Tested running on flip1 server

IA3 - Group 26

Cheng Zhen
Bharath Padmaraju
Bharghav Srikhakollu

Sentiment Analysis with support vector machines

Part - 0 : Preprocessing

(a) CountVectorizer

Question: Please report the ten most frequent words (frequency is measured by the total counts of each word are maximal) for the positive tweet and negative tweet respectively. Do you find these words to be informative of the classes?

→ Ten most frequent words for the positive tweet

Most frequent word - Positive	**Frequency of word count**
the	746
you	716
to	716
for	504
thanks	462
jetblue	455
southwestair	452
united	414
thank	355
and	330

→ Ten most frequent words for the negative tweet

Most frequent word - Negative	Frequency of word count
to	4741
the	3236
flight	2313
united	2254
on	2202
and	2191
you	2130
for	2104
my	1926
usairways	1885

Answer:

→ Yes, the most frequent words do seem to be indicative of the class. In the positive class we can see that the words thank, and thanks are present, but omitted in the negative class. Since CountVectorizer can only be helpful in understanding the type of text by the frequency of the words in it, it is not fully informative in identifying more important and less important words for analysis. This is because it considers the words that are abundant in a corpus.

(b)TfidfVectorizer

Question: Please report the ten words with the highest total TF-IDF's for the positive tweet and negative tweet respectively. How do they compare to the list in 0(a)? Which one do you think is more informative and why?

→ Ten most frequent words for the positive tweet

Most frequent word - Positive	Frequency of word count
you	110.87825510124587
thanks	91.69079325308267
thank	86.89812339864528
the	81.97299862208087
jetblue	81.19395636146685
united	73.53405657485403
to	73.41805852212777
southwestair	72.43287660004934
for	64.76012443299349
americanair	60.99872365090127

→ Ten most frequent words for the negative tweet

Most frequent word - Negative	Frequency of word count
to	359.2906838073978
the	295.37105839383537
flight	241.58371560220323
you	238.11512888455562
united	237.01120827396764
on	229.74896490472437
and	222.17548084678558

for	221.41670018189913
usairways	214.98042505497082
my	214.8651738529854

Answer:
→ Yes, as in the case of the count vectorizer we can see that “thank”, and “thanks” are present in the positive class, whereas the negative class just has filler words. TF-IDF showed better results with words like “jetblue”, “southwestair” showing the importance of words rather than showing the results based on the count.
→ TF-IDF is better than Countvectorizer because it not only focuses on the frequency of words present in the corpus but also provides the importance of the words. This helps in removing the least important words for analysis and it would help in model less complex.

Part - 1 : Linear SVM

Question: Use sklearn to train linear SVMs and tune hyperparameter c. You should start with c = 10^i with i = −4,−3, · · · , 3, 4. With the results of these experiments, you should then expand your search to include other possible values of c. The basic principle is that if the best c value is found on the boundary of the search range, you want to further expand that particular boundary. If it is the middle, you should refine the search grid to look further in the neighborhood of the current best. You should search for additional values of c with the goal of optimizing the validation performance.

1. What is the best validation performance you are able to achieve with linear SVM? What c value is used?
Answer:

The best validation performance we were able to achieve with linear SVM is "**0.926**". Achieved this result for c value of "**10^0**" (equal to 1 where i = 0)

Since we got the best performance when i = 0 which is actually in the middle of the given range of i [-4, -3, . . . 3, 4]. So, we tried to refine the search grid to look further in the neighborhood of the current best. Hence experimented with in the range of [-1, 1] and observed the below.
Tested for the values of i → -1, -0.9, -0.8, . . . 0.7, 0.8, 0.9, 1. Just reporting the performance which showed close/better results to the above mentioned best performance at i = 0.

'i'value	**Validation Performance**
-0.4	0.9156000
-0.3	0.9180000
-0.2	0.9224000
-0.1	0.9256000
0	0.9260000
0.1	0.9264000
0.2	0.9260000
0.3	0.9252000
0.4	0.9264000

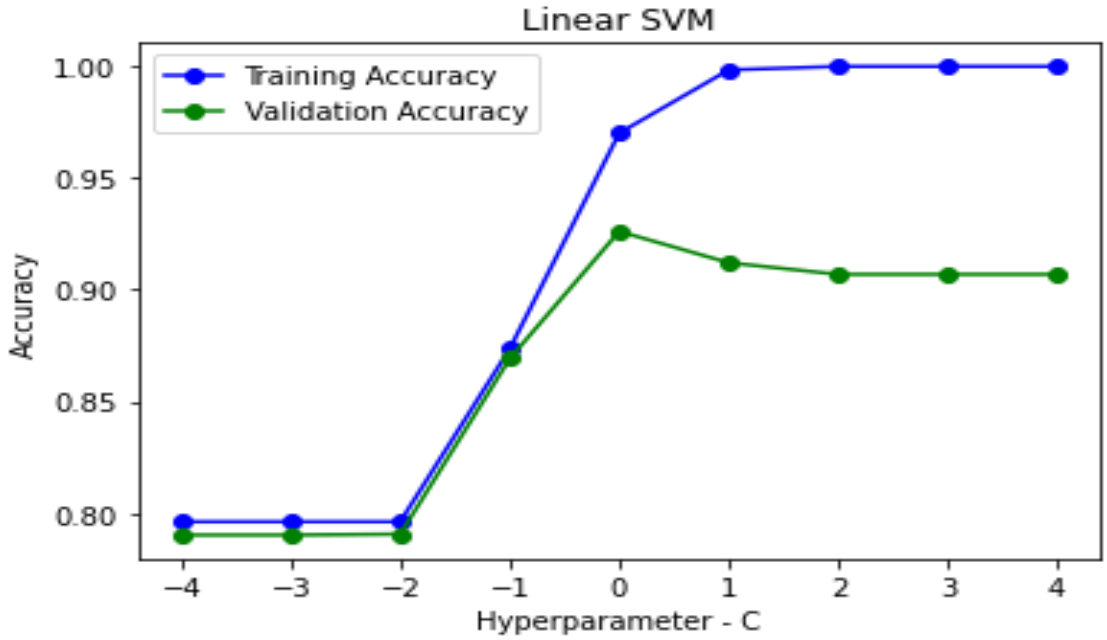
From the above results, we can say that when we improved the search grid with closer "c" values (i values). We found that the best validation performance is "**0.9264**" for c value of **10^0.1** (where i = 0.1).

2. What trend do you theoretically expect to see for training and validation performance as we increase c? Plot the training and validation accuracy against different values of c. Does the trend you observe match your expectation?

Answer:

→ Theoretically, as we increase c (higher), the performance increases because with larger c value the SVM tries to minimize the number of misclassified examples due to the high penalty. This inturn results in a decision boundary with a smaller margin. Similarly, we can say that when the c is small, the penalty for misclassified points is low so a decision boundary with a large margin is selected.

→ Training vs Validation accuracy against different values of c



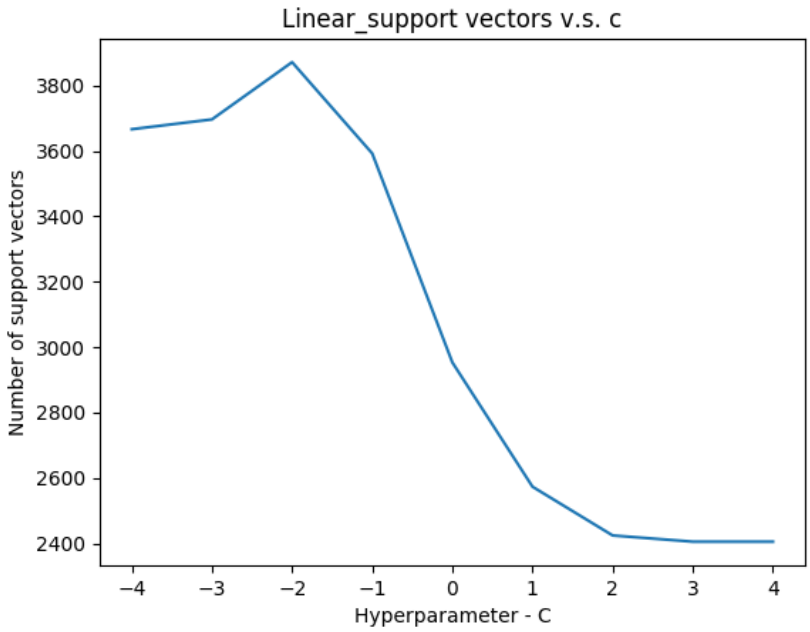
→ The trend matches our expectations. From the above plot, we can observe that although there is a slight dip in the performance inbetween for validation, for almost all other values with the increment in c value, the performance increases for both the training and validation. After a certain c value, the accuracy almost tends to stagnate for both training and validation.

3. What relationship do we theoretically expect between c and the number of support vectors? Plot the number of support vectors against the different values of c. Does the trend you see match your theoretical expectations?

Answer:

→ Theoretically, we expect a smaller number of support vectors as the c increases. This is because a larger regularization parameter c leads to a smaller margin to avoid the penalty from misclassification [1]. Therefore, the number of support vectors gets smaller from the smaller margin.

