Assignment 3

Gandhar Vaidya

October 18, 2015

Contents

1 Introduction 1

2 PART 1: SVM with no bias term 1

3 PART 3: Soft-margin SVM for separable data 2

4 PART 3: Soft-margin SVM for separable data 2

5 PART 4: SVM accuracy on a grid of parameter values for Gaussian and polynomial kernels 2

5.1 Accuracy as a function of parameters and discussion of the results 3

5.2 Comparison of normalized and non-normalized results 8

5.3 Visualization of the kernel matrix and observations made about it 8

6 References 12

1 Introduction

This assignment report explores the kernelized and non-kernelized soft-margin SVM classifiers on basis of classifier accuracy. It also explores the effect of normalization on accuracy. More, it emphasizes the need for parameter selection process to get better accuracy using kernelized SVM classifiers.

2 PART 1: SVM with no bias term

For given f(x) = $w^T x$, the KKT conditions of the svm and the lagrangian with no bias term are given in equations 1 & 2 respectively. The saddle point equations are derived by taking partial derivatives of the lagrangian with respect to w and ξ_i. These equations are given in equation 3. By plugging in the value of w got from the saddle point equations into the lagrangian, the svm dual is derived which is shown in a set of equations under a sub-heading "Derivation of the svm dual".

The soft margin svm primal:

$$\underset{(w,b)}{\text{minimize}} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i, \qquad \text{subject to: } y_i(w^{tr}\mathbf{x}_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \ i = 1, \ldots, n$$

KKT conditions:

$$\bigtriangledown\Lambda(w,\alpha,\xi) = 0; \qquad \alpha_i \geq 0; \qquad \xi_i \geq 0; \qquad g(x) = (1 - \xi_i - y_i w^T\mathbf{x}_i) \leq 0; \qquad \alpha_i g(x) = 0 \tag{1}$$

The Lagrangian for the soft margin svm:

$$\Lambda(w,\alpha,\xi) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i[1 - \xi_i - y_i w^T\mathbf{x}_i] + \sum_{i=1}^{n}\beta_i\xi_i \tag{2}$$

Saddle point equations:

$$\frac{\partial \Lambda}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \Longrightarrow w = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i; \qquad \frac{\partial \Lambda}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \tag{3}$$

Derivation of the svm dual:

By plugging in the value of w got from the saddle point equations into the lagrangian, the svm dual is derived which is shown in here. Since, just the bias term is not present, the derivation process remain the same and only the constraints change. Therefore, the derivation process is referred from Dr Ben-Hur's lecture notes and it's not included in this report in detail step by step to avoid redundancy.

The svm dual:

$$\underset{\alpha}{\text{maximize}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^t r \mathbf{x}_j \qquad \text{subject to:} \quad 0 \leq \alpha_i \leq C \tag{4}$$

Implications on the design of SMO-like algorithms:

SMO algorithms work by iteratively optimizing two variables at a time. The main reason behind is that doing two variables at a time makes sure that the $\sum_{i=1}^{n} \alpha_i y_i = 0$ constraint of the dual (for decision function with the bias term) is met in each iteration. But, in this case of the decision function without the bias term, there is no such constraint. Therefore, SMO like algorithms now can choose even one variable to optimize at a time, which further simplifies the problem by making it easier to solve analytically. It also relaxes the need for "shrinking" while the optimization process is going on.

3 PART 3: Soft-margin SVM for separable data

It is ***false*** that for linearly separable data, at the optimal solution to the primal problem, all the training examples will have xi equal to zero. How many examples will have non-zero slack variables and the value of slack variables for such examples depend on the value of soft margin parameter C.

Also, soft-margin SVM is advantageous to use even for linearly separable data, because the value of the soft margin parameter C for soft margin SVM classifier set a degree of freedom to choose the tolerance level of noise. This way, all the examples which are classified within the margin with positive slack variables for a given C can be treated as very noisy or odd.

4 PART 3: Soft-margin SVM for separable data

It is ***false*** that for linearly separable data, at the optimal solution to the primal problem, all the training examples will have ξ_i equal to zero. How many examples will have non-zero slack variables and the value of slack variables for such examples depend on the value of soft margin parameter C.

