CSci 5525 Homework 2

Luis Guzman

Thursday October 15, 2020

1. a

$$K(x, x') = \sum_{j=1}^{\infty} w_l K_j(x, x')$$

For K to be a valid kernel, we need the Gram matrx G with $G_{i,j} = K(x_i, x_j)$ to be positive semidefinite for any $x_1, \ldots, x_n \in \mathcal{X}$. I begin by noting that the Gram matrix of this kernel is the weighted sum of the other Gram matrices:

$$G_{i,j} = \sum_{l=1}^{m} w_l K_l(x_i, x_j) = \sum_{l=1}^{m} w_l G_{l,(i,j)}$$

where $G_{l,(i,j)}$ is the (i, j)th element of the gram matrix of the lth kernel. Using the definition of a PSD matrix, we can show that G is PSD.

$$\begin{aligned} u^T G u &= u^T \sum_{l=1}^{m} w_l G_l u \\ &= \sum_{l=1}^{m} w_l (u^T G_l u) \end{aligned}$$

Since all $w_l \geq 0$ and all $u^T G_l u \geq 0$, we know that $u^T G u \geq 0 \, \forall \, u$ and G is a PSD matrix. Thus, $K(x, x')$ is a valid kernel.

b Using the result for problem 1a, with $w_1 = w_2 = 1$ and $m = 2$,

$$K(x, x') = \sum_{j=1}^{2} K_j(x, x') = K_1(x, x') + K_2(x, x')$$

we know that $K(x, x')$ is valid kernel since it is a special case of problem 1a.

2. In this problem, I implement a linear SVM for classifying the "hw2_data_2020.csv" dataset. The dual form SVM problem can be expressed as follows:

$$\max \left[\sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m K(\boldsymbol{x}_n, \boldsymbol{x}_m) \right]$$

$$\text{subject to } 0 \leq a_n \leq C$$

$$\text{and } \sum_{n=1}^{N} a_n t_n = 0$$

This is a quadratic objective function with linear constraints, so we need to use an optimizer to solve it. For this implementation, I will be using cvxopt. We can reformulate the above problem to match the cvxopt quadratic program (qp) function:

$$\min \left[\frac{1}{2}a^T P a - a\right]$$
$$\text{subject to } Ga \preceq h$$
$$\text{and } t^T a = 0$$

By comparing with the cvxopt qp() documentation, we can easily read off the variables we need:

$$P_{i,j} = tK(x_i, x_j)t^T$$
$$q = -\boldsymbol{e}$$
$$G = \begin{bmatrix} -\mathbb{1} \\ \mathbb{1} \end{bmatrix}$$
$$h = \begin{bmatrix} \mathbf{0} \\ C\boldsymbol{e} \end{bmatrix}$$
$$A = t^T$$
$$b = 0$$

Here, $\boldsymbol{e}$ is a $N \times 1$ vector containing all 1's, $\mathbf{0}$ is a $N \times 1$ vector containing all 0's, and $\mathbb{1}$ is the $N \times N$ identity matrix. After defining these parameters, we can run cvxopt's qp function, and it will calculate our optimal solution a^*. Our weight vecotr is then

$$w^* = \sum_{i:a_i^*>0} a_i t_i x_i$$

and we can predict using

$$\hat{y} = \text{sign}(Xw)$$

In order to identify the ideal value for C, I tested the values $\{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1, 10, 100, 1000\}$ and ran 10-fold cross validation with an 80-20 test data split to compute the error rate for each value. The average error rates and standard deviations are reported below.

C	Error rate	St. Dev.
0.0001	0.498	0.04106
0.001	0.5045	0.04865
0.01	0.497	0.03415
0.1	0.491	0.03813
1	0.4805	0.03718
10	0.4995	0.03475
100	0.485	0.02933
1000	0.5055	0.02162

The best value of C was 1, which makes sense because this parameter controls how large of a margin we will use and, subsequently, how well the model generalizes to the testing data. A high value of C will lead to a very narrow margin and the model may not generalize well to testing data. For this reason, a reasonable choice for C shouldn't be too large or too small.

The error rates are very high for this algorithm, but this is to be expected. This dataset is not linearly separable, so using the linear Kernel does not produce satisfactory results. I plotted the data set below.