→ Number of support vectors against the different values of c.



→ The experimental results generally match our theoretical expectations. From the above plot, we can conclude that when the hyperparameter c is increased, there is a decrease in the number of support vectors.

Part - 2 : Quadratic SVM

Question: For this part, you will use sklearn to train SVMs with quadratic kernels and tune hyperparameter c following the same protocol as outlined in part 1.

1. What is the best validation performance you are able to achieve with quadratic SVM? What c value is used?

Answer:

The best validation performance we were able to achieve with linear SVM is "**0.844**". Achieved this result for c value of "**10^1**" i.e. 10^i (where i = 1) and also the same validation performance we observed for the "i" values 2, 3, 4 as well.

So the best validation performance of 0.844 is achieved for 10^1, 10^2, 10^3, 10^4.

Since we got the best performance when i = 1 till 4 which is actually in the middle of the given range of i [-4, -3, . . . 3, 4] and as well at the boundary of the range. So, we tried to refine the search grid to look further in the neighborhood of the current best and also tried to expand the boundary of the range. Hence experimented with in the range of [0, 10] and observed the below.

Tested for the values of i → 0, 0.1, 0.2, . . . 1, 2, 3, . . . 9, 10.

Just reporting the performance which showed close/better results to the above mentioned best performance.

'i'value	**Validation Performance**
0.8	0.8440000
0.9	0.8440000
1	0.8440000

2	0.8440000
3	0.8440000
4	0.8440000
5	0.8440000
6	0.8440000
7	0.8440000

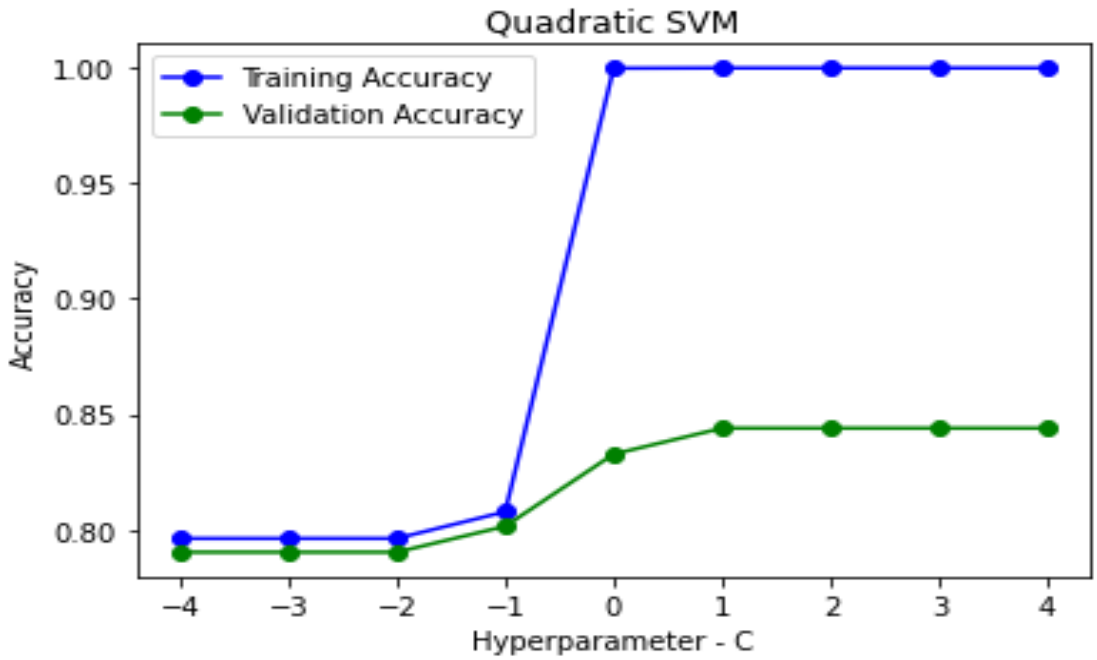
From the above results, we can say that when we improved the search grid and as well as expanded the boundary range with different "i" values. We found that the best validation performance is "**0.844**" for c value 10^i (where i is [0.2, 10]).

2. What trend do you theoretically expect to see for training and validation performance as we increase c? Plot the training and validation accuracy against different values of c. Does the trend you observe match your expectation?

Answer:

→ Similar to Linear SVM, theoretically, as we increase c (higher), the performance increases because with larger c value the SVM tries to minimize the number of misclassified examples due to the high penalty. This inturn results in a decision boundary with a smaller margin. Similarly, we can say that when the c is small, the penalty for misclassified points is low so a decision boundary with a large margin is selected. In addition to the hyperparameter it will have an impact of gamma for a polynomial kernel (degree 2) which can be considered as a fixed value with varying value of c in the model.