Also, soft-margin SVM is advantageous to use even for linearly separable data, because the value of the soft margin parameter C for soft margin SVM classifier set a degree of freedom to choose the tolerance level of noise. This way, all the examples which are classified within the margin with positive slack variables for a given C can be treated as very noisy or odd.

5 PART 4: SVM accuracy on a grid of parameter values for Gaussian and polynomial kernels

The "motifs" dataset used in this part is very sparse. Therefore, the following script was used to input the data into python:

```
# Function for implementing kernel matrix
f_in = open('scop_new.data')
f_out = open('new_file', 'w')
for line in f_in:
    new_str = ''.join(line.split(',')[1:])
    f_out.write(new_str)
print f_out
from sklearn.datasets import load_svmlight_file
X, y = load_svmlight_file("new_file")
```

After importing the dataset into python, the accuracy of the SVM classifier as a function of soft margin parameters and kernel parameters were studied for normalized as well as non-normalized versions of the data using both the gaussian as well as the polynomial kenels.

5.1 Accuracy as a function of parameters and discussion of the results

After importing the dataset into python, the dataset was spilt into training and testing sub-parts using split() function over the data object. The same testing and training sub-parts were used through out the study of accuracy.

Th same split ratio of 0.4 was used for normalized as well as non-normalized data sets. In fact, to generate the results of normalized data, the same training and testing sub-parts were normalized and used after they were used for the non-normalized data.

The plots for accuracy (balanced success rate) of gaussian kernel and polynomial kernel based soft margin SVM classifier as a function of soft margin parameter of SVM and free parameter of the kernel function were created by using matplotlib.pyplot.pcolormesh() method. The same method was used for the normalized data as well.

The procedure/code used to create these plots is given in Appendix I.

Discussion:

The accuracy on a grid of parameter values for Gaussian kernel is given in Figure 1. The free parameter (gamma) values ranges from 0.00001 to 100,000 and the C ranges from 0.00001 to 100,000 for this plots. Figure 2 shows similar grid for polynomial kernel. The free parameter (degree) values ranges from 1 to 10 and the C ranges from 100,000 to 0.00001 for these plots.

