Introduction to Machine Learning, Assignment 4

Diederik Vink

July 12, 2017

Pledge

I, Diederik Adriaan Vink, 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.

Contents

Problems not done: 2d & 3

1 Problem 1

1.1 Problem 1a

Finding the closed form expression for the minimizer w^* requires differentiating $J(w)$ to get $J(w) = 0$. This is done by first rewriting $J(w)$ as follows:

$$\begin{aligned} J(w) &= \sum_{j=1}^{p}\sum_{i=1}^{n}\gamma_i((w^T x_i)_j - y_{i,j})^2 + \sum_{j=1}^{p}\sum_{k=1}^{d} w_{k,j}^2 \\ &= \sum_{i=1}^{n}\gamma_i((x_i^T w) - y_i)^2 + \mathbf{Tr}(ww^T) \qquad \text{(as the double summation is equal to trace of } ww^T) \end{aligned}$$

Simplifying $J(w)$ in this way make it a lot easier to perform the differentiation of the function $J(w)$. For the differentiation, it is important to know that $\frac{\partial \mathbf{Tr}(ww^T)}{\partial w} = 2w$ and $\frac{\partial \sum_{j=1}^{p}\sum_{i=1}^{n}\gamma_i((w^T x_i)_j - y_{i,j})^2}{\partial w} = \sum_{i=1}^{n}\gamma_i - 2(y_i - (x_i^T w))$ according to http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf.

$$\begin{aligned} J(w) &= \sum_{i=1}^{n}\gamma_i((x_i^T w) - y_i^T)^2 + \mathbf{Tr}(ww^T) \\ &= \sum_{i=1}^{n}\gamma_i((x_i^T w) - y_i^T)^T((x_i^T w) - y_i^T) + \mathbf{Tr}(ww^T) \\ &= \sum_{i=1}^{n}\gamma_i(y_i^T - (x_i^T w))^T(y_i^T - (x_i^T w)) + \mathbf{Tr}(ww^T) \quad \text{(to model to the equation from the pdf source stated above} \\ \frac{\partial J(\omega)}{\partial \omega} &= \sum_{i=1}^{n} -2\gamma_i x_i(y_i^T - (x_i^T w)) + 2w \qquad \text{to perform the following differentiation)} \\ &= \sum_{i=1}^{n} 2\gamma_i x_i((x_i^T w) - y_i^T) + 2w = 0 \end{aligned}$$

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

as $\sum_{i=1}^{n}\gamma_i = \Gamma$, $\sum_{i=1}^{n} x_i = X^T$ and $\sum_{i=1}^{n} y_i = Y^T$. Then rewriting for w:

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

1.2 Problem 1b

If we assume that $\gamma_i = \gamma$ for all i, then we can perform the non-linear transformation creating:

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

Performing a non-linear transformation ϕ on the minimizer $w*$ for Problem 1a will result in:

$$w^* = (Z^T Z + \frac{1}{\gamma}I)^{-1}Z^T Y \tag{2}$$

To obtain a predictor for linear regression, we have to perform $g(Z) = Zw^*$. Consider:

$$\begin{aligned} (Z^T Z + \frac{1}{\gamma}I)Z^T &= (Z^T Z)Z^T + (\frac{1}{\gamma}I)Z^T \qquad \text{(multiplying out)} \\ &= Z^T(ZZ^T + \frac{1}{\gamma}I) \end{aligned}$$

Writing this predictor in terms of only $ZZ^T (= Z^T Z)$ will prove that this can be entirely kernelized. Then multiply both sides by $(Z^T Z + \frac{1}{\gamma} I)^{-1}$ on the left and by $(ZZ^T + \frac{1}{\gamma} I)^{-1} Y$ on the right.

$$(\mathbf{Z^T Z} + \frac{\mathbf{1}}{\gamma}\mathbf{I})^{-\mathbf{1}}(Z^T Z + \frac{1}{\gamma} I)Z^T(\mathbf{ZZ^T} + \frac{\mathbf{1}}{\gamma}\mathbf{I})^{-\mathbf{1}}\mathbf{Y} = (\mathbf{Z^T Z} + \frac{\mathbf{1}}{\gamma}\mathbf{I})^{-\mathbf{1}} Z^T (ZZ^T + \frac{1}{\gamma} I)(\mathbf{ZZ^T} + \frac{\mathbf{1}}{\gamma}\mathbf{I})^{-\mathbf{1}}\mathbf{Y}$$

$$Z^T(ZZ^T + \frac{1}{\gamma} I)^{-1} Y = (Z^T Z + \frac{1}{\gamma} I)^{-1} Z^T Y$$

$$Z^T(ZZ^T + \frac{1}{\gamma} I)^{-1} Y = w^*$$

Therefore,

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

2 Problem 2

2.1 Problem 2a

For Problem 2a, g is the maximum linear classifier for $\mathcal{X}$ where $\mathcal{X}$ is a dataset of dimensions, $\mathcal{X}_-$ is $\mathcal{X}$ with a single non-support vector training data point removed and g_- is the maximum linear classifier for $\mathcal{X}_-$.

