Ravisutha Sakrepatna Srinivasamurthy

SVM

A. SVM software used: libSVM

The main reason for choosing libSVM is to get hyperplane (w) values after getting svm vectors. Moreover, it a very easy to use package which can be run on MATLAB. The tools folder in the libSVM package consists of assisting tool software. For this assignment, "**grid.py**" was helpful in getting best "c" and "gamma" values and to increase the accuracy.

The libSVM is an open source machine learning tool and hence in the future if required, I can dig deep into the code and tweak if necessary. It supports various other languages and learning curve for implementing the SVM in other languages will be minimal.

Finally, it provides all necessary API's to classify our dataset. For example, for converting dataset to libSVM format, we can use libsvmwrite. Similarly, we can choose various kernels by passing the appropriate options. Hence, it is an efficient and easy to use SVM tool.

B. Linear model:

a. Support Vector Set

The model.SVs contains the support vectors. It contains 4549 support vectors.

b. Hyperplane parameters

W = [9.745e+04 -1807e+4 7.5592e+02 -1.876e+05]

b = -0.12558

c. Classification Performance: Accuracy of the SVM classifier using linear kernel is found to be 80.56

C. RBF model:

a. Support Vector Set
   Contains **4549 support vectors.**
b. Hyperplane parameters
   W = [22.94 8.6 20.3 -38]
   B= 8.26
c. Classification Performance (c = 32 g = 0.00122)
   Accuracy = **90.3**
d. Comparing the performance with other models
   Accuracy:
      Bayesian classifier: 91.4
      KNNR: 89.5
      Ho-kashyap: 81.7
      SVM: 80.56

Crisp c-means

A. For different values of C

C = 2:

Number of elements in

C1: 11226 C2: 3776

Mean:

52.0052 0.4335 -2.5391 -48.9614

11.0617 -0.7716 -25.4943 13.6792

C = 3:

Number of elements in

C1: 2959 C2: 3412 C3: 8632

Mean:

C1: 49.8691 36.4770 30.2715 -49.8329

C2: 7.6560 0.8181 -25.0719 16.3031

C3: 52.3565 -12.5984 -14.9190 -47.0580

C = 4

C1: 2248 C2: 3247 C3: 7212 C4: 2297

Mean:

48.6423 -37.6798 29.6443 -50.3416

5.0909 -0.0168 -24.9263 16.9725

53.3596 -3.2804 -24.0537 -45.1066

50.0502 48.0527 27.4429 -49.9411

C = 5

C1: 6180 C2: 2369 C3: 2312 C4: 2046 C5: 2093

Mean:

48.9844 -4.7496 -23.0382 -52.7648

48.8295 47.6181 26.8505 -49.7647

60.6936 -0.6311 -25.1788 16.1650

-20.8257 0.4380 -24.9206 5.3677

51.0642 -38.8822 32.5953 -49.7756

Using 2-norm:

C = 2:

Number of elements in

C1: 10089 C2: 4913

Mean:

49.4988 0.5978 0.2847 -53.2093

25.6866 -0.8301 -25.9803 7.9037

C = 3:

Number of elements in

C1: 4331 C2: 7996 C3: 2676

Mean:

24.2509 1.2018 -25.7466 12.3615

48.2019 -14.8481 -10.3431 -52.3328

50.5178 43.1626 25.9562 -49.7485C = 4

C = 4:

C1: 2248 C2: 3247 C3: 7212 C4: 2297

Mean:

48.6423 -37.6798 29.6443 -50.3416

5.0909 -0.0168 -24.9263 16.9725

53.3596 -3.2804 -24.0537 -45.1066

50.0502 48.0527 27.4429 -49.9411

C = 5:

C1: 6180 C2: 2369 C3: 2312 C4: 2046 C5: 2093

Mean:

48.9844 -4.7496 -23.0382 -52.7648

48.8295 47.6181 26.8505 -49.7647

60.6936 -0.6311 -25.1788 16.1650

-20.8257 0.4380 -24.9206 5.3677

51.0642 -38.8822 32.5953 -49.7756

Observations:

A. **Naturally developed clusters**
   As the mean value remains almost the same for both norm 2 and norm1, cluster1 can be thought as natural cluster

B. The means of the original classes considering each class of 5000 data points, are M1=[50.1171 -4.9704 -24.8118 -49.8120], M2=[24.1311 -0.1184 -25.0483 0.2915], M3= [49.7022 5.4015 24.5501 -49.9373].

The obtained means are comparable to the computed mean in **kmeans**.