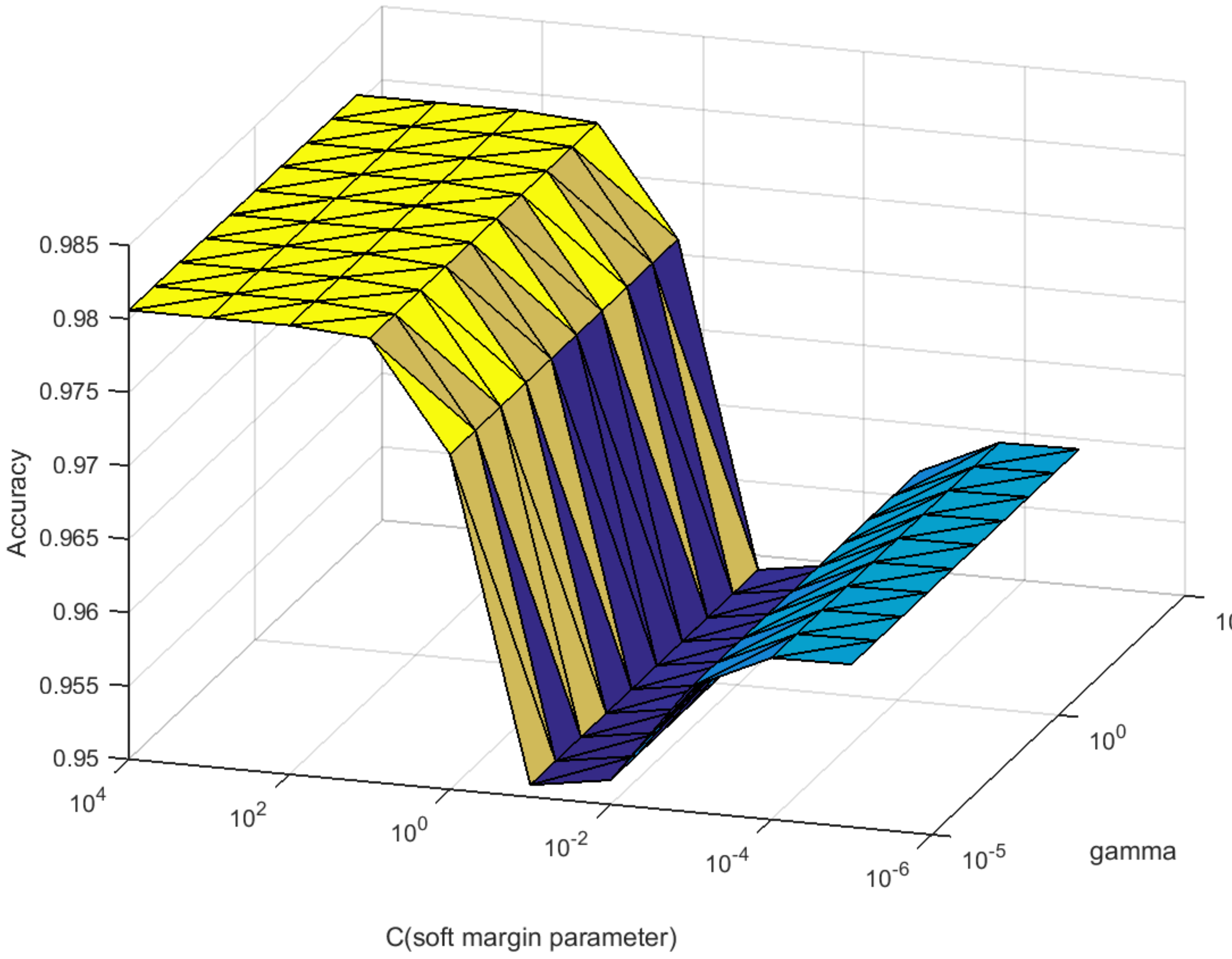


Figure 1: gamma vs C vs Accuracy

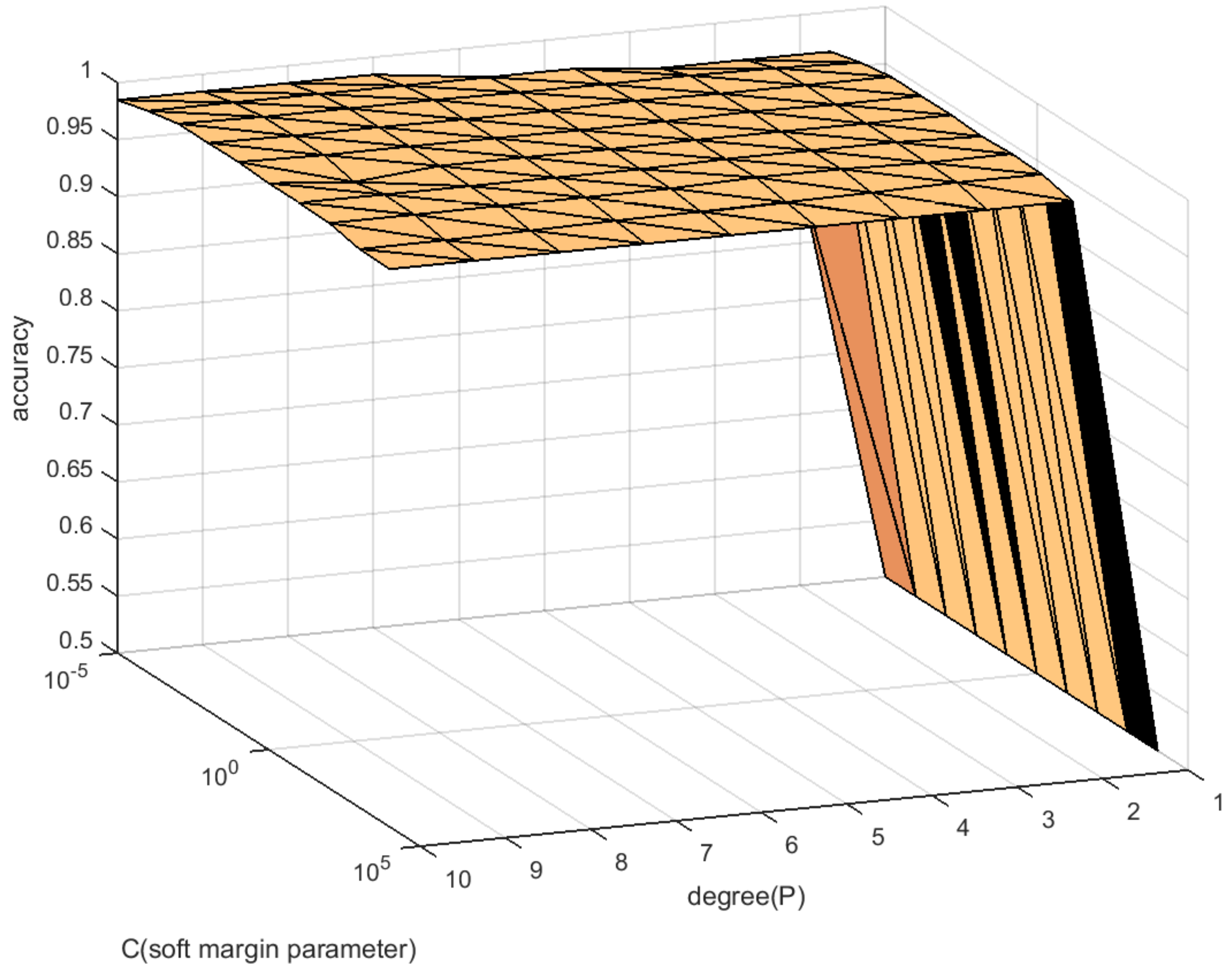


Figure 2: degree vs C vs Accuracy

Figure 3 and Figure 4 gives accuracy values for all the values of soft margin parameter C for two (small and large) specific values of gamma of gaussian kernel and degree of polynomial kernal respectively. Accuracy values for all the values of free parameter for two (small and large) specific values of C are given in Figure 5 and Figure 6.

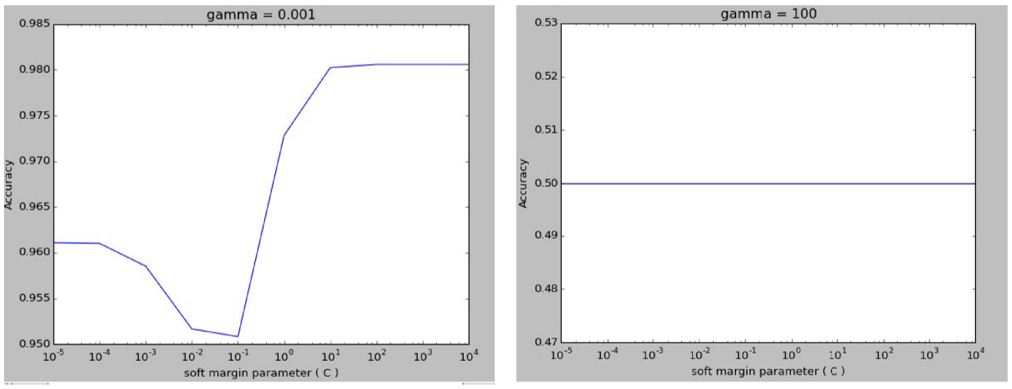


Figure 3: specific gamma values using gauss kernel and varying C on a log scale

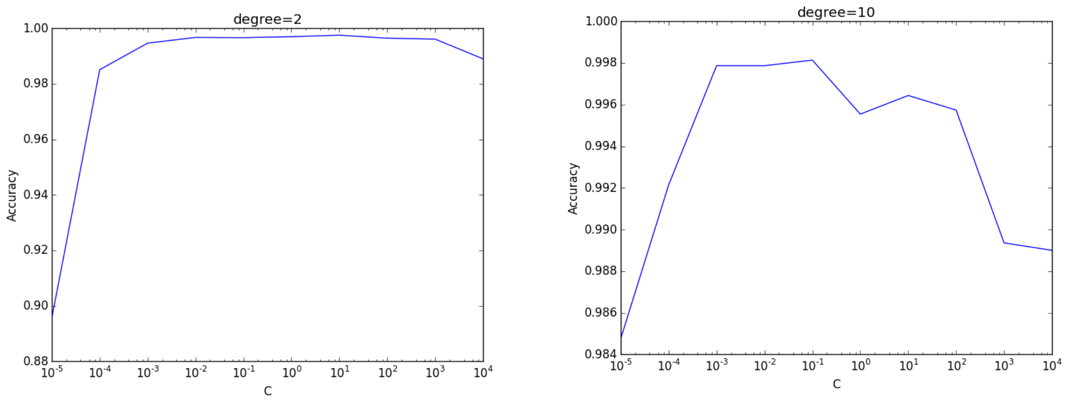


Figure 4: specific degree values using poly kernel and varying C on a log scale

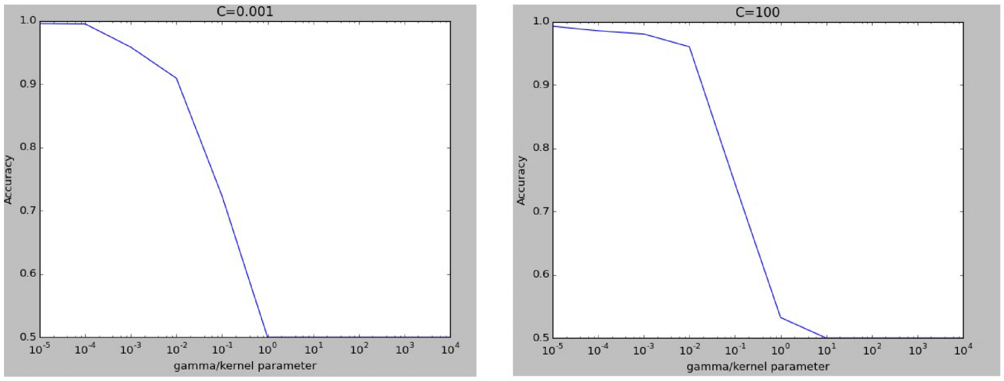


Figure 5: specific C values using gauss kernel and varying gamma on a log scale

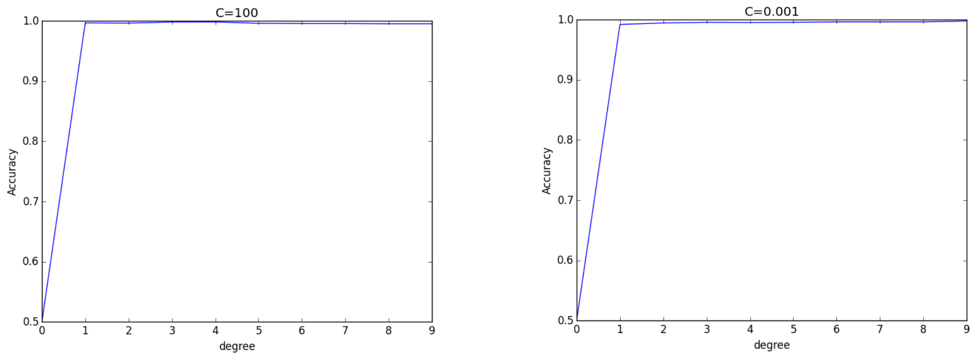


Figure 6: specific C values using poly kernel and varying degree on a log scale

It is evident from Figure 3 and Figure 5 for gaussian kernels that, SVM classifier has relatively high accuracies (0.94 to 0.98) for relatively high values of gamma (in the range from 10 to 100,000) and relatively low values of C (in the range from 0.1 to 0.001). Simillarly, it is evident from Figure 4 and Figure 6 for polynomial kernels that, SVM classifier has relatively high accuracies (0.85 to 0.95) for relatively high values of degree (in the range from 4 to 10) and relatively low values of C (in the range from 0.0001 to 0.0000001).