For Problem 2a, to show that (g) is not affected by removing or adding a training point to $\mathcal{X}$ that is not a support vector (SV) and does not violate the current maximum margin is done in the following three steps:

1. Show that g is a classifier in $\mathcal{X}_-$ if a non-support vector (NSV) training point is removed, and that g is still the maximum linear classifier of $\mathcal{X}$
2. Prove that g is unique
3. Show that it is not possible to have that if g_- has a larger margin on $\mathcal{X}_-$ than g, then it also has a larger margin on $\mathcal{X}$ which is a contradiction.

As is stated in the question, the data is linearly separable, so a hard-margin SVM will be used for the rest of the question. $w = g$ when Equation (3) is minimized.

Step 1

From Lecture 8 Slide 19, we know that:

$$\mathcal{L}(\mathrm{w}, \mathrm{b}, \alpha) = \frac{1}{2}\mathrm{w}^T\mathrm{w} + \sum_{i=1}^{n} \alpha_i[1 - y_i(\mathrm{w}^T x_i + \mathrm{b})] \tag{3}$$

where we aim to minimize w and b while maximizing α. From the KKT conditions, we see that at the minimum:

$$\mathrm{w} = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{4}$$

and either $\alpha_i = 0$ or $y_i(w^T x_i + b) = 1$. More specifically, for a NSV $\alpha_i = 0$ and for an SV: $y_i(w^T x_i + b) = 1$, x_i with $\alpha_i \neq 0$. As a result, applying this to Equation (4) it can be concluded that for all NSV training points, $w = 0$. We can therefore rewrite Equation (4) as:

$$\mathrm{w} = \sum_{x_i in SV} \alpha_i y_i x_i \tag{5}$$

From Equation (5) we can see that w is only affected by SV training points. As a result the optimal g (when w is at its minimum) is not affected by the addition and/or removal of any NSV training points. This shows and concludes step 1.

Step 2

Step 2 needs to prove that g is unique. As g is found by minimizing Equation (3) and it can be proved that Equation (3) is a convex function, then minimizing Equation (3) leads to a global minimum. Following the properties of convex functions, g is a global minimum and will be unique every time. It must bed noted that the sum of two convex functions is also a convex function. Therefore, to show that Equation (3) is a convex functions, Equations (6) and (7) must be shown to be convex as well.

$$\sum_{i=1}^{n} \alpha_i[1 - y_i(\mathrm{w}^T x_i + \mathrm{b})] \tag{6}$$

$$\frac{1}{2}\mathrm{w}^T\mathrm{w} \tag{7}$$

It is clear that Equation (6) is convex as both w and b are both linear and therefore the entire equation is linear. As any linear function in convex, this shows that Equation (6) is convex. For Equation (7) it shown to be a minimum as follows:

$$\begin{aligned} f(\mathrm{w}) &= \frac{1}{2}\mathrm{w}^T\mathrm{w} \\ &= \frac{1}{2}\|\mathrm{w}\|^2 \\ f_{\mathrm{w}}(\mathrm{w}) &= \|\mathrm{w}\| && \text{(differentiate } f(\mathrm{w})\text{)} \\ f_{\mathrm{ww}}(\mathrm{w}) &= 1 && \text{(differentiate } f_{\mathrm{w}}(\mathrm{w})\text{)} \end{aligned}$$

As the second derivate of $f(\mathrm{w}) > 0$, it can be concluded that this is a a global minimum, which is only possible if $f(\mathrm{w})$ is a convex function. This is further reinforced in Lecture 8 Slide 8. Putting everything together shows that Equation (3) is the sum of two convex functions and therefore a convex function itself. This means that minimizing Equation (3) will result in a global minimum making g unique.

Step 3

The final step is a proof by contradiction. From Step 1, we know that g is a classifier in $\mathcal{X}_-$. Now assume that g_- has a larger margin than g in $\mathcal{X}_-$. If an NSV training point is added to $\mathcal{X}_-$ to create $\mathcal{X}$, then suddenly g_- has a larger margin in $\mathcal{X}$ (backed up by Step 1). As proven in Step 2, the maximum margin classifier is unique, so it is not possible for g_- to have a larger margin in $\mathcal{X}$ than g. As a result it can be concluded, that g is the maximum margin classifier for both $\mathcal{X}$ and $\mathcal{X}_-$. This demonstrates that removing an NSV training point has no effect on the maximum margin classifier g.

