Imperial College
London

EE3-23 Machine Learning

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Assignment 4

Anand Kasture (CID: 00832896)

Date: March 25, 2017

Pledge

I, **Anand Kasture**, 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.

Problem 1

In this section, we are provided with a training data set $\{(x_i, y_i), i = 1, \ldots, n\}$ where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}^p$. We aim to learn the weight matrix $w^\star \in \mathbb{R}^{d \times p}$ that minimises the cost function shown in Equation 1. The problem is effectively set-up as an empirical error formulation with Tykhonov regularisation. Note that the $\gamma_i > 0$ scalar variable allows us to specify the weighting associated with the errors across different (x_i, y_i) pairs.

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

Part (A)

We will now derive a closed-form expression for the optimal weight matrix $w^\star \in \mathbb{R}^{d \times p}$. Let the data matrix $X = (x_1, \ldots, x_n)^{\mathrm{T}}$, the matrix of target values $Y = (y_1, \ldots, y_n)^{\mathrm{T}}$ and the diagonal γ-weight matrix $\Gamma = \mathrm{diag}(\gamma_1, \ldots, \gamma_n)$. The first step in minimising the cost function $J(w)$ is to re-express Equation 1 in terms of the matrices X, Y and Γ. The steps carried out in achieving this are shown below:

$$\begin{aligned}
J(w) &= \sum_{j=1}^{p} \sum_{i=1}^{n} \gamma_i \left[\left(w^{\mathrm{T}} x_i \right)_j - y_{i,j} \right]^2 + \sum_{j=1}^{p} \sum_{k=1}^{d} w_{k,j}^2 && \text{(2a)} \\
&= \mathrm{tr}[(Xw - Y)^{\mathrm{T}} \Gamma (Xw - Y)] + \mathrm{tr}[w^{\mathrm{T}} w] && \text{(2b)} \\
&= \mathrm{tr}[(Xw - Y)^{\mathrm{T}} (\Gamma Xw - \Gamma Y)] + \mathrm{tr}[w^{\mathrm{T}} w] && \text{(2c)} \\
&= \mathrm{tr}[(Xw)^{\mathrm{T}} \Gamma Xw - (Xw)^{\mathrm{T}} \Gamma Y - Y^{\mathrm{T}} \Gamma Xw + Y^{\mathrm{T}} \Gamma Y] + \mathrm{tr}[w^{\mathrm{T}} w] && \text{(2d)} \\
&= \mathrm{tr}[w^{\mathrm{T}} X^{\mathrm{T}} \Gamma Xw - w^{\mathrm{T}} X^{\mathrm{T}} \Gamma Y - Y^{\mathrm{T}} \Gamma Xw + Y^{\mathrm{T}} \Gamma Y] + \mathrm{tr}[w^{\mathrm{T}} w] && \text{(2e)} \\
&= \mathrm{tr}[w^{\mathrm{T}} X^{\mathrm{T}} \Gamma Xw] - \mathrm{tr}[w^{\mathrm{T}} X^{\mathrm{T}} \Gamma Y] - \mathrm{tr}[Y^{\mathrm{T}} \Gamma Xw] + \mathrm{tr}[Y^{\mathrm{T}} \Gamma Y] + \mathrm{tr}[w^{\mathrm{T}} w] && \text{(2f)}
\end{aligned}$$

There exist several well-documented expressions for the (partial) derivatives of traces in literature. We use those presented in [1] and evaluate as follows:

$$\begin{aligned}
&\frac{\partial}{\partial w} \mathrm{tr}[w^{\mathrm{T}} X^{\mathrm{T}} \Gamma Xw] = X^{\mathrm{T}} \Gamma Xw + (X^{\mathrm{T}} \Gamma X)^{\mathrm{T}} w = 2X^{\mathrm{T}} \Gamma Xw && \text{(3a)} \\
&\frac{\partial}{\partial w} \mathrm{tr}[w^{\mathrm{T}} X^{\mathrm{T}} \Gamma Y] = X^{\mathrm{T}} \Gamma Y && \text{(3b)} \\
&\frac{\partial}{\partial w} \mathrm{tr}[Y^{\mathrm{T}} \Gamma Xw] = (Y^{\mathrm{T}} \Gamma X)^{\mathrm{T}} = X^{\mathrm{T}} \Gamma Y && \text{(3c)} \\
&\frac{\partial}{\partial w} \mathrm{tr}[Y^{\mathrm{T}} \Gamma Y] = 0 && \text{(3d)} \\
&\frac{\partial}{\partial w} \mathrm{tr}[w^{\mathrm{T}} w] = 2w && \text{(3e)}
\end{aligned}$$

[1] http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf

Hereafter, we compute the partial derivative of the cost function w.r.t w, equate the resulting expression to 0 and solve for w.

$$\frac{\partial}{\partial w}J(w) = 2X^{\mathrm{T}}\Gamma Xw - X^{\mathrm{T}}\Gamma Y - X^{\mathrm{T}}\Gamma Y + 0 + 2w = 0 \tag{4a}$$

$$2X^{\mathrm{T}}\Gamma Xw - 2X^{\mathrm{T}}\Gamma Y + 2w = 0 \tag{4b}$$

$$X^{\mathrm{T}}\Gamma Xw - X^{\mathrm{T}}\Gamma Y + w = 0 \tag{4c}$$

$$(X^{\mathrm{T}}\Gamma X + I)w = X^{\mathrm{T}}\Gamma Y \tag{4d}$$

$$\therefore w = (X^{\mathrm{T}}\Gamma X + I)^{-1}(X^{\mathrm{T}}\Gamma Y) \tag{4e}$$

In order to confirm that the stationary point is indeed a minimum, we take the second derivative as shown below. Since Γ is a positive-definite matrix, the expression for the second-derivative also represents a positive-definite matrix.

$$\frac{\partial^2}{\partial w^2}J(w) = \frac{\partial}{\partial w}[2X^{\mathrm{T}}\Gamma Xw - 2X^{\mathrm{T}}\Gamma Y + 2w] \tag{5a}$$

$$= 2X^{\mathrm{T}}\Gamma X + 2I > 0 \tag{5b}$$

Therefore, we conclude that $w^\star = (X^{\mathrm{T}}\Gamma X + I)^{-1}(X^{\mathrm{T}}\Gamma Y)$ minimises Equation 1

Part (B)

For $\gamma_i = \gamma$, we can replace the diagonal Γ matrix with γI to obtain the new expression for $w^\star$ as shown below:

$$w^\star = \gamma(X^{\mathrm{T}}X + I)^{-1}(X^{\mathrm{T}}Y) \tag{6}$$

The objective here is to avoid computing any terms including $\phi(x)$, since these can prove to be computationally expensive especially for non-linear transforms to very high dimensions. A valid kernel enables us to replace any inner products in the $\mathcal{Z}$ with a single function that uses the original feature vectors. Consider the following equation for a prediction in the transformed space:

$$w^{\mathrm{T}}\phi(x) = \left(\gamma\left(\Phi^{\mathrm{T}}\Phi + I\right)^{-1}\Phi^{\mathrm{T}}Y\right)^{\mathrm{T}}\phi(x) \tag{7a}$$

$$= \gamma Y^{\mathrm{T}}\Phi\left(\Phi^{\mathrm{T}}\Phi + I\right)^{-1}\phi(x) \tag{7b}$$

Using the identity [2] shown in Equation 8, we transform Equation 7b to Equation 9.

$$(I + AB)^{-1}A = A(I + BA)^{-1} \tag{8}$$

$$\gamma Y^{\mathrm{T}}\left(\Phi\Phi^{\mathrm{T}} + I\right)^{-1}\Phi\phi(x) \tag{9}$$

We can now replace the cross-product in Equation 9 with some valid kernel $K(x_i, x_j)$ to obtain the following expression for $w^{\mathrm{T}}\phi(x)$:

$$w^{\mathrm{T}}\phi(x) = \gamma Y^{\mathrm{T}}\left(\bar{K} + I\right)^{-1}\bar{k} \tag{10}$$

where

$$\bar{k} = \Phi\phi(x) = \begin{bmatrix} K(x_1, x) \\ K(x_2, x) \\ \vdots \\ K(x_n, x) \end{bmatrix} \tag{11a}$$

$$\bar{K} = \Phi\Phi^{\mathrm{T}} = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \dots & K(x_1, x_n) \\ K(x_2, x_1) & K(x_2, x_2) & \dots & K(x_2, x_n) \\ \vdots & \dots & \ddots & \vdots \\ K(x_n, x_1) & K(x_n, x_2) & \dots & K(x_n, x_n) \end{bmatrix} \tag{11b}$$

Problem 2

Part (A)

In this section, we must show that the hard-margin SVM classifier remains unchanged when we remove points that are not flagged as support vectors and/or add new points that do not violate the existing margin.

The equations below specify the Lagrangian dual of the quadratic program that arises out of the the margin-maximising optimisation formulation. The α variables represent the Lagrange multipliers associated with the inequality constraint in the primal problem, and the C variable penalises the level of margin violation denoted by δ. We address the dual problem because the corresponding constraints are computationally simpler to solve [3].

$$\max \mathcal{L}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathrm{x_n^T x}_m \tag{12a}$$

$$\text{subject to} \quad 0 \le \alpha_n \le C \;\forall \mathrm{n} \in [1, \mathrm{N}] \quad \text{and} \quad \sum_{n=1}^{N} \alpha_n y_n = 0 \tag{12b}$$

[2] using matrix inversion identities found in `http://www0.cs.ucl.ac.uk/staff/gridgway/mil/mil.pdf`

[3] `https://www.csie.ntu.edu.tw/~cjlin/papers/bottou_lin.pdf`

$$\text{where} \quad \mathrm{w} = \sum_{\alpha_n > 0} \alpha_n y_n \mathrm{x}_n \quad \text{minimises} \quad \frac{1}{2}\mathrm{w}^T\mathrm{w} + C\sum_{n=1}^{N}\delta_n \tag{12c}$$

For linearly separable data, we can set $C = \infty$ i.e. require a solution with no margin violations. This implementation is known as the hard-margin SVM. Alternatively, a soft-margin SVM method can be implemented using $C < \infty$.

The complementary slack KKT condition stated below implicitly points to the fact that all support vectors are associated with non-zero values of the dual variable α_n. The points associated with the interior $\alpha_n = 0$ condition are known as interior points because these points are not located on the constraints i.e. the inequality constraint remains inactive.

$$\alpha_n[1 - y_n(w^{\mathrm{T}}x_n + b)] = 0 \tag{13}$$

$$w = \sum_{x \text{ is SV}} \alpha_n y_n x_n \tag{14}$$

$$b = 1 - y_n w^{\mathrm{T}} x_n, \; x_n \text{ is a support vector} \tag{15}$$

The separating hyperplane is fully defined by w and b. The interior points have no effect on these two variables since they correspond to $\alpha = 0$. Therefore, we may conclude that the addition and/or removal of samples that are not marked as support vectors does not have any effect on the classifier.

Part (B)

N/A

Part (C)

To prove that the expected test error of the maximum margin classifier is at most $\frac{d+1}{n+1}$ we use the error bound on support vectors as defined in Lecture 8[4]:

$$\mathbb{E}_{\mathcal{D}}[R(\mathbf{w}, b)] \leq \mathbb{E}_{\mathcal{D}}\left[\frac{\#supp.vec}{n+1}\right]$$

Part (c) concludes that the upper bound of support vectors that are required to change the maximum classifier as $d + 1$, therefore the inequality simplifies to,

$$\mathbb{E}_{\mathcal{D}}[R(\mathbf{w}, b)] \leq \frac{d+1}{n+1}$$

Part (D)

N/A

[4] https://bb.ic.ac.uk/bbcswebdav/pid-1035005-dt-content-rid-3593657_1/courses/DSS-EE3_23-16_17/lecture08.pdf

1 Problem 3

N/A

Problem 4

In Assignment 1, we used the perceptron learning algorithm to classify the digits 2 and 8. In this section, we investigate the performance of Support Vector Machine (SVM) methods applied on the same handwritten digit data set.

SVM methods are based on the concept of maximising the *margin* i.e. the euclidean distance between some candidate classification hyperplane and the nearest interior points on each side. This margin requirement effectively places a restriction on the growth function, and is therefore likely to improve out-of-sample generalisation.

Note that we make use of the in-built SVM functions provided with the Statistics and Machine Learning Toolbox (MATLAB 2016b) in order to generate the results presented in this section. The exact function signatures (including the configuration parameters) will be specified and explained accordingly.

Part (A)

The raw training data comprising of the higher dimensional feature vectors $x_i \in \mathbb{R}^{256}$ for the digits 2 and 8 is linearly separable. We know this because the a simple linear predictor gives rise to a zero training error on this data set. Therefore, the hard-margin SVM method can be utilised to separate this data.

The MATLAB function `fitcsvm` is used to produce a hard-margin SVM classifier on the training data [5]. This function is called in conjunction with `crossval` and `kfoldLoss` in order to obtain a 10-fold cross validation error for each value of the γ parameter in the user-defined search space. Note that γ arises in the expression for the Gaussian kernel as shown below:

$$K(x_i, x_j) = \exp\left(-\gamma \left\|x_i - x_j\right\|^2\right)$$

Figure 1 shows the cross-validation and test-error plots when cycling through different values of γ. The plot on the left uses a geometric sequence range $[10^{-5}, 10^5]$ whereas the plot on the right uses a narrow arithmetic sequence range $[10^{-5}, 10^2]$.

[5] `https://www.mathworks.com/help/stats/fitcsvm.html`

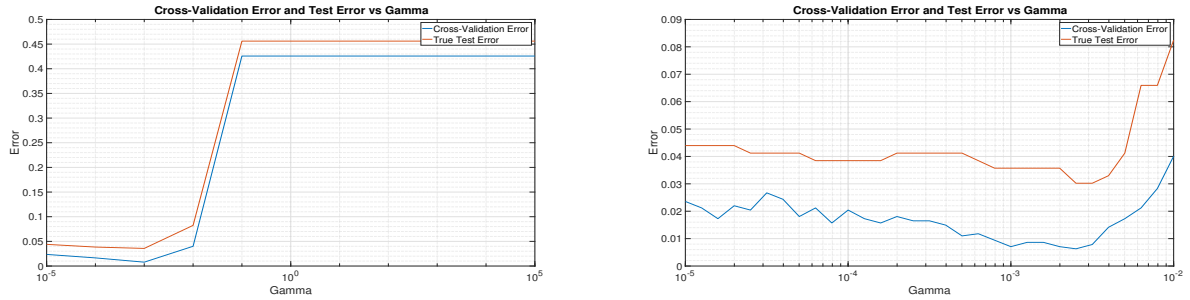

Figure 1: Cross-Validation Errors and Test Errors against γ parameter

These plots show that $\gamma > 10^{-2}$ produces sub-optimal test performance. This is an expected result since a relatively large γ value can lead to over-fitting. Note that the cross-validation error serves as a good approximation to the test error. Table 1 shows the optimal $\gamma_\star$ results obtained when searching across the two ranges. A test error of **3.0220%** compares well with a **5.4945%** value that was achieved using a simple perceptron.

	$\gamma^\star$	**Cross-Validation Error**	**Test Error**
Wide Range	0.001000	0.7855%	3.5714%
Narrow Range	0.002512	0.6284%	3.0220%

Table 1: Optimal SVM parameter selection results

Figure 2 illustrate the plots obtained when we search for a single optimal SVM Parameter γ by enabