# Machine Learning CW4

Aditya Rajagopal (ar4414)

March 25, 2017

## Pledge

I, Aditya Rajagopal, pledge that this assignment is completely my own work, and that I did not take, borrow or steal work from any other person, and that I did not allow any other person to use, have, borrow or steal portions of my work. I understand that if I violate this honesty pledge, I am subject to disciplinary action pursuant to the appropriate sections of Imperial College London.

The problems answered are : 1(a), 1(b), 2(a), 2(b), 2(c), 4(a), 4(b), 4(c)

## 1 Problem 1

### 1.1 Linear Regression Minimiser (1a)

The minimiser $w^*$ is obtained by differentiating the equation below and setting it to 0.

$$J(w) = \sum_{j=1}^{p}\sum_{i=1}^{n} \gamma_i((w^T x_i)_j - y_{i,j})^2 + \sum_{j=1}^{p}\sum_{k=1}^{d} w_{k,j}^2$$

Each term inside the summation is an individual term indexed as $i$ and $j$. It can be noticed that the summation over of $j$ can be removed and replaced with matrix additions, to give the following equation.

$$\begin{aligned} J(w) &= \sum_{i=1}^{n} \gamma_i((x_i^T w) - y_i^T)^2 + Tr(ww^T) \\ &= \sum_{i=1}^{n} \gamma_i((x_i^T w) - y_i^T)^T((x_i^T w) - y_i^T) + Tr(ww^T) \\ &= \sum_{i=1}^{n} \gamma_i(y_i^T - (x_i^T w))^T(y_i^T - (x_i^T w)) + Tr(ww^T) \end{aligned}$$

From standard matrix differentiation definitions (84) and (115) in http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf, we get,

$$\begin{aligned} \frac{\partial J(\omega)}{\partial \omega} &= \sum_{i=1}^{n} -2\gamma_i x_i(y_i^T - (x_i^T w)) + 2w \\ &= \sum_{i=1}^{n} 2\gamma_i x_i((x_i^T w) - y_i^T) + 2w \\ &= 0 \end{aligned}$$

This can then be written completely in terms of matrix operations as

$$2X^T\Gamma(Xw - Y) + 2w = 0$$

which simplifies to

$$w^* = (X^T\Gamma X + I)^{-1}X^T\Gamma Y$$

## 1.2 Kernelised Version (1b)

If all $\gamma_i = \gamma$, then

$$\begin{aligned} w^* &= (\gamma X^T X + I)^{-1} \gamma X^T Y \\ &= (X^T X + \frac{1}{\gamma} I)^{-1} X^T Y \end{aligned}$$

The minimiser from above after a non-linear transformation is $w^* = (Z^T Z + \frac{1}{\gamma} I)^{-1} Z^T Y$. For linear regression, the predictor $g(Z)$ is obtained as $g(Z) = Zw^*$. If we can show that this predictor can be written only in terms of $Z^T Z$, then we show that a kernelised version of the linear predictor is possible. $g(Z) = Z(Z^T Z + \frac{1}{\gamma} I)^{-1} Z^T Y$. Consider the equation,

$$(Z^T Z + \frac{1}{\gamma} I) Z^T = Z^T (Z Z^T + \frac{1}{\gamma} I) \tag{1}$$

Multiplying both sides of Eq.1 as follows we get an equivalent version of $w^*$,

$$\begin{aligned} (\mathbf{Z^T Z} + \frac{\mathbf{1}}{\gamma}\mathbf{I})^{-\mathbf{1}} (Z^T Z + \frac{1}{\gamma} I) Z^T (\mathbf{Z Z^T} + \frac{\mathbf{1}}{\gamma}\mathbf{I})^{-\mathbf{1}} \mathbf{Y} &= (\mathbf{Z^T Z} + \frac{\mathbf{1}}{\gamma}\mathbf{I})^{-\mathbf{1}} Z^T (Z Z^T + \frac{1}{\gamma} I)(\mathbf{Z Z^T} + \frac{\mathbf{1}}{\gamma}\mathbf{I})^{-\mathbf{1}} \mathbf{Y} \\ Z^T (Z Z^T + \frac{1}{\gamma} I)^{-1} Y &= (Z^T Z + \frac{1}{\gamma} I)^{-1} Z^T Y \\ Z^T (Z Z^T + \frac{1}{\gamma} I)^{-1} Y &= w^* \end{aligned}$$

Therefore,

$$\begin{aligned} g(Z) &= Zw^* \\ &= Z Z^T (Z Z^T + \frac{1}{\gamma} I)^{-1} Y \\ &= K (K + \frac{1}{\gamma} I)^{-1} Y \end{aligned}$$

# 2 Problem 2

## 2.1 Maximum Margin Classifier (2a)

Consider the following definitions. Let $w*$ be the maximum margin classifier on all our data points which make up set $\mathcal{D}$. Let $w^-$ be the maximum margin classifier on the data set $\mathcal{D}^-$ that is made by removing a data point that is not a support vector.

For linearly separable data points, to obtain a maximum margin classifier, we aim to minimise,

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{n=1}^{N} \alpha_n (y_n (w^T x_n + b) - 1))$$

w.r.t $w$ and $b$ and maximise the lagrangian w.r.t $\alpha_n \geq 0$. Doing so result in the KKT conditions from which we imply that:

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n \tag{2}$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0 \tag{3}$$

$$\textit{either} \quad \alpha_n = 0 \quad \textit{or} \quad y_n (w^T x_n + b) = 1 \tag{4}$$

$y_n(w^T x_n + b) = 1$ is true only when $x_n$ is a support vector, hence, if $x_n$ is a support vector, $\alpha_n \neq 0$. Otherwise, $\alpha_n = 0$. Therefore, the only data points that affect the maximum margin classifier $w^*$, are the support vectors. Hence, removing a non-support vector point from $\mathcal{D}$ to give $\mathcal{D}^-$ does not change the classifier $w^*$. Therefore, **$\mathbf{w}^*$ is a classifier in $\mathcal{D}^-$.**

Next we show that any maximum margin classifier obtained is unique. To show this, we show that the lagrangian, that we minimise is convex. If this can be shown, then the minimisation will result in a global minimum, hence making the maximum margin classifier obtained unique.

The first term, in the lagrangian ($\frac{1}{2}w^Tw$) is shown to be 1-strongly convex as shown in lecture 8, slide 8. The second term in the lagrangian is a function of $w$ and $b$. As there are no terms that contain the interaction of these 2 variables, the second term is linear in $w$ and $b$. By definition, any linear function is convex. As the sum of 2 convex terms is convex, we have shown that the lagrangian being minimised is convex. Therefore, **the maximum margin minimiser obtained is unique**.

Now we show that $w^*$ is also the maximum margin minimiser in $\mathcal{D}^-$. Let us assume that $w^-$ is instead the maximum margin minimiser in $\mathcal{D}^-$. This means that in $\mathcal{D}^-$, it has a larger margin than $w^*$. From the first part of this proof, it was shown that adding or removing a non-support vector does not change the classifier. Therefore, adding back a non-support vector to $\mathcal{D}^-$ to get back $\mathcal{D}$ would not change the classifier $w^-$. This would mean that $w^-$ also has a larger margin than $w^*$ in $\mathcal{D}$, which is a contradiction to the initial assumptions. Therefore, $w^*$ **is also the maximum margin minimiser in** $\mathcal{D}^-$.

Thus, we can conclude that adding or removing a non-support vector does not change the maximum margin classifier.

## 2.2 Number of Essential Support Vectors 2(b)

A support vector is an essential support vector if the maximum margin classifier changes if the support vector is removed. For a given training matrix $X \in \mathbb{R}^{(d+1)\times n}$ which is made of upto $k \leq n$ support vectors and the rest of the columns as non-support vectors and classifiers $y \in \mathbb{R}$, we know that,

$$\begin{aligned} y &= w^T X \\ &= [b \quad w_1 \quad w_2 \dots w_d] \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{d,1} & x_{d,2} & \dots & x_{d,n} \end{bmatrix} \end{aligned}$$

From 2(a), we know that $w$ is unique, and to have a unique solution to the above equation, the matrix X needs to have full column rank as this means there are no free variables and hence there exists no null space. As we are in a $d+1$ dimensional space, we can have upto $d+1$ linearly independent vectors, but as we need full column rank, $n \leq d+1$. Therefore, there can be at most $d+1$ essential support vectors.

## 2.3 Test Error of the maximum margin classifier 2(c)

A property of the cross validation error $E_{cv}$ obtained using leave-one-out cross validation, is that it is an unbiased estimator of the expected value of the test error for a training data set of size N-1 points. Let the cross validation error obtained on one run of leave-one-out cross validation be $e_i$.

$$E_{cv} = \frac{1}{N}\sum_{i=1}^{N} e_i \tag{5}$$

From part question 2a, we know that the maximum margin separator is not changed if a point that is not a support vector is removed. As we are assuming linear separability, the classifier correctly classifies the data points with no error. Therefore, any $i$ that corresponds to training on a dataset that has a non-support vector point removed has $e_i = 0$.

For any $i$ that corresponds to a point which is a support vector, the error $e_i \leq 1$. Combining these 2 facts, we see that

$$\begin{aligned} E_{cv} &= \frac{1}{N}\sum_{i=1}^{N} e_i \\ &\leq \frac{num\ support\ vectors}{N} \end{aligned}$$

From part 2b, we know that there can be at most d+1 support vectors, therefore

$$E_{cv} \leq \frac{d+1}{N}$$

From the property stated above, if $\mathcal{D}$ is a data set of N-1 points, then $\mathbb{E}_{\mathcal{D}}[E_{out}] \leq \frac{d+1}{N}$. Therefore, for a data set $\mathcal{D}_1$ of n points, $\mathbb{E}_{\mathcal{D}_1}[E_{out}] \leq \frac{d+1}{n+1}$.

## 3 Problem 4 - Digit Classification

### 3.1 RBF Kernel Large Margin SVM 4(a)

From the analysis of the dataset from Section 3.3.2 in Assignment 1, it is known that the raw data provided is linearly classifiable as the training error obtained from the perceptron algorithm is 0. This means that it would be ideal to implement a hard margin SVM. However, the library used (scikit-learn) provides an SVM classifier class that only allows for the use of soft-margin SVM implementations. However, this can be solved by setting the value of C (soft-margin SVM error regulariser) is set to a high enough value. A high value of C will penalise the classifiers which don't perfectly classify the data set, hence forcing the SVM to select a classifier that gives a training error of 0.

The default value of C is set to 1.0 in the SVM function provided by the library, and testing on this did indeed give a training error of 0. However, higher values of C were also tested and it was noticed that the maximum margin classifier obtained did not change from the case of C = 1.0, hence confirming the aforementioned theory. As using a C of 1.0 ensured that the soft-margin SVM library essentially worked as a hard margin SVM, for the remainder of this question, the value of C used in tests is 1.0.

The RBF-Kernel is defined as $K(x, x') = e^{-\gamma||x-x'||^2}$ for any two $x, x' \in \mathcal{X} = \mathcal{R}^d$. The next step in finding the RBF Kernel large margin SVM was to cross-validate across $\gamma$ values to find the best classifier. Leave-100-out cross validation was carried across a range of 20 equally spaced $\gamma$ values in the interval (0, 0.015]. Note, as the data set is of size 1273, this takes 12 cross validation sets of size 100 and 1 of size 73. The results of the cross validation are shown in Fig.1 below.

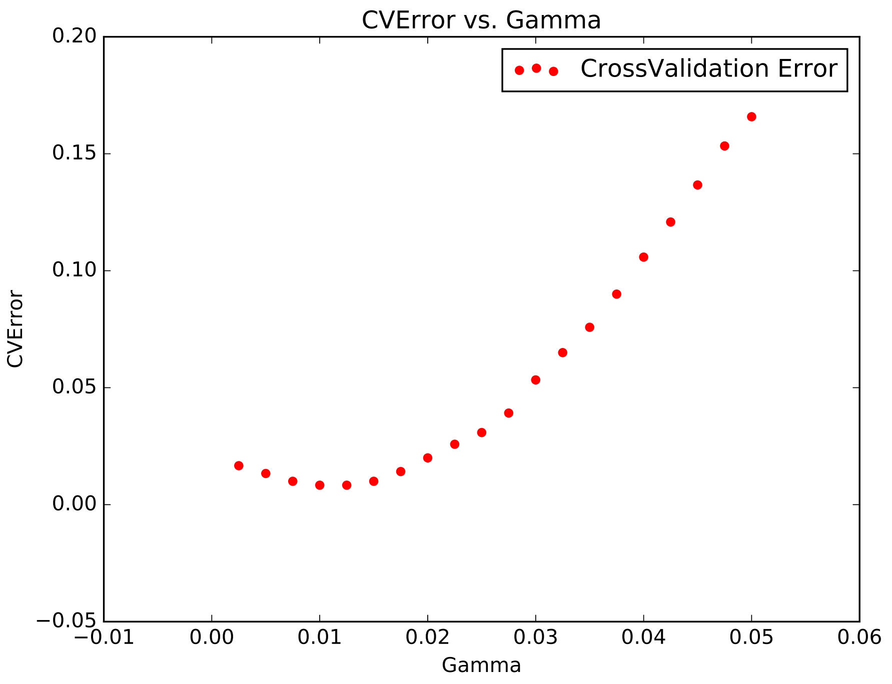


Figure 1: Cross Validation Errors for each Gamma

The results of the cross validation tests were:

- Best Gamma = 0.012
- Best Cross Validation Error = 0.00833
- Training Error = 0.0

- Test Error = 0.0357

Compared with Assignment 1, where the perceptron gave a test error of 0.0495, the SVM implementation improves on this by 27.8%. This is as one would expect as the SVM ensures that the classifier learnt has the largest possible margin, hence improving performance on a test data set.

## 3.2 Principal Component Analysis (PCA) 4(b)

The raw training data set used in the previous section, and the dataset that will be used in the following section, has 256 features. The concept behind PCA is that it ranks each feature according to the variance of the data points along that feature's axis. The greater the variance of the feature, the more useful the information it provides when learning. Reducing the feature set to contain $k$ features is then done by choosing the the $k$ features with the largest variance, and generating an orthogonal set of features from those.

For this test, an RBF-Kernel was again used to transform each of the $k$-dimensional PCA features matrices to an "infinite" dimensional space and classification was performed in that space using a soft-margin SVM. It is important to use a soft margin SVM, as there is no guarantee of linear separability even with the use of the rbf-kernel. For such a setup, the possible parameters to set using cross validation are $k$, $gamma$, and C where C is the soft margin SVM regularisation parameter. The procedure followed was as follows

- Perform leave-100-out (as done in the previous section) nested cross validation across various values for $k$, $gamma$, C
- The values of $k$ were chosen from the interval (0, 100] in steps of 1. The range was capped to $k = 100$ from quicker preliminary testing that showed that a plateau was reached in the cross validation and test errors well within this region.
- The values of $gamma$ were chosen from the range (0, 0.015] with 10 steps.
- The values of C were chosen from the range (0, 10] with 20 steps.
- Following this, for each value of $k$, the (C, $\gamma$) combination that produced the lowest cross validation error was obtained.
- For each of this ($k$, C, $\gamma$) combination, a training and test error was obtained and the cross validation error, training error, and test error was plotted as a function of $k$ as seen in Fig.2

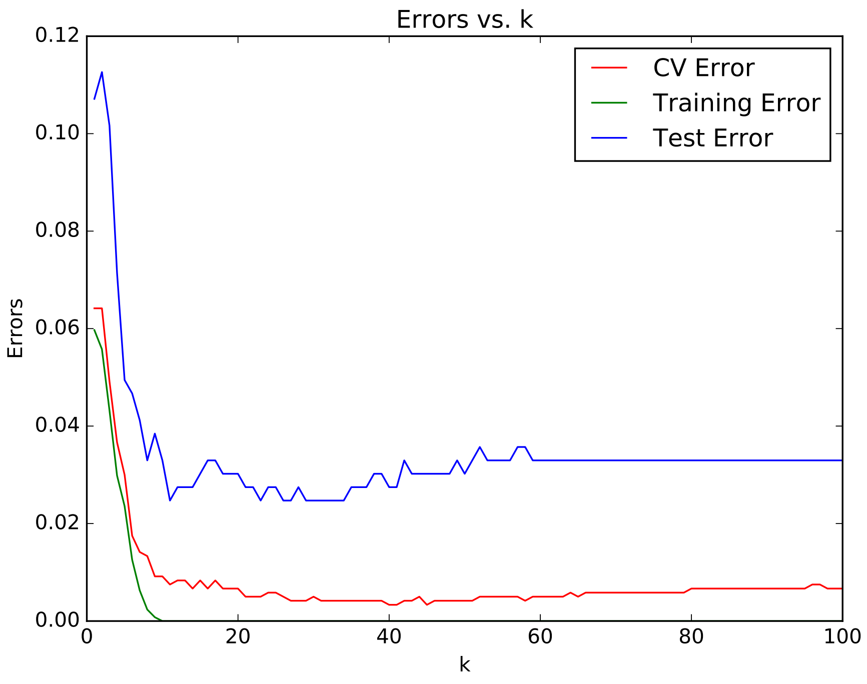


Figure 2: CV/Train/Test Errors as a function of K

The first point to notice is that the training error falls to 0 rapidly by around $k = 10$. This shows that for all values of $k \geq 10$, the data points are linearly separable in the rbf-kernel transformed $\mathcal{Z}$ space. The lowest value achieved by the test error was first reached for the ($k$, C, $\gamma$) combination of (11, 10.0, 0.011). The value of all three errors were:

- Cross Validation Error = 0.0075
- Training Error = 0.0
- Test Error = 0.0247

This is a significant improvement on the test error from the previous section of 30.8%. Furthermore, it is seen that the plateau in test error occurs for $k \geq 60$ at an error of 0.0357 which is the same as what was obtained in the previous section as is expected. The presence of this plateau in the test error gives way for interesting discussion about the effect of increasing the number of training data available for different values of $k$. The argument the following discussion aims to prove is that for values of $k < 60$, i.e. before the plateau, increasing the number of data points will benefit the test error, whereas for values of $k \geq 60$ adding more training data points will not benefit the test error.

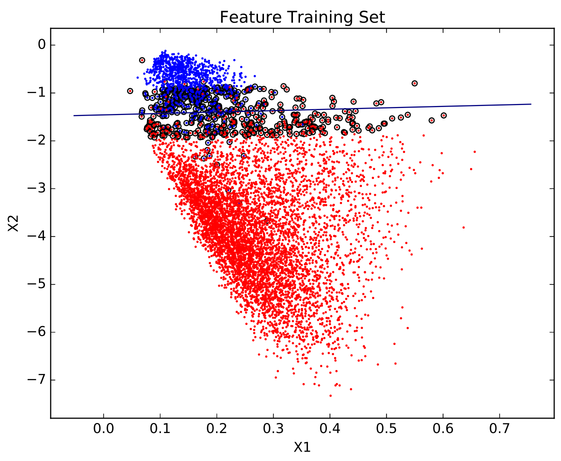
From Problem 2b, it is known that for a given $d$, there are at most $d+1$ support vectors such that the maximum margin classifier changes if the support vector is removed. Let these support vectors be called essential support vectors. In order to use this argument, it must first be shown that the non-linear transformation although going to an "infinite" dimensional space, can be approximated as a finite space. This argument follows from the fact that the RBF kernel can be expressed as taylor series expansion (an infinite) sum of features that show factorial decay in size with increasing dimension as seen in lecture 09, slide 9. Therefore, for large dimensions the effect of the features become negligible very quickly hence allowing us to approximate it as a finite space. As $k$ is increased by 1, the number of essential support vectors also increases by 1. However, from the plateau, it is seen that increasing this, does not change the maximum margin classifier. This means that every extra support vector that is selected is non-essential. Therefore, increasing the number of training data points for these values of $k$ will have no effect on the test error.

On the other hand, for values of $k < 60$, increasing k changes the maximum margin classifier, hence changing the test error. Therefore, each new support vector selected is essential and hence increasing the number of data points will help improve the test error as it will attempt the reach the plateau of the best possible test error. Training error reaches 0 very quickly, hence for most $k$, increasing the number of data points can't improve training error further. In terms of cross validation error, as the cross validation error is a reliable estimator of the out of sample error, it should follow the same trend as the test error.

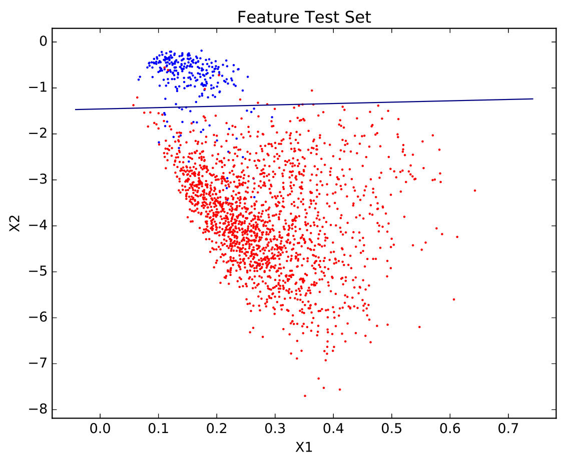
Another interesting point to note from the graphs obtained above is that the point at which the cross validation error is minimised and the test error is minimised are different. The cross validation error minimises around a value of $k = 40$, which is much higher than that of the test error. This is rather counter intuitive, as theory would suggest that the best test error would be obtained at the combination of ($k$, C, $\gamma$) that minimises the cross validation error. This discrepancy can be attributed to the fact that the cross validation error is only an expectation of the out of sample error, and there is a trade off between accuracy of prediction of the test error and the precision of the test error based on the fold used during cross validation.

## 3.3 Digit 1 against All 4(c)

In the graphs below, the blue points represent those points classified as 1's and circles around points represent points selected as support vectors.



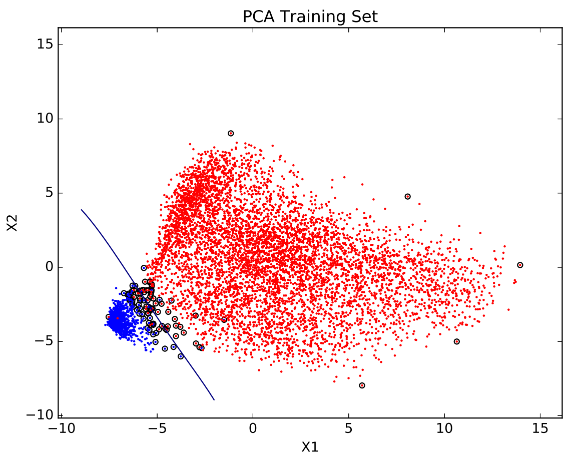
(a) Training Set



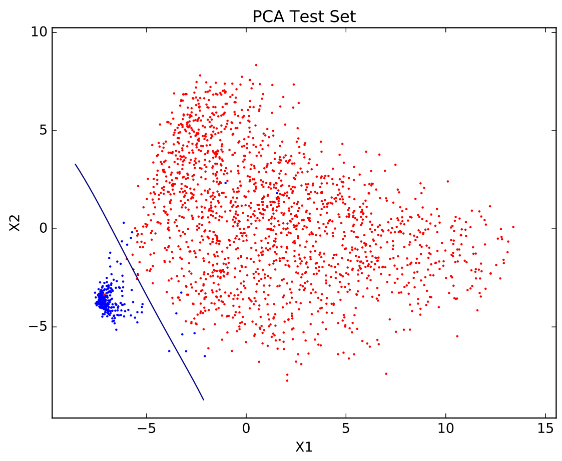
(b) Test Set

Figure 3: Hand Crafted Feature Vectors

For the hand crafted feature vectors, the errors obtained were:
Train Error = 0.0130
Test Error = 0.0209



(a) Training Set



(b) Test Set

Figure 4: 2-dimensional PCA Approximation

For the 2-dimensional PCA estimation of the data, the errors obtained were:
Train Error = 0.00329
Test Error = 0.00897

To obtain all the above values, cross validation was carried out on parameters $\gamma$ and C, for the range of values of (0,0.02] with 10 steps for $\gamma$ and (0, 10] with 20 steps for C. The large difference in test errors, can possibly be explained by the fact that the hand crafted features, may have been generated from a subjective analysis of which features may represent the dataset. However, the PCA approximation chooses features that give the most information about the dataset, hence fitting it better. This observation is further justified by the fact that the expectation of the test error $\mathcal{E}_{\mathcal{D}}[R(w,b)] \leq \mathcal{E}_{\mathcal{D}}[\frac{\#supportvec}{n+1}]$ as stated in lecture 08, slide 20. As can be seen from Fig.3a and Fig.4a, the number of support vectors selected when using PCA is far less than that when using the hand crafted feature set.