2.2 Problem 2b

If a support vector is removed from the data set and this causes the maximum margin classifier to change, it is called an essential support vector. The aim of Problem 2b is to prove that there are at most $d+1$ essential support vectors. As shown in Problem 2a, we know that w is always unique if it is the maximum margin classifier. We also know that connection between the classification vector, y, the classifier w and the data set $\mathcal{X}$ can be represented by the following equation:

$$y = \mathrm{w}^T \mathcal{X} \tag{8}$$

In this Problem 2b for Equation (8) we try to solve for w. For our training data we know both the unique values for $\mathcal{X}$ and y, so to get a unique solution for w, we need $\mathcal{X}$ to be a full column rank matrix. This is a basic property of matrices. If $\mathcal{X}$ is full column rank, then it will have no free variables and the solution for w will be unique. $\mathcal{X}$ is known to be a $(d+1) \times n$ matrix with at most n support vectors x_k, $k = 1 \dots n$ and all other columns are filled with non-support vectors.

$$w^T \mathcal{X} = \begin{bmatrix} b & w_1 & w_2 & \dots & w_d \end{bmatrix} \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{d,1} & x_{d,2} & \dots & x_{d,n} \end{bmatrix} \tag{9}$$

As can be seen, this creates the matrix described above. The reason for a entire row of ones is because w_0 is a bias value b and the ones are in place to compensate for this bias. As this is in a $d+1$ dimensional space, there can be up to n linearly independent vectors. Therefore, there must be $n \leq d+1$ linearly independent vectors, meaning there can be at most $d+1$ essential support vectors.

2.3 Problem 2c

To obtain an unbiased estimator of the of the expected value of the test error of a data set of size $n-1$ points, one must use leave-one-out cross-validation error. The cross-validation error (E_{cv}) is formulated as :

$$E_{cv} = \frac{1}{N} \sum_{k=1}^{N} e_k \tag{10}$$

where e_k is the error for each leave-one-out test.

From Problem 2a, it is know that removing an non-support vector training point has no effect on the maximum margin classifier. The classifier will correctly classify all data points as the data is assumed to be linearly separable. It is important to note that, $e_k = 0$ if k corresponds to a training point that is a non-support vector. Furthermore, $e_k = 1$ if k corresponds to a training point that is a support vector. As a result:

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

From Problem 2b, the maximum number of essential support vectors is $d+1$, so

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

Using the above, stating that $\mathcal{X}$ is a data set of $n-1$ points, then $\mathbb{E}_{\mathcal{D}}[E_{out}] \leq \frac{d+1}{n}$. If the dataset is now of size n instead of $n-1$; $\mathbb{E}_{\mathcal{D}}[E_{out}] \leq \frac{d+1}{n+1}$.

3 Problem 4

3.1 Problem 4a

Problem 4a aimed at finding the optimal parameters for using SVM with an RBF kernel. From Assignment 1 it was deduced that the data which is being used for this problem is linearly separable. This was done by seeing that the perceptron algorithm returned a training error of 0, indicating linear separability.

The SVM used for the problem utilized an RBF kernel. An RBF kernel is defined as the function: $K(x, x') = e^{-\gamma\|x-x'\|^2}$ where $x, x' \in X = \mathbb{R}^d$.

Due to the linear separability of the data, it is clear that in an ideal situation a hard-margin SVM is used, as this SVM produces the best results for linearly separable data. Unfortunately, the svc class that is available in python is only a soft margin SVM. As a result, the C parameter can be utilized to replicate a hard-margin SVM nonetheless. The C parameter penalizes the error of a training point. The greater this parameter becomes, the greater the penalization of an error. As this penalization increases, eventually the SVM will act like a hard-margin SVM as any mistake is so harshly penalized that it is not tolerated and produce a training error of 0. To replicate a hard-margin SVM to a great enough degree, a high enough C must be used. After some minor empirical testing, it was concluded that having a $C = 1$ was enough to produce a hard-margin SVM classifier for this particular data set. Further increasing the value of C did not change the classifier (leaving training error at 0) and as a result $C = 1$ was settled upon.

After selecting the value of the parameter C, the next parameter to be selected is γ. γ was selected through cross-validation. The cross-validation that was employed was leave-100-out cross-validation across a range from 0 to 0.05 with intervals of 0.0001. Because $\frac{1273}{100}$ (1273 is the size of the dataset) does not result in a integer value, and this is a 13-fold cross-validation, the last cross-validation test set is only of size 73, not 100.

The results of running this cross-validation is shown below in Figure 1:

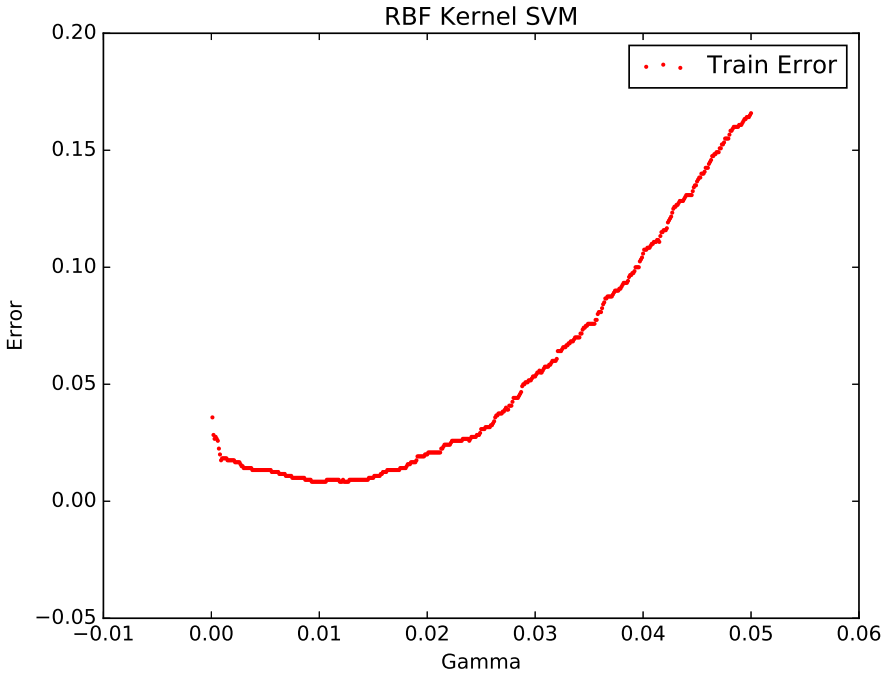

Figure 1: Cross validation error as γ increases

As can be seen in in the graph, there is a a clear minimum value of γ, at $\gamma = 0.0119$. On either side of this minimum γ value, the cross-validation error only increases. The training error of $\gamma = 0.0119$ is 0 as was expected with a hard-margin SVM. The test error of $\gamma = 0.0119$ is 3.57142857143%. This is a 27.78% improvement compared to the 4.945054945% achieved by using the perceptron algorithm from Assignment 1. Such a degree of improvement demonstrates the quality of using SVM.

3.2 Problem 4b

Part b of Problem 4 utilizes a PCA transformation to reduce the amount features present in the data matrix that is used in the SVM. PCA uses an orthogonal transformation to the feature set, and ranks each transformed set (principal component) by variance from largest to smallest. This results in a matrix of k uncorrelated orthogonal vectors for a k-dimensional PCA transformation. The value of k is always less than the value amount of features in the original matrix.

The implementation of PCA was done using the pca library from sklearn. This allowed for a simple procedure to transform the data matrix into a k-dimensional matrix according to any value of k where $k < 256$. Due to the nature of the PCA transformation, the resultant k-dimensional data set is no longer guaranteed to be linearly separable. As a result, a hard-margin SVM will no longer function properly, and a soft-margin SVM must be used.

Due to the fact that the SVM is now soft-margin, cross-validation across C was implemented. Compared to Problem 4a, this lead to various changes. The ranges for the parameters for k and the cross-validated parameters (C and γ) were as follows:

- k: 1-100 in steps of 1
- C: 1-25 in steps of 1
- γ: 0.0015-0.015 in steps of 0.0015

In Problem 4a a large range of gamma with very small increments were used to find the optimal value of γ. As there was already extensive data regarding the optimal value of γ and because computational resources were limited, as smaller range with lower precision of γ was used. The range of k that was used had to be less than 256 as mentioned before, but was otherwise not restricted. The reason no value greater than 100 was used can be seen in Figure 2 as it no longer provided any benefit. The same argument goes for the range of C values.

Regarding the procedure followed in the code, for every value of k, a new PCA transformation was executed to create a k-dimensional matrix. For every value of k, to implement a cross-validation across C and γ, two nested for loops were implemented to test every possible combination C and γ. The minimum cross-validation error was then recorded along with the C and γ that produced this minimum error. After having run the cross-validation for every value of k as specified above, these C and γ were used for the SVM of the train and test data sets (transformed to the corresponding k-dimensional matrix using PCA) to find the train and test error for each k, C and γ combination. The test error, training error and cross-validation error for the best performing C and γ for each k in the range from 1-100 is show in the Appendix as Figures 7 and 8.

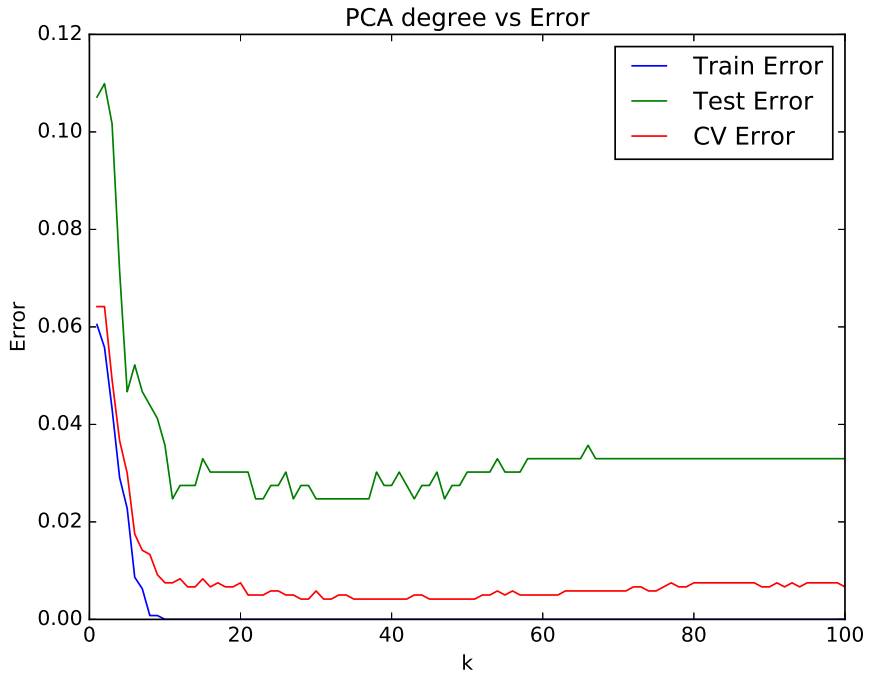

Figure 2: Test error, Training error and Cross-validation error for various values amounts of features (k)