Summary:

In summary, SVM classifier gives high accuracies for relatively large gamma/degree values and relatively small C. For polynomial kernel compared to for gaussian kernel, even more small vlaues of C are requied for accuracy of 0.9 or higher.

Relatively small vlaues of soft margin parameters and relatively large values of free parameters are preffered for better accuracy.

It is possible to get higher accuracy (≥ 0.85) for largest C (least preferred C)for largest gamma/degree (most preferred gamma/degree). But, it is not possible to get high accuracy for smallest C (most preferred C) for smallest gamma/degree (least preferred gamma/degree). Hence, choosing a preferred value of a free parameter for a given kernel is more critical than choosing a soft margin parameter.

5.2 Comparison of normalized and non-normalized results

Gaussian kernel:

For gaussian kernel for non-normalized data, we get accuracy for the change in the range of gamma from (0-0.001) to (10-100000) ranges from (0.55-0.90) to (0.95) for C = 10 when we run the code. In contrast, for gaussian kernel for normalized data, it is evident from the Figure 1(b) that the change in the accuracy for the change in the range of gamma from (0-0.001) to (100-100000) ranges from (0.5) to (0.9-0.95) for C = 0.001.

It is also evident that for gaussian kernel for non-normalized data the change in the accuracy for the change in the range of C from (0-0.001) to (10-100000) ranges from (0.95) to (0.82) for gamma = 100000. In contrast, for gaussian kernel for normalized data the change in the accuracy for the change in the range of C from (0-0.001) to (10-100000) ranges from (0.95) to (0.87) for gamma = 100000.

For non-normalized data, SVM accuracy is the smallest value (0.5) for C in the range (0.1-0.001) and gamma in the range (0.1-0.001). For non-normalized data, SVM accuracy is the largest value (above 0.90) for C in the range (0.1-0.001) and gamma in the range (10-100000).

For normalized data, SVM accuracy is the smallest value (0.5) for C in the range (10-0.001) and gamma in the range (10-0.001). For non-normalized data, SVM accuracy is the largest value (0.90) for C in the range (10-0.001) and gamma in the range (1000-100000).

All the above plots are done using non-normalized data. The above analysis involve normalized data plots which can be found in the appendix 2 below.

Polynomial kernel:

For non-normalized data, SVM accuracy is the smallest value (0.5) for log10C in the range (-6 to -7) and degree in the range (2-3). For non-normalized data, SVM accuracy is the largest value (above 0.85) for log10C in the range (-7 to -6) and degree in the range (6-10).

For normalized data, SVM accuracy is the smallest value (0.5) for log10C in the range (-3 to -1) and degree in the range (2-7). For non-normalized data, SVM accuracy is the largest value (above 0.90) for log10C in the range (-2 to 0) and degree in the range (9-10).

Summary:

Normalizing data increases the highest possible accuracy from 0.95 to 0.99 for gaussian kernel and from 0.95 to 0.98 for polynomial kernel. Normalizing data relaxes the need for choosing smaller (optimum) value of C and makes it more needful to choose larger (optimum) value of free paramater in order to get better (above 0.9) accuracy.

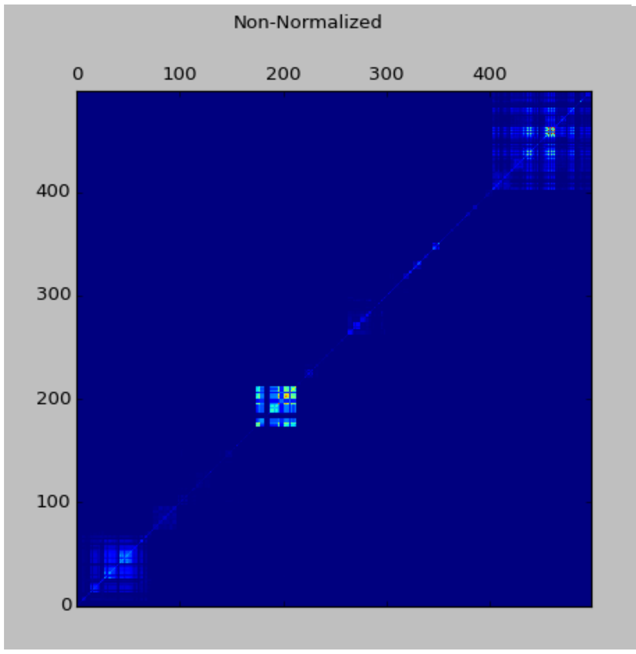
This results are obvious because normalization

