5.5916e+02 -1.3361e+03  1e+03  6e-02  3e-11
 7: -5.1598e+02 -9.9408e+02  1e+03  4e-02  3e-11
 8: -4.8790e+02 -8.5997e+02  8e+02  3e-02  3e-11
 9: -4.6155e+02 -7.4645e+02  7e+02  2e-02  2e-11
10: -4.4036e+02 -6.7853e+02  6e+02  2e-02  2e-11
11: -4.2211e+02 -6.3159e+02  5e+02  1e-02  2e-11
12: -4.0859e+02 -5.9867e+02  5e+02  1e-02  2e-11
13: -3.9619e+02 -5.7168e+02  5e+02  1e-02  2e-11
14: -3.8490e+02 -5.4620e+02  4e+02  1e-02  2e-11
15: -3.7433e+02 -5.2452e+02  4e+02  9e-03  2e-11
16: -3.6672e+02 -5.1345e+02  4e+02  8e-03  2e-11
17: -3.5529e+02 -4.9152e+02  4e+02  7e-03  2e-11
18: -3.4768e+02 -4.7744e+02  4e+02  6e-03  2e-11
19: -3.3920e+02 -4.6140e+02  3e+02  6e-03  2e-11
20: -3.3254e+02 -4.5056e+02  3e+02  5e-03  2e-11
21: -3.2644e+02 -4.4148e+02  3e+02  5e-03  2e-11
22: -3.2076e+02 -4.3192e+02  3e+02  4e-03  2e-11
23: -3.1449e+02 -4.2069e+02  3e+02  4e-03  2e-11
24: -3.0941e+02 -4.1285e+02  3e+02  4e-03  2e-11
25: -3.0369e+02 -4.0271e+02  3e+02  3e-03  2e-11
26: -2.9835e+02 -3.9534e+02  3e+02  3e-03  2e-11
27: -2.9216e+02 -3.8516e+02  3e+02  3e-03  2e-11
```

```
28: -2.8822e+02 -3.8081e+02  3e+02  3e-03  2e-11
29: -2.8296e+02 -3.7265e+02  3e+02  2e-03  2e-11
30: -2.7837e+02 -3.6678e+02  3e+02  2e-03  2e-11
31: -2.7242e+02 -3.5842e+02  2e+02  2e-03  2e-11
32: -2.6653e+02 -3.5062e+02  2e+02  2e-03  2e-11
33: -2.6098e+02 -3.4348e+02  2e+02  2e-03  2e-11
34: -2.5717e+02 -3.3924e+02  2e+02  1e-03  2e-11
35: -2.5237e+02 -3.3151e+02  2e+02  1e-03  2e-11
36: -2.4779e+02 -3.2618e+02  2e+02  1e-03  2e-11
37: -2.4295e+02 -3.2054e+02  2e+02  1e-03  2e-11
38: -2.3666e+02 -3.1764e+02  2e+02  9e-04  2e-11
39: -2.3225e+02 -3.1568e+02  2e+02  8e-04  2e-11
40: -2.2910e+02 -3.0862e+02  2e+02  7e-04  2e-11
41: -2.2612e+02 -3.0530e+02  2e+02  6e-04  2e-11
42: -2.2309e+02 -3.0165e+02  2e+02  6e-04  2e-11
43: -2.2036e+02 -2.9767e+02  1e+02  5e-04  2e-11
44: -2.1692e+02 -2.9280e+02  1e+02  4e-04  2e-11
45: -2.1487e+02 -2.9078e+02  1e+02  4e-04  2e-11
46: -2.1264e+02 -2.7896e+02  1e+02  2e-04  2e-11
47: -2.1181e+02 -2.7356e+02  9e+01  2e-04  2e-11
48: -2.1527e+02 -2.5927e+02  6e+01  1e-04  2e-11
49: -2.1615e+02 -2.5258e+02  5e+01  9e-05  3e-11
50: -2.1885e+02 -2.4112e+02  3e+01  4e-05  3e-11
51: -2.2140e+02 -2.3380e+02  1e+01  9e-06  3e-11
52: -2.2385e+02 -2.2972e+02  6e+00  2e-06  3e-11
53: -2.2492e+02 -2.2824e+02  3e+00  7e-07  3e-11
54: -2.2559e+02 -2.2740e+02  2e+00  2e-07  3e-11
55: -2.2619e+02 -2.2675e+02  6e-01  4e-08  3e-11
56: -2.2640e+02 -2.2652e+02  1e-01  7e-09  3e-11
57: -2.2645e+02 -2.2647e+02  1e-02  6e-10  3e-11
58: -2.2646e+02 -2.2646e+02  8e-04  3e-11  3e-11
59: -2.2646e+02 -2.2646e+02  1e-05  5e-13  3e-11
Optimal solution found.
0.9605555555555556
```

The error rate is very low, but the training process is so tedious, where I let it run before I go to bed, and it remained unfinished after I got up next morning, and it took the convex optimizor a long time to get the optimal value. So we need to introduce some other ways to solve this problem. You can view the code in class SoftMargin.

4 ISVM and LSVM

I also implemented the LSVM, which is an update of ISVM, and other than the functions provided by the Soft Margin SVM, LSVM and ISVM can support online training, which

significantly reduce the resourses needed to train a SVM model.

For ISVM, we suppose that with some data, we trained a SVM model, and get the support vectors. We record the support vectors from the current model, and with the external data coming in, we train the model again with the whole external data as well as the current support vectors. Since with a small amount of support vectors, the whole image can thus be discribed, then the vectors which are not support vectors can be thrown away. We only care about the support vectors.

However, there are some concerns for the ISVM, since both the newly added vectors and the former support vectors are considered of same weight, the support vectors are more likely to be ignored when the external data has a great population. So we multiple an "L" on the former supporting vectors, making it LSVM.

The following function shows the lose function of LSVM with a Soft Margin.

$$min(w, b, \xi) \quad \frac{1}{2}||w||^2 + C(\Sigma_{i \in I}\xi_i + L \cdot \Sigma_{i \in S}\xi_i)$$

with L

$$L = \frac{\#examples}{\#SVs}$$

By this way, if we split a large data set into small data sets with size $O(1)$, then the total time needed is $O(n)$, but calculating the Kernel Matrix for a data set with size n is already $O(n^2)$, so for large data set, using this method can significantly reduce the time and memory needed for training the model.

We test the method with data set 3, the result is shown below.

```
start iteration number:  10
     pcost       dcost       gap    pres   dres
 0: -1.0781e+03 -2.2256e+03  1e+04  4e+00  2e-14
 1: -7.5547e+02 -1.7479e+03  2e+03  3e-01  2e-14
 2: -8.0091e+02 -1.0464e+03  3e+02  1e-02  1e-14
 3: -8.8732e+02 -9.1484e+02  3e+01  1e-03  2e-14
 4: -8.9941e+02 -9.0563e+02  6e+00  2e-04  2e-14
 5: -9.0171e+02 -9.0381e+02  2e+00  6e-05  2e-14
 6: -9.0254e+02 -9.0313e+02  6e-01  1e-05  2e-14
 7: -9.0281e+02 -9.0289e+02  8e-02  1e-06  2e-14
 8: -9.0285e+02 -9.0286e+02  9e-03  8e-08  2e-14
 9: -9.0285e+02 -9.0285e+02  6e-04  5e-09  2e-14
Optimal solution found.
0.9855555555555555
```

The error rate is even lower than the Soft Margin SVM. See the code in class ISVM.

5 Reference

1. VE445 Lab 1 Manual
2. Wikipedia Page for SVM
   `https://en.m.wikipedia.org/wiki/Support-vector_machine`