Figure 2 demonstrates the cross-validation (cv), train and test error for each value of k. As stated before, the values of C and γ are specific for each value of k (as can be found in Figures 7 and 8). It can clearly be seen that the test error drops to zero at $k = 10$ and stays there. This is because the transformation to 11 dimensions or higher makes the data linearly separable. This means that the SVM can become a hard margin SVM for these values of k as in Problem 4a.

Furthermore, looking at the test error in Figure 2 (refer to Figures 7 and 8 for more detail), it can be seen that the best test error is achieved with $k = 11$ with a test error of 2.472%. This test error is the lowest that has been seen so far, and indicates that $k = 11$ is the ideal dimensional PCA transformation.

Beyond the observation of the optimal value of k to be used, an area of interest is where the values of $k \geq 66$. As can be seen there is a very clear and stable plateau in the value of the test error. This means that the maximum error margin is not changing for these values of $k \geq 66$. The explanation for the test error plateau can be derived from 2b) and Lecture 9 Slide 9. In Lecture 9 Slide 9 we see that:

$$K(x, x') = e^{-x^2} e^{-x'^2} \sum_{n=0}^{\infty} \frac{2^n x^n (x')^n}{n!} \tag{11}$$

This is the Taylor series expansion $K(x, x')$ which is an infinite sum. This series demonstrates how the size of each feature decays for increasing dimensions. As a result, for very high values of n, the summation section of $K(x, x')$ is essentially negligible. This will mean that even though using the RBF kernel is a transformation into an infinite dimension, it can essentially be considered finite at a high dimension. At this point, using the information from Problem 2b, we can say that there is a maximum number of essential support vectors of $k + 1$. When a support vector is essential, removing this support vector will change the maximum margin classifier. From the data it can be concluded that for values of $k \geq 66$, even though there is the possibility of having more essential support vectors, every data point that is a support vector is not essential and the maximum margin classifier does not change. For the values of $k < 66$, it is seen that there is a frequent change in the maximum margin classifier. This goes to show that as it becomes possible to have more essential support vectors. As this is possible it is utilized and the maximum margin classifier changes. If the amount of training data is increases, then the point at which the best set of essential support vectors is found is more likely to happen for lower and lower values of k. In other words, if there is more training data, the chance that a set of essential support vectors is found which is less than $k + 1$ increases for lower values of k. This implies that the plateau will happen earlier as the amount of training data increases.

The same conclusion could be made during training, but to a far lesser degree. The cross-validation error does show similar properties to the test error, as expected. Even though the properties are similar, they are not as prevalent for the cross-valdiation error in comparison to the test error. Nonetheless, a similar conclusion could be drawn during training. No real useful information like this can be drawn from the training error as this quickly drops down to 0.

3.3 Problem 4c

The approach for Problem 4c is very similar to before. The cross-validation was again for C and γ, where the ranges were 0.5-10 with steps of 0.5 for C and 0.002-0.02 with steps of 0.002 for γ. The handcrafted feature data is referred to as "Feature Data" and the data set that was transformed down to 2 features using a PCA transform is referred to as the "PCA Data".

Figure 3: Decision boundary feature data test set

Figure 4: Decision boundary feature data training set

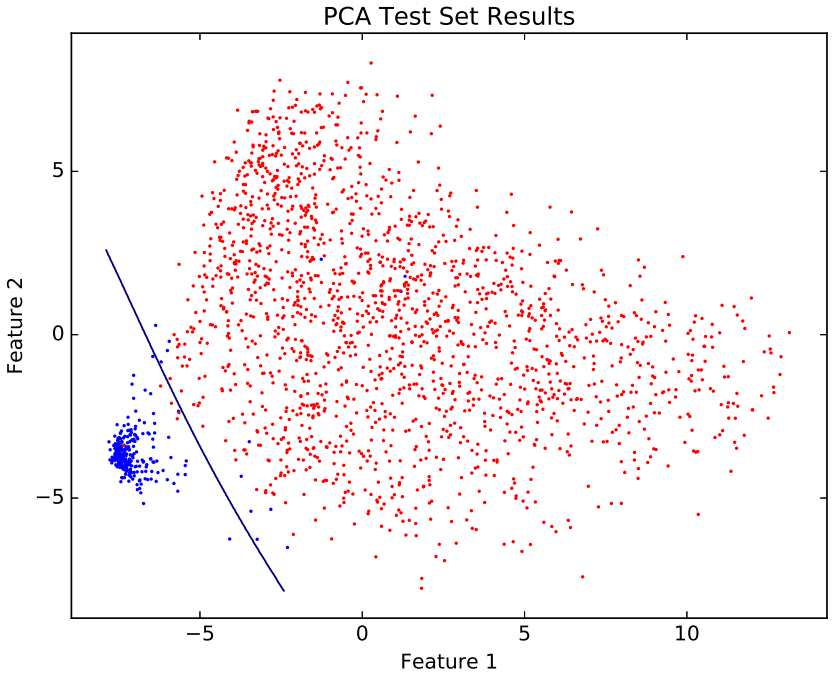

Figure 5: Decision boundary PCA data test set

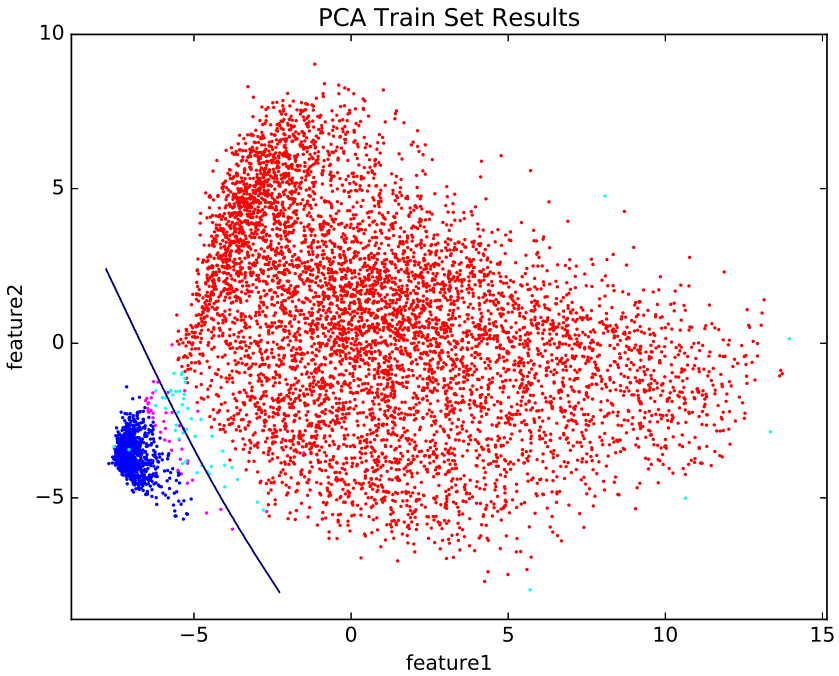

Figure 6: Decision boundary PCA data training set

	Feature Data	**PCA Data**
Optimal C, γ	1, 0.02	2.5, 0.018
Test Error	0.0224215246637	0.00896860986547
Train Error	0.014812782883	0.00342888492662
Cross-validation Error	0.015416666667	0.00347222222222

Table 1: Optimal C & γ values as well as test, train and cross-validation error for Feature and PCA Data

In all the graphs from here on out, the blue dots represent the non-support vector data for the digit 1, the

magenta dots represent the support vector data for digit 1, the cyan dots represent the support vector data for all the other digits and the red dots represent the non-support vector data for all other digits. As can be seen, none of the the graphs explicitly have a linear classification. This is confirmed in the graphs and in Table 1 as none of the training errors are 0. It can also be noticed that there is a considerably lower test error for the PCA Data compared to Feature Data. Furthermore, the amount of support vectors for the PCA data is notably lower. Looking at Lecture 8 Slide 20, it is seen that the expectation of the test error is bounded by $\frac{\#supp.vec}{n+1}$ where n is the number of training points. As a result, from the training set graphs for both Feature Data and PCA Data, it could have clearly been predicted that PCA Data would outperform Feature Data. An explanation for this occurrence is that the features from PCA Data is directly derived from the data, and is calculated as a result from the data. On the other hand, the features in Feature Data were handcrafted and therefore may have been attempted to be aimed at accurately describing the data but have less of a guarantee to do so. As a result, the features for PCA Data are far more appropriate, resulting in less support vectors and a lower test error. The cross-validation error and train errors was also lower for the same reasons.

```
K:  1   C:  23.0        Gamma:  0.009   cvError:  0.0641666666667       trainError:  0.0604870384918    testError:  0.107142857143
K:  2   C:  3.0         Gamma:  0.006   cvError:  0.0641666666667       trainError:  0.0557737627651    testError:  0.10989010989
K:  3   C:  4.0         Gamma:  0.0135  cvError:  0.0491666666667       trainError:  0.0432050274941    testError:  0.101648351648
K:  4   C:  5.0         Gamma:  0.0045  cvError:  0.0366666666667       trainError:  0.0290652003142    testError:  0.0714285714286
K:  5   C:  5.0         Gamma:  0.012   cvError:  0.03  trainError:  0.0227808326787    testError:  0.046703296703
K:  6   C:  4.0         Gamma:  0.009   cvError:  0.0175        trainError:  0.00864100549882   testError:  0.052197802197
K:  7   C:  6.0         Gamma:  0.006   cvError:  0.0141666666667       trainError:  0.00628436763551   testError:  0.046703296703
K:  8   C:  25.0        Gamma:  0.0135  cvError:  0.0133333333333       trainError:  0.000785545954438  testError:  0.043956043956
K:  9   C:  17.0        Gamma:  0.0105  cvError:  0.00916666666667      trainError:  0.000785545954438  testError:  0.041208791208
K:  10  C:  25.0        Gamma:  0.0135  cvError:  0.0075        trainError:  0.0       testError:  0.035714285714
K:  11  C:  17.0        Gamma:  0.0105  cvError:  0.0075        trainError:  0.0       testError:  0.024725274725
K:  12  C:  25.0        Gamma:  0.0135  cvError:  0.00833333333333      trainError:  0.0       testError:  0.0274725274725
K:  13  C:  17.0        Gamma:  0.0105  cvError:  0.00666666666667      trainError:  0.0       testError:  0.0274725274725
K:  14  C:  8.0         Gamma:  0.0135  cvError:  0.00666666666667      trainError:  0.0       testError:  0.0274725274725
K:  15  C:  25.0        Gamma:  0.0135  cvError:  0.00833333333333      trainError:  0.0       testError:  0.032967032967
K:  16  C:  13.0        Gamma:  0.015   cvError:  0.00666666666667      trainError:  0.0       testError:  0.0302197802198
K:  17  C:  3.0         Gamma:  0.015   cvError:  0.0075        trainError:  0.0       testError:  0.0302197802198
K:  18  C:  4.0         Gamma:  0.015   cvError:  0.00666666666667      trainError:  0.0       testError:  0.0302197802198
K:  19  C:  4.0         Gamma:  0.015   cvError:  0.00666666666667      trainError:  0.0       testError:  0.0302197802198
K:  20  C:  25.0        Gamma:  0.0135  cvError:  0.0075        trainError:  0.0       testError:  0.0302197802198
K:  21  C:  3.0         Gamma:  0.0135  cvError:  0.005         trainError:  0.0       testError:  0.0302197802198
K:  22  C:  5.0         Gamma:  0.0135  cvError:  0.005         trainError:  0.0       testError:  0.0247252747253
K:  23  C:  25.0        Gamma:  0.0135  cvError:  0.005         trainError:  0.0       testError:  0.0247252747253
K:  24  C:  4.0         Gamma:  0.0135  cvError:  0.00583333333333      trainError:  0.0       testError:  0.0274725274725
K:  25  C:  3.0         Gamma:  0.0135  cvError:  0.00583333333333      trainError:  0.0       testError:  0.0274725274725
K:  26  C:  4.0         Gamma:  0.015   cvError:  0.005         trainError:  0.0       testError:  0.0302197802198
K:  27  C:  25.0        Gamma:  0.0135  cvError:  0.005         trainError:  0.0       testError:  0.0247252747253
K:  28  C:  25.0        Gamma:  0.0135  cvError:  0.00416666666667      trainError:  0.0       testError:  0.0274725274725
K:  29  C:  25.0        Gamma:  0.0135  cvError:  0.00416666666667      trainError:  0.0       testError:  0.0274725274725
K:  30  C:  25.0        Gamma:  0.0135  cvError:  0.00583333333333      trainError:  0.0       testError:  0.0247252747253
K:  31  C:  7.0         Gamma:  0.012   cvError:  0.00416666666667      trainError:  0.0       testError:  0.0247252747253
K:  32  C:  6.0         Gamma:  0.012   cvError:  0.00416666666667      trainError:  0.0       testError:  0.0247252747253
K:  33  C:  25.0        Gamma:  0.0135  cvError:  0.005         trainError:  0.0       testError:  0.0247252747253
K:  34  C:  25.0        Gamma:  0.0135  cvError:  0.005         trainError:  0.0       testError:  0.0247252747253
K:  35  C:  6.0         Gamma:  0.012   cvError:  0.00416666666667      trainError:  0.0       testError:  0.0247252747253
K:  36  C:  6.0         Gamma:  0.012   cvError:  0.00416666666667      trainError:  0.0       testError:  0.0247252747253
K:  37  C:  6.0         Gamma:  0.012   cvError:  0.00416666666667      trainError:  0.0       testError:  0.0247252747253
K:  38  C:  25.0        Gamma:  0.0135  cvError:  0.00416666666667      trainError:  0.0       testError:  0.0302197802198
K:  39  C:  6.0         Gamma:  0.012   cvError:  0.00416666666667      trainError:  0.0       testError:  0.0274725274725
K:  40  C:  17.0        Gamma:  0.0105  cvError:  0.00416666666667      trainError:  0.0       testError:  0.0274725274725
K:  41  C:  6.0         Gamma:  0.012   cvError:  0.00416666666667      trainError:  0.0       testError:  0.0302197802198
K:  42  C:  17.0        Gamma:  0.0105  cvError:  0.00416666666667      trainError:  0.0       testError:  0.0274725274725
K:  43  C:  3.0         Gamma:  0.0105  cvError:  0.005         trainError:  0.0       testError:  0.0247252747253
K:  44  C:  3.0         Gamma:  0.0105  cvError:  0.005         trainError:  0.0       testError:  0.0274725274725
K:  45  C:  6.0         Gamma:  0.012   cvError:  0.00416666666667      trainError:  0.0       testError:  0.0274725274725
K:  46  C:  17.0        Gamma:  0.0105  cvError:  0.00416666666667      trainError:  0.0       testError:  0.0302197802198
K:  47  C:  13.0        Gamma:  0.0075  cvError:  0.00416666666667      trainError:  0.0       testError:  0.0247252747253
K:  48  C:  6.0         Gamma:  0.012   cvError:  0.00416666666667      trainError:  0.0       testError:  0.0274725274725
K:  49  C:  17.0        Gamma:  0.0105  cvError:  0.00416666666667      trainError:  0.0       testError:  0.0274725274725
K:  50  C:  17.0        Gamma:  0.0105  cvError:  0.00416666666667      trainError:  0.0       testError:  0.0302197802198
```