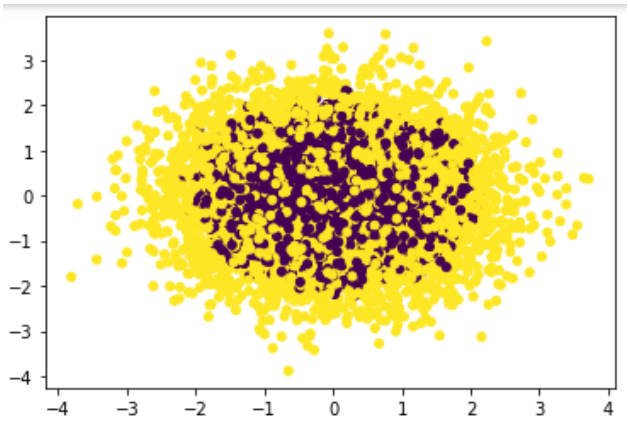

Figure 1: The dataset is not linearly separable

3. In this problem, I use the linear and the Radial Basis Function (RBF) kernel with the dual form SVM to classify the "hw2_data_2020.csv" dataset. The linear kernel is described below and is equivalent to the method in Problem 2:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$$

The RBF Kernel is

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\Big(- \frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\sigma^2} \Big)$$

We can simply plug this into the SVM we created in Problem 2 to get our optimal solution. We can then calculate the bias using equation 7.37 from *Pattern Recognition and Machine Learning*:

$$b = \frac{1}{N_{\mathcal{M}}} \sum_{n \in \mathcal{M}} \Big(t_n - \sum_{m \in \mathcal{S}} a_m t_m K(\boldsymbol{x}_n, \boldsymbol{x}_m) \Big)$$

and we can begin our predictions using

$$\hat{y} = \text{sign}\Big(\sum_{i: a_i^* > 0} t_i a_i K(\boldsymbol{x}_y, \boldsymbol{x}) + b \Big)$$

This problem does include an additional hyperparameter σ, however, so we must implement a grid search to optimize over both C and σ. I again performed 10-fold cross validation with an 80-20 split, and the results are reported below:

Linear Kernel

C	Average	St. Dev.
0.001	0.496	0.03105
0.01	0.4915	0.01845
0.1	0.503	0.01926
1	0.4905	0.0356
10	0.4935	0.04765
100	0.484	0.04067
1000	0.496	0.02998

The results are comparable to Problem 2, since the data is still not linearly separable.

RBF kernel

C/sigma	2.2	10	100	500	750	1000	1250	1500
0.001	0.05	0.455	0.505	0.52375	0.51375	0.50125	0.50625	0.4725
0.01	0.05	0.4225	0.51625	0.50375	0.515	0.5125	0.47875	0.51625
0.1	0.04625	0.29375	0.50875	0.51875	0.47125	0.50625	0.48	0.48875
1	0.055	0.27625	0.505	0.5025	0.5	0.53125	0.4775	0.50625
10	0.03	0.04625	0.48125	0.49	0.53375	0.51375	0.51375	0.53125
100	0.03375	0.03625	0.50625	0.50375	0.5	0.4625	0.5175	0.51125
1000	0.04	0.02625	0.44	0.45	0.50375	0.47375	0.50125	0.4975

The standard deviations of the above data are listed in the appendix. From these results, the best combination of hyper-parameters was C=1000 and sigma = 10; however, for my final SVM, I chose to use C=10 and sigma = 2.2. The validation error rates were within 0.4%, and a more modest C value will generalize better to future testing data.

4. In the final problem, I created a multiclass SVM using the one-vs-all method to classify the mfeat dataset (which is a version of MNIST that includes pre-computed features). I started by formatting the dataset (which comes in six separate files) into a format that can be used by my SVM. I concatenated all features, and generated labels in blocks of 200, as specified by the dataset description. The labels were converted from (0, 9) to {-1, 1} corresponding to whether or not the data point belongs to the class of interest.

   The prediction for the one-vs-all method is a bit different than a binary classifier. Because it's possible for a datapoint to get classified as belonging to multiple labels, we need the raw output of the prediction function to tell "how strongly" the datapoint belongs to each class. The prediction is then the index corresponding to the maximum of those values.

$$\hat{y} = \arg\max \Big(\sum_{i:a_i^*>0} t_i a_i K(\boldsymbol{x}_y, \boldsymbol{x}) + b \Big)$$

   For each kernel, combination of hyper-parameters, and label, I trained an SVM using the method described in problem 2 (10 fold cross validation with 80-20 splits). I chose C = 10 and 100 for this dataset because they are a good balance between over and underfit. I also explored $\sigma = \{2.2, 500, 750, 1000, 1250, 1500\}$. Some of these numbers were chosen because other students had success with them, but judging from the range of the dataset, these should be sufficient. The results are given below:

C	sigma	Average	St. Dev.
10	2.2	0.895	0.02
10	500	0.9075	0.0025
10	750	0.885	0.005
10	1000	0.9075	0.0075
10	1250	0.9	0.005
10	1500	0.885	0.01
100	2.2	0.905	0.02
100	500	0.935	0.0175
100	750	0.8725	0.0025
100	1000	0.9175	0.0125
100	1250	0.9125	0.0025
100	1500	0.8875	0.02

I believe there are still some bugs in my implementation for this question because I wan't seeing error rates significantly better than 90% (random guessing). I verified that my predictions are adequately distributed across all labels, but the accuracy is not great. I spent many hours across multiple days

trying different hyper-parameters and debugging the SVM, but I could not get better error rates than what is listed above.