→ Training vs Validation accuracy against different values of c

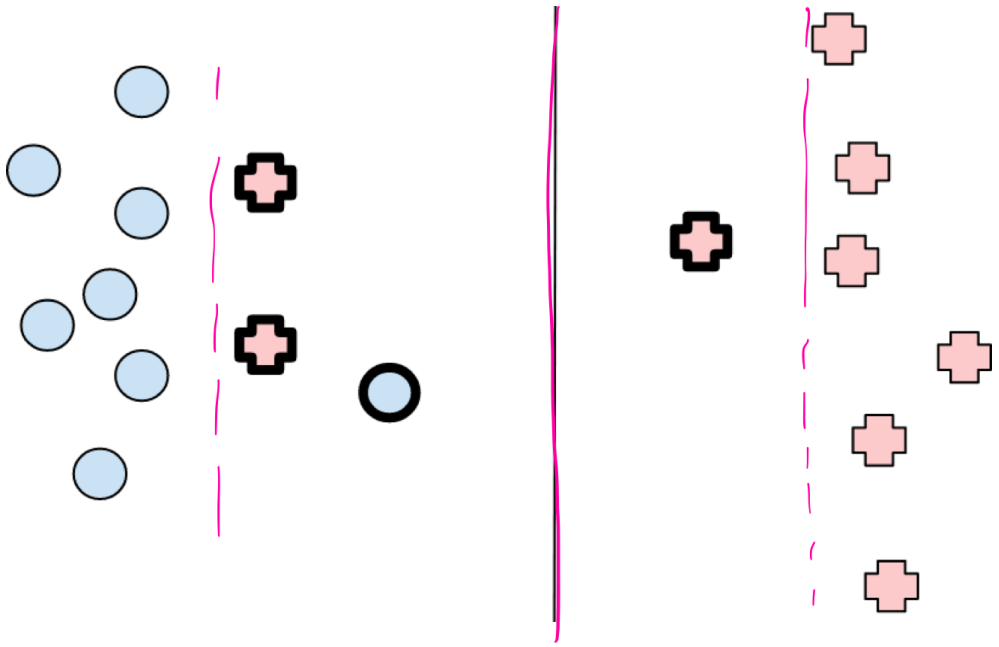


→ The trend matches our expectations. From the above plot, we can observe that, for all the values with the increment in c value, the performance increases for both the training and validation after a certain c value, the accuracy stagnates for both training and validation.

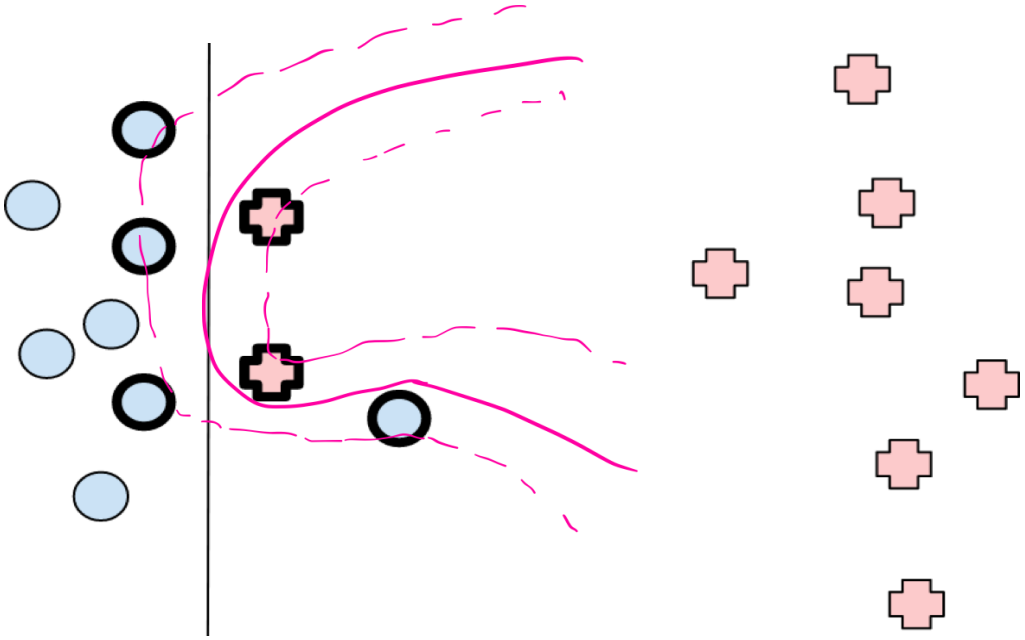
3. What relationship do we theoretically expect between c and the number of support vectors? Plot the number of support vectors against the different values of c. Does the trend you see match your theoretical expectations?

Answer:

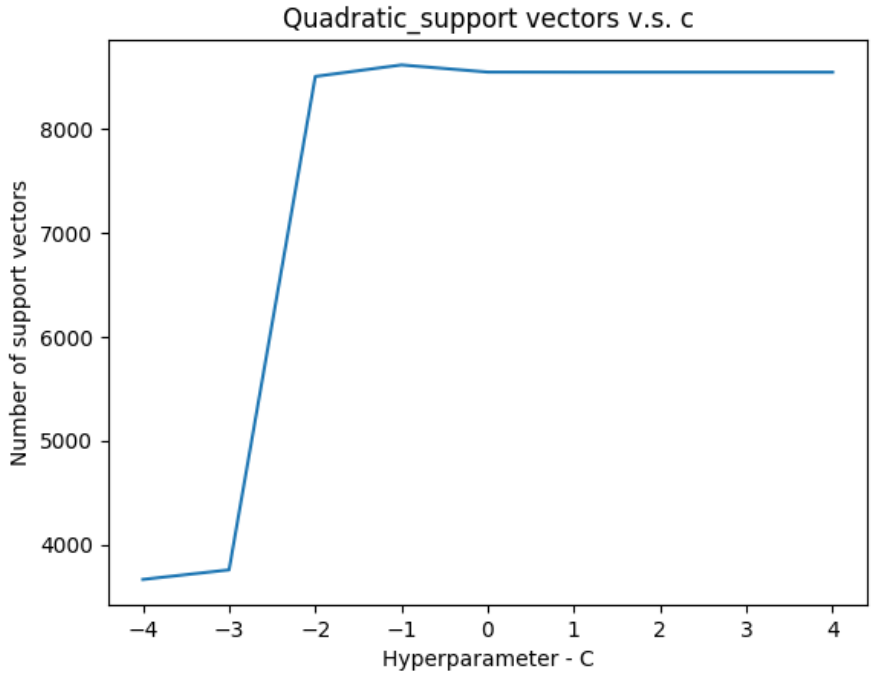
→ Theoretically, we expect a larger number of support vectors as the c increases.This is because a larger regularization parameter c leads to a more curved classifier to avoid the penalty from misclassification. Referring to the figures from Dr. Fern below, with a small c, the classifier is relatively straight and simple because it does not have to avoid many misclassifications: the penalty for misclassification is small.



If the c is high as the figure below, we have to learn a curved classifier to avoid as many misclassifications as possible because of the huge misclassification penalty. This curved and tricky classifier will include more data points around, which leads to more support vectors. As the c keeps going up, the number of support vectors will not increase much because there is no more room to avoid misclassification. The trend is also dependent on another parameter coef0 as the TA instructed.



→ Number of support vectors against the different values of c.



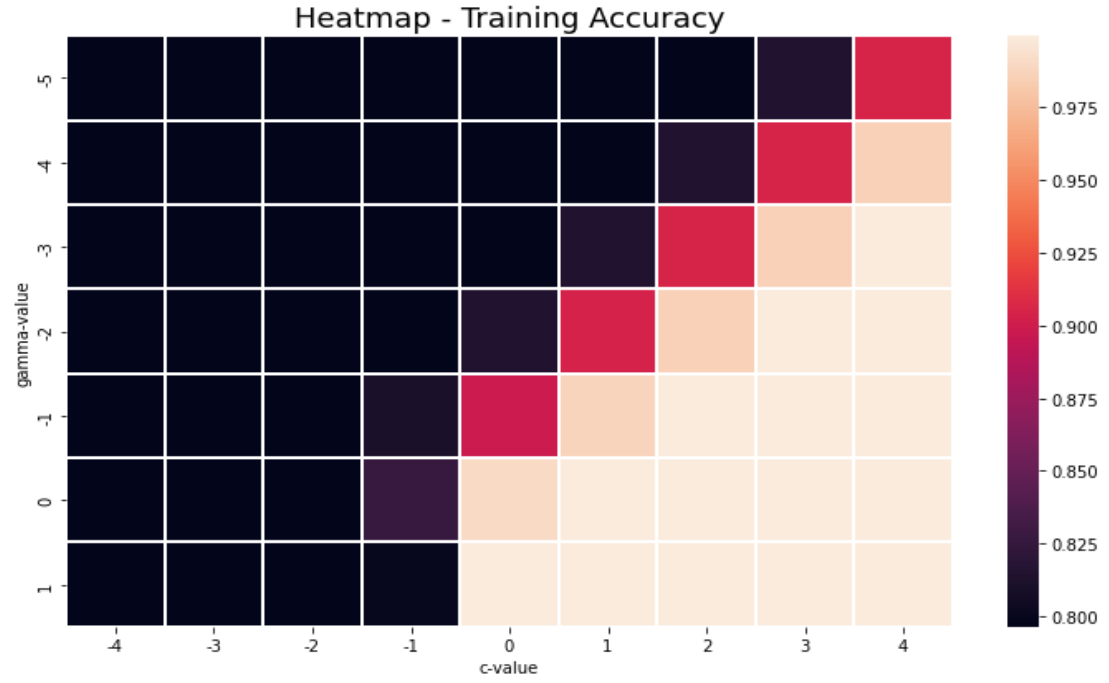
→ The experimental results match the trend in theory with little impact because of the parameter coef0.

Part - 3 : SVM with RBF kernel

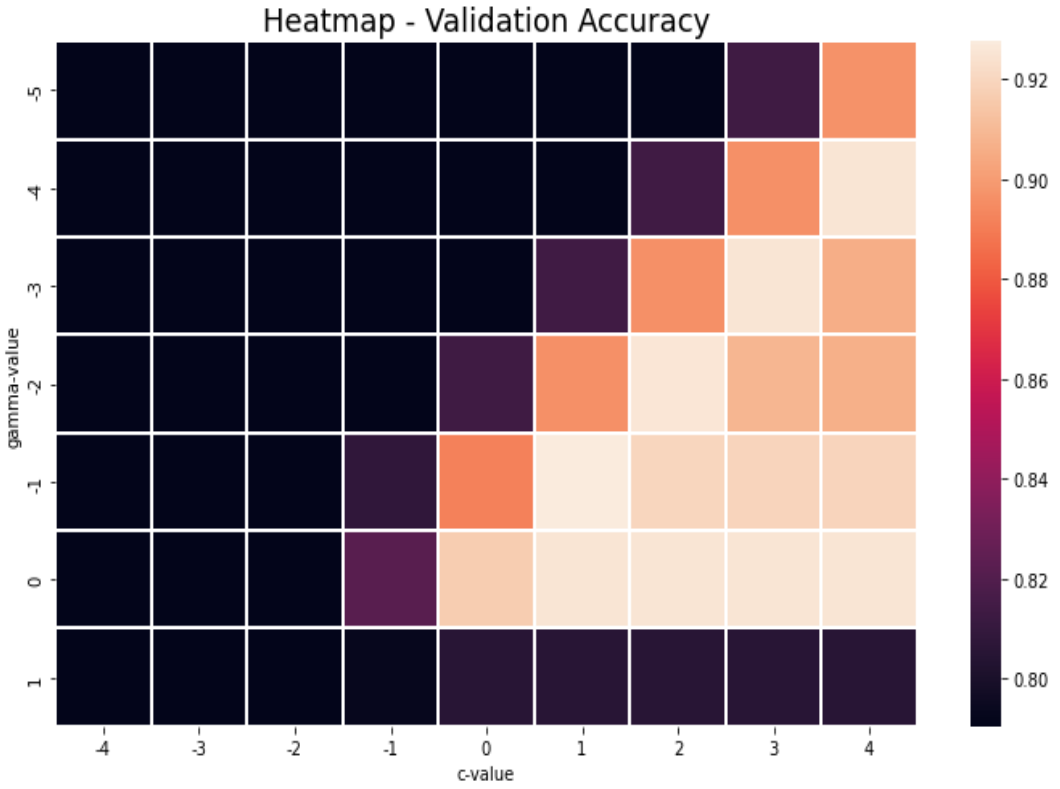
Question: Use sklearn to train RBF kernel SVMs and tune hyperparameters c and γ (γ = 1/σ2 in the RBF kernel definition provided in the lecture slides). For c you should start with the same range as described above. For γ, you should start with the following range: 10^i with i = −5,−4, · · · , 1. Use the heatmap function from seaborn https://seaborn.pydata.org/generated/seaborn.heatmap.html to plot the training accuracy and validation accuracy (separately in two different heatmaps) as a function of c and γ. Please expand your search beyond the provided values to optimize validation performance but your heatmap doesn't have to include these additional values beyond the specified range.

Answer:

→ Training Accuracy as a function of c and γ



→ Validation accuracy as a function of c and γ



1. What is the best validation performance you are able to achieve for the RBF kernel? What c and γ parameters are used?

Answer:

The best validation performance we were able to achieve for the RBF kernel is "**0.9276**" for c value of **10^1** (where i = 1) and γ value of **-1**

Now with c as 10^1 and γ as -1 we tried to experiment further by accessing the neighborhood values of c and γ combinations.
For different combinations of c and γ, we found some optimized validation performance as below.
Experimented with the grid of c = [0.6, 0.7, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4] and γ = [-1.5, -1.4, -1.3, -1.2, -1.1, -1, -0.9, -0.8, -0.7, -0.6, -0.5]
Just mentioning the some of the values which gave better validation performance than "0.9276"

'i' value	'γ' value	Validation Performance
0.6	-0.6	0.9292000
0.7	-0.8	0.9292000
0.7	-0.6	0.9292000
0.7	-0.5	0.9304000
0.8	-0.9	0.9292000
0.8	-0.6	0.9300000
0.8	-0.5	0.9292000
1	-1	0.9276

From the above observations, we can say that when we refined the search grid further, we can see some optimized validation performances.

Ex: 0.9304000 for c value of 0.7 and γ of -0.5
0.9300000 for c value of 0.8 and γ of -0.6

2. What trend do you theoretically expect to see for training and validation performance as we increase c with fixed γ? Does the trend you observe match your expectation?

Answer:

→ For a fixed γ and a varying c value from low to high values we can expect the following. When the c value is small, the classifier is tolerant of misclassified data points. With increment in c value we are expected to see the classifier less tolerant to misclassified data points and the decision boundary is more severe. With a very huge value of c, the classifier finds it very difficult to not misclassify data points and we might see the signs of overfitting.

→ We believe the trend we observed matches our expectations. From the plots, we can observe that for a fixed γ, when the c value increments we can observe the increase in the accuracies but when we tend to take very huge values of c there are chances of overfitting the model.

3. What trend do you theoretically expect to see for training and validation performance as we decrease γ with fixed c? Does the trend you observe match your expectation?

Answer:

→ For a fixed value c and a varying γ value from low to high values we can expect the following. With a very high value of γ, the decision boundary will be completely dependent on the individual data points, which leads to overfitting. With decrement in γ value we are expected to see the decision boundary to cover the data and it is affected by the individual data points. When the γ value is small, the decision boundary will look like an arch but not very curved shape.

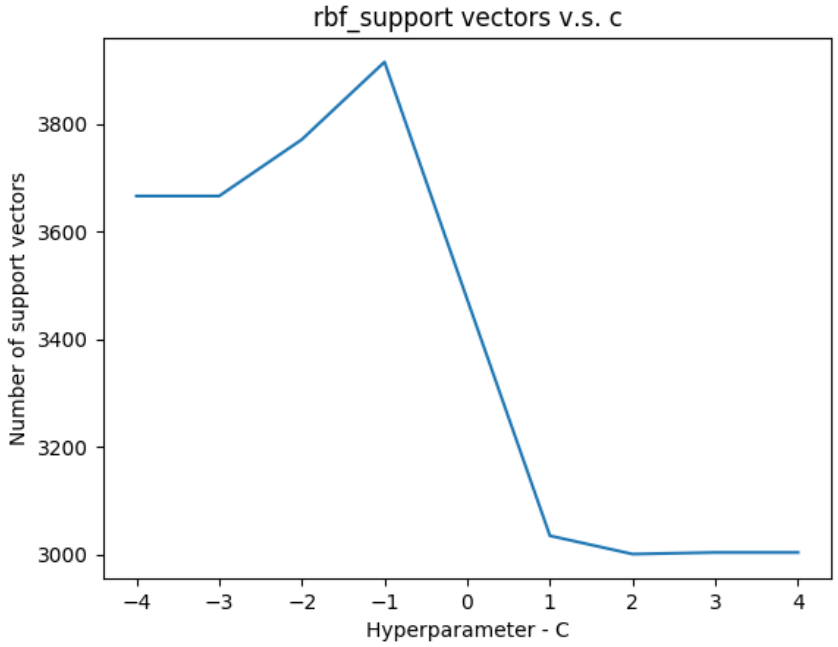
→ We believe the trend we observed matches our expectations. From the plots, we can observe that for a fixed c, when the γ value decreases we can observe the increase in the accuracy. When the γ value is very high, it shows signs of overfitting.

4. With fixed γ What relationship do we theoretically expect between c and the number of support vectors? Plot the number of support vectors as a function of c value for γ = 0.1. Does the trend you see match your theoretical expectations?

Answer:

→ Generally, higher c leads to a smaller number of support vectors. A high c pushes the classifier to eliminate misclassification as much as possible to avoid high penalty of misclassification. As a result, the margin becomes narrow, and the number of support vectors becomes small.

→ Number of support vectors as a function of c value for γ = 0.1.



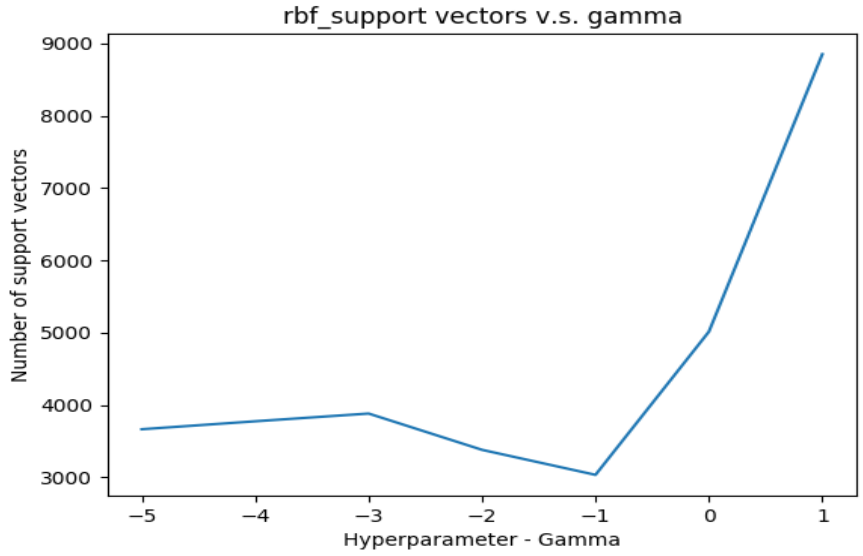
→ The experimental results match the theoretical trend, except for a small fluctuation at the beginning. The general trend still holds.

5. With fixed c, what relationship do we theoretically expect between γ and the number of support vectors? Plot the number of support vectors as a function of γ for c = 10. Does the trend you see match your theoretical expectations?

Answer:

→ Generally, the higher the gamma, the larger the number of support vectors. Based on an introduction article from sklearn [1], a high gamma means the influence of each data point only applies to the space geometrically nearby. This further suggests that each data point can only affect the classifier very locally. Therefore, more points have a chance to decide the decision boundary. In other words, every small piece of the decision boundary is decided by the data point very nearby. In total, the whole decision boundary depends on a great number of points, i.e. a large number of support vectors. As a comparison, low gamma leads to a more broad influence so that we don't need many points to decide a classifier.

→ Number of support vectors as a function of γ value for c = 10.



→ The experimental results match the theoretical trend. The observation can be found from the above plot.

Part - 4 : Final Discussion

Question: Comparing across the three different models and their performances, please discuss, for each model, the main sources for their error. Specifically, do you think for each case, the error is primarily the modeling, estimation, optimization or Bayes error? Or a combination of multiple types of errors? Please provide a clear rationale for your claims.

Answer:

→ Linear SVM: estimation error, bayes error
Rational: as the validation accuracy is much lower than the training accuracy for many c values, we can tell an overfitting from it — estimation error. Bayes error exists more or less in real-world data sets. No significant evidence for modeling error because neither the quadratic SVM or RBF SVM gave a higher validation accuracy. No evidence of optimization error because: 1) sklearn is a mature framework in terms of its optimization module 2) the training accuracies for many c values are very high: the algorithm seems fine optimizing the loss to a small value, i.e. convergence.

→ Quadratic SVM: estimation error, bayes error
Rational: as the validation accuracy is much lower than the training accuracy for many c values, we can tell an overfitting from it — estimation error. Bayes error exists more or less in real-world data sets. No significant evidence for modeling error because the training accuracy is almost 100%. This means the model is not simple, i.e. no underfitting. No evidence of optimization error because: 1) sklearn is a

mature framework in terms of its optimization module 2) the training accuracies for many c values are very high: the algorithm seems fine optimizing the loss to a small value, i.e. convergence.

→ RBF SVM: bayes error
Rational: when gamma = 1, and c =1, 2, 3, or 4, both the training and the validation accuracy were almost 100%. This indicates no substantial estimation error. Bayes error exists more or less in real-world data sets. No significant evidence for modeling error because the training accuracy is almost 100% for some gamma and c. This means the model is not simple, i.e. no underfitting. No evidence of optimization error because: 1) sklearn is a mature framework in terms of its optimization module 2) the training accuracies for many c values are very high: the algorithm seems fine optimizing the loss to a small value, i.e. convergence.

Reference:
[1] An introduction article from sklearn:
https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html