Figure 7: Optimal C and γ values for every value of K from 1-50

```
K:  51  C:  17.0        Gamma:  0.0105  cvError:  0.00416666666667      trainError:  0.0       testError:  0.0302197802198
K:  52  C:  7.0         Gamma:  0.012   cvError:  0.005         trainError:  0.0       testError:  0.0302197802198
K:  53  C:  6.0         Gamma:  0.012   cvError:  0.005         trainError:  0.0       testError:  0.0302197802198
K:  54  C:  25.0        Gamma:  0.0135  cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  55  C:  6.0         Gamma:  0.012   cvError:  0.005         trainError:  0.0       testError:  0.0302197802198
K:  56  C:  6.0         Gamma:  0.012   cvError:  0.00583333333333      trainError:  0.0       testError:  0.0302197802198
K:  57  C:  17.0        Gamma:  0.0105  cvError:  0.005         trainError:  0.0       testError:  0.0302197802198
K:  58  C:  17.0        Gamma:  0.0105  cvError:  0.005         trainError:  0.0       testError:  0.032967032967
K:  59  C:  17.0        Gamma:  0.0105  cvError:  0.005         trainError:  0.0       testError:  0.032967032967
K:  60  C:  17.0        Gamma:  0.0105  cvError:  0.005         trainError:  0.0       testError:  0.032967032967
K:  61  C:  17.0        Gamma:  0.0105  cvError:  0.005         trainError:  0.0       testError:  0.032967032967
K:  62  C:  17.0        Gamma:  0.0105  cvError:  0.005         trainError:  0.0       testError:  0.032967032967
K:  63  C:  17.0        Gamma:  0.0105  cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  64  C:  17.0        Gamma:  0.0105  cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  65  C:  23.0        Gamma:  0.009   cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  66  C:  25.0        Gamma:  0.0135  cvError:  0.00583333333333      trainError:  0.0       testError:  0.0357142857143
K:  67  C:  23.0        Gamma:  0.009   cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  68  C:  23.0        Gamma:  0.009   cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  69  C:  23.0        Gamma:  0.009   cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  70  C:  23.0        Gamma:  0.009   cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  71  C:  23.0        Gamma:  0.009   cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  72  C:  23.0        Gamma:  0.009   cvError:  0.00666666666667      trainError:  0.0       testError:  0.032967032967
K:  73  C:  23.0        Gamma:  0.009   cvError:  0.00666666666667      trainError:  0.0       testError:  0.032967032967
K:  74  C:  23.0        Gamma:  0.009   cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  75  C:  17.0        Gamma:  0.0105  cvError:  0.00583333333333      trainError:  0.0       testError:  0.032967032967
K:  76  C:  17.0        Gamma:  0.0105  cvError:  0.00666666666667      trainError:  0.0       testError:  0.032967032967
K:  77  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  78  C:  23.0        Gamma:  0.009   cvError:  0.00666666666667      trainError:  0.0       testError:  0.032967032967
K:  79  C:  23.0        Gamma:  0.009   cvError:  0.00666666666667      trainError:  0.0       testError:  0.032967032967
K:  80  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  81  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  82  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  83  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  84  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  85  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  86  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  87  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  88  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  89  C:  17.0        Gamma:  0.0105  cvError:  0.00666666666667      trainError:  0.0       testError:  0.032967032967
K:  90  C:  17.0        Gamma:  0.0105  cvError:  0.00666666666667      trainError:  0.0       testError:  0.032967032967
K:  91  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  92  C:  23.0        Gamma:  0.009   cvError:  0.00666666666667      trainError:  0.0       testError:  0.032967032967
K:  93  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  94  C:  23.0        Gamma:  0.009   cvError:  0.00666666666667      trainError:  0.0       testError:  0.032967032967
K:  95  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  96  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  97  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  98  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  99  C:  23.0        Gamma:  0.009   cvError:  0.0075        trainError:  0.0       testError:  0.032967032967
K:  100         C:  23.0        Gamma:  0.009   cvError:  0.00666666666667      trainError:  0.0       testError:  0.032967032967
```

Figure 8: Optimal C and γ values for every value of K from 51-100