1 Appendix

C = 0.001 sigma = 2.2
average error: 0.05
standard deviation: 0.01458
C = 0.001 sigma = 10
average error: 0.455
standard deviation: 0.06195
C = 0.001 sigma = 100
average error: 0.505
standard deviation: 0.01871
C = 0.001 sigma = 500
average error: 0.52375
standard deviation: 0.04263
C = 0.001 sigma = 750
average error: 0.51375
standard deviation: 0.03577
C = 0.001 sigma = 1000
average error: 0.50125
standard deviation: 0.02534
C = 0.001 sigma = 1250
average error: 0.50625
standard deviation: 0.04174
C = 0.001 sigma = 1500
average error: 0.4725
standard deviation: 0.03288
C = 0.01 sigma = 2.2
average error: 0.05
standard deviation: 0.0162
C = 0.01 sigma = 10
average error: 0.4225
standard deviation: 0.02194
C = 0.01 sigma = 100
average error: 0.51625
standard deviation: 0.01083
C = 0.01 sigma = 500
average error: 0.50375
standard deviation: 0.03542
C = 0.01 sigma = 750
average error: 0.515
standard deviation: 0.04047
C = 0.01 sigma = 1000
average error: 0.5125
standard deviation: 0.0627
C = 0.01 sigma = 1250
average error: 0.47875
standard deviation: 0.03879
C = 0.01 sigma = 1500
average error: 0.51625
standard deviation: 0.00545
C = 0.1 sigma = 2.2
average error: 0.04625
standard deviation: 0.01431
C = 0.1 sigma = 10
average error: 0.29375
standard deviation: 0.06397
C = 0.1 sigma = 100
average error: 0.50875
standard deviation: 0.04292
C = 0.1 sigma = 500
average error: 0.51875
standard deviation: 0.01746
C = 0.1 sigma = 750
average error: 0.47125
standard deviation: 0.04491
C = 0.1 sigma = 1000
average error: 0.50625
standard deviation: 0.01192
C = 0.1 sigma = 1250
average error: 0.48
standard deviation: 0.0326
C = 0.1 sigma = 1500
average error: 0.48875
standard deviation: 0.02274
C = 1 sigma = 2.2
average error: 0.055
standard deviation: 0.0
C = 1 sigma = 10
average error: 0.27625
standard deviation: 0.07171
C = 1 sigma = 100
average error: 0.505
standard deviation: 0.02894
C = 1 sigma = 500
average error: 0.5025
standard deviation: 0.01146
C = 1 sigma = 750
average error: 0.5
standard deviation: 0.02318
C = 1 sigma = 1000
average error: 0.53125
standard deviation: 0.02655
C = 1 sigma = 1250
average error: 0.4775
standard deviation: 0.02385
C = 1 sigma = 1500
average error: 0.50625
standard deviation: 0.01139
C = 10 sigma = 2.2
average error: 0.03
standard deviation: 0.00612
C = 10 sigma = 10
average error: 0.04625
standard deviation: 0.01139
C = 10 sigma = 100
average error: 0.48125
standard deviation: 0.03798
C = 10 sigma = 500
average error: 0.49
standard deviation: 0.02761
C = 10 sigma = 750
average error: 0.53375
standard deviation: 0.0198
C = 10 sigma = 1000
average error: 0.51375
standard deviation: 0.02219
C = 10 sigma = 1250
average error: 0.51375
standard deviation: 0.01883
C = 10 sigma = 1500
average error: 0.53125
standard deviation: 0.01816
C = 100 sigma = 2.2
average error: 0.03375
standard deviation: 0.00545
C = 100 sigma = 10
average error: 0.03625
standard deviation: 0.0065
C = 100 sigma = 100
average error: 0.50625
standard deviation: 0.02534
C = 100 sigma = 500
average error: 0.50375
standard deviation: 0.04435
C = 100 sigma = 750
average error: 0.5
standard deviation: 0.05906
C = 100 sigma = 1000
average error: 0.4625
standard deviation: 0.03631
C = 100 sigma = 1250
average error: 0.5175
standard deviation: 0.0175
C = 100 sigma = 1500
average error: 0.51125
standard deviation: 0.04204
C = 1000 sigma = 2.2
average error: 0.04
standard deviation: 0.00707
C = 1000 sigma = 10
average error: 0.02625
standard deviation: 0.0074
C = 1000 sigma = 100
average error: 0.44
standard deviation: 0.01541
C = 1000 sigma = 500
average error: 0.45
standard deviation: 0.01458

C = 1000 sigma = 750
average error: 0.50375
standard deviation: 0.0484
C = 1000 sigma = 1000
average error: 0.47375
standard deviation: 0.05561
C = 1000 sigma = 1250
average error: 0.50125
standard deviation: 0.04144
C = 1000 sigma = 1500
average error: 0.4975
standard deviation: 0.05728
Best error: C = 1000 sigma = 10