Assignment 6

Gaurav Baney, Wei-Han Chang gbaney@usc.edu, weihanc@usc.edu
04/20/2020

1. Implementation

Our implementation of Support Vector Machines is in the file 'svm.py'
The training and test files have been hardcoded to run upon executing the file.
We used the python package cvxopt as the quadratic program solver.

1.1 Results

The results of the linear kernel were:

```
************** Linear **************
     pcost       dcost       gap    pres   dres
 0: -2.0636e+01 -4.3905e+01  3e+02  2e+01  2e+00
 1: -2.2372e+01 -3.7202e+01  9e+01  5e+00  5e-01
 2: -2.3112e+01 -3.8857e+01  5e+01  2e+00  2e-01
 3: -2.8318e+01 -3.3963e+01  1e+01  4e-01  4e-02
 4: -3.2264e+01 -3.3927e+01  2e+00  1e-02  1e-03
 5: -3.3568e+01 -3.3764e+01  2e-01  1e-03  1e-04
 6: -3.3737e+01 -3.3739e+01  2e-03  1e-05  1e-06
 7: -3.3739e+01 -3.3739e+01  2e-05  1e-07  1e-08
 8: -3.3739e+01 -3.3739e+01  2e-07  1e-09  1e-10
Optimal solution found.
W:
[ 7.25005616 -3.86188932]
B:
-0.10698726795886O6
Support vectors
[[0.24979414 0.18230306]
 [0.3917889  0.96675591]
 [0.02066458 0.27003158]]
**************************************
```

The results of the non-linear (radial) kernel were:

```
********** Non-Linear **********
     pcost       dcost       gap    pres   dres
 0: -1.9159e+01 -5.4322e+01  3e+02  1e+01  2e+00
 1: -4.5057e+01 -8.8795e+01  2e+02  6e+00  1e+00
 2: -8.7553e+01 -1.3069e+02  1e+02  5e+00  8e-01
 3: -2.2938e+02 -2.7700e+02  2e+02  4e+00  8e-01
 4: -4.0227e+02 -4.8316e+02  2e+02  3e+00  5e-01
 5: -4.0985e+02 -4.2875e+02  4e+01  5e-01  8e-02
 6: -4.0009e+02 -4.0076e+02  1e+00  1e-02  2e-03
 7: -3.9987e+02 -3.9988e+02  3e-02  2e-04  4e-05
 8: -3.9986e+02 -3.9986e+02  4e-04  2e-06  4e-07
 9: -3.9986e+02 -3.9986e+02  4e-06  2e-08  4e-09
Optimal solution found.
W:
[-183.54580633   22.48790584]
B:
6.640727720055638
Support vectors
[[-10.260969     2.07391791]
 [  1.66404809  12.68562818]
 [  1.3393313  -10.29098822]
 [  9.67917724   4.3759541 ]
 [ -9.53754332  -0.51895777]
 [ -9.46760885   2.36139525]
 [  6.99249663  -6.41143087]
 [  9.90143538  -0.31483149]]
**************************************
```

1.1.1 Implementation

In our implementation we use numpy arrays to improve the speed of the algorithm (SIMD operations) and also for certain math features that can be directly applied to the numpy

arrays(Transpose, dot, summations, etc)
We use the matrix class from the quadratic solver for the appropriate matrix operations involved in the quadratic functions.

1.1.2 Data Structures

We used numpy arrays to vectorize all the point input and they labels they carry
We use a dictionary to store the results of single forward and single backward movement
We save the weights and biases in a separate dictionaries.

1.1.3 Code Level Optimizations

We used the point class from one of our previous assignments to help with debugging and other information storing optimizations

1.2 Software Familiarization

1.2.1 sklearn.svm.SVC

The SkLearn Results were as follows:

```
'Linear'
'Support Vectors'
array([[0.23307747, 0.86884518],
       [0.23918196, 0.81585285],
       [0.14301642, 0.85313079],
       [0.14586533, 0.74931925],
       [0.26419794, 0.91067489],
       [0.06756031, 0.65812372],
       [0.17422964, 0.6157447 ],
       [0.01107579, 0.39873158],
       [0.15267995, 0.8006936 ],
       [0.03436631, 0.50247843],
       [0.3917889 , 0.96675591],
       [0.02066458, 0.27003158],
       [0.55919837, 0.70372314],
       [0.2498981 , 0.15693917],
       [0.65628367, 0.77812372],
       [0.27872572, 0.23552777],
       [0.24979414, 0.18230306],
       [0.70631503, 0.87261758],
       [0.22068726, 0.11139981],
       [0.36354491, 0.25915653],
       [0.42066002, 0.43762265],
       [0.76570056, 0.98727513],
       [0.45552411, 0.49956489],
       [0.6798148 , 0.90468041]])
'Coefficient'
array([[ 3.59965788, -2.03198838]])
'Intercept'
array([0.21848298])
```

```
'Nonlinear'
'Support Vectors'
array([[ -8.47422847,   5.15621613],
       [ -2.7471552 ,  -8.47244873],
       [ -2.52922727,  -8.29282482],
       [  7.38012912,  -1.36077284],
       [ -6.90647562,   7.14833849],
       [ -1.40132717,  -7.89496002],
       [ -9.53754332,  -0.51895777],
       [ -9.46760885,   2.36139525],
       [  6.3666283 ,  -6.49712918],
       [  0.20162846,  -8.81260121],
       [  0.44946335,   8.41292208],
       [  6.99249663,  -6.41143087],
       [ -6.80002274,  -7.02384335],
       [  9.90143538,  -0.31483149],
       [ -4.98349411,   8.31816584],
       [  4.27289989,   8.67079427],
       [ 10.24592717,   7.95373492],
       [-14.23121874,   8.57661163],
       [-15.64719728,   3.32039056],
       [-11.64621294,  -0.87217731],
       [ 12.74780931,   0.19913032],
       [-10.02833317,  11.09354511],
       [  3.28969027, -14.15854536],
       [  2.91722251, -12.27214032],
       [ -6.41766882, -16.57062517],
       [ -1.08933763, -14.10562483],
       [-10.260969  ,   2.07391791],
       [  1.66404809,  12.68562818],
       [  1.3393313 , -10.29098822],
       [ 11.75880948,  -9.85890377],
       [  9.67917724,   4.3759541 ]])
'Intercept'
array([1.13838513])
```

Sklearns Support Vector Machine package also offered a lot more control giving more precision on the kind of fit you want to create with the data. It also has a variety kernel

functions to find a better fit for the separable data. Compared to our approach, sklearn offers more freedom with parameter types and selections.

2. Applications

SVMs are currently being used to segregate datapoints is widely used in the field of biological or chemical science for compound classification and recognition. More info here: https://www.tandfonline.com/doi/abs/10.1517/17460441.2014.866943