5.3 Visualization of the kernel matrix and observations made about it

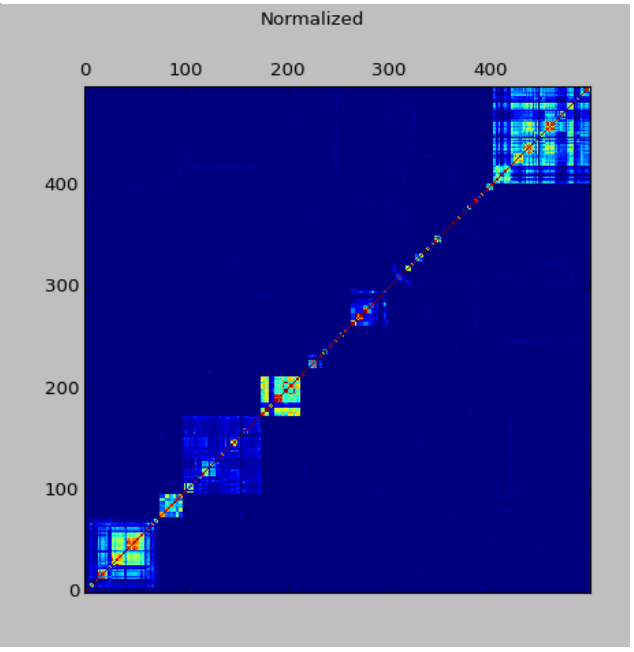
Kernel matrices for normalized and non-normalized datasets were created using following code:

```
# Function for implementing kernel matrix
Xt=np.transpose(X)
km = (X*Xt)
km = km.toarray()
plt.matshow(km, origin='lower')
plt.suptitle('Non-Normalized')
plt.show()
```

Kernel matrices for non-normalized and normalized data are shown in Figure 7. It is evident that, for the normalized dataset, the examples are redistributed across the feature space and for normalized dataset there are larger number of clusters than in the nom-normalized dataset. The larger number clusters suggest that there are more number of similar examples in the normalized dataset than there are in the non-normalized dataset.

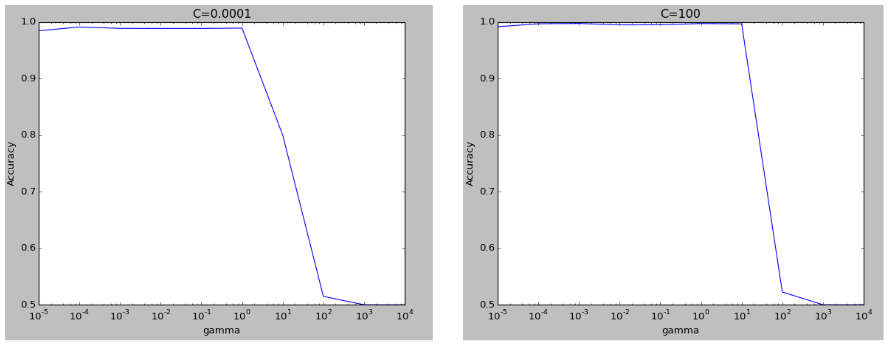


(a) Non-normalized data



(b) Normalized data

Figure 7: Kernel Matrix



[H]

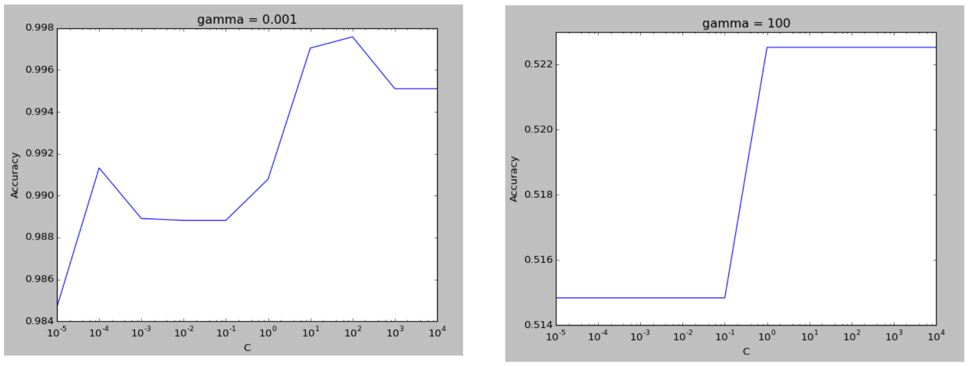
Figure 8: Accuracy for constant C with varying gamma for gauss

Appendix 1 - Code for creating 3D plot

The code inserted here is a matlab code. Data is imported using a simple python code in Appendix 3

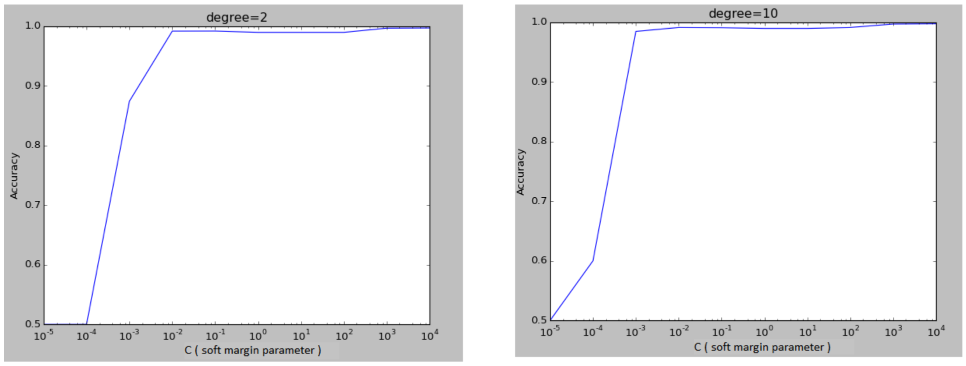
```
X = importdata('gamma.txt');
Y = importdata('C.txt');
Z = importdata('acc1.txt');

Tri = delaunay(X,Y);
C = Z;
trisurf(Tri,X,Y,Z,C);
set(gca,'XScale','log');
set(gca,'YScale','log');
X = importdata('degree.txt');
Y = importdata('C.txt');
Z = importdata('acc2.txt');
Tri = delaunay(X,Y);
C = Z;
trisurf(Tri,X,Y,C);
set(gca,'YScale','log');
colormap copper
```

Appendix 2 - Images for Normalized data



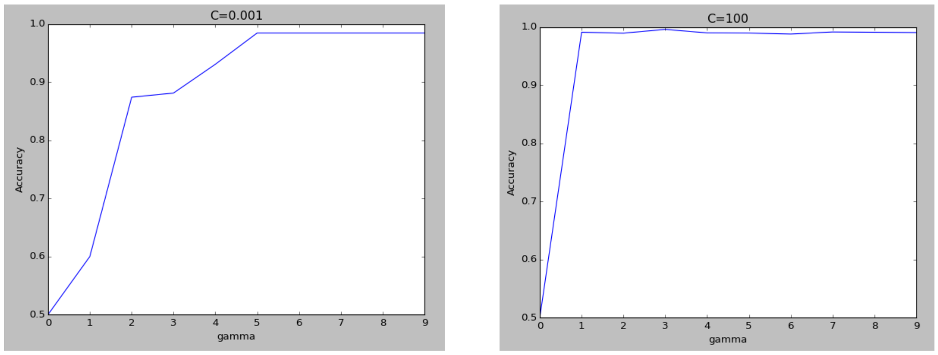
[H]

Figure 9: Accuracy for constant gamma with varying C for gauss



[H]

Figure 10: Accuracy for constant degree with varying C for gauss



[H]

Figure 11: Accuracy for constant C with varying degree for gauss

6 References

1. Dr. Ben Hur's lecture notes2. Code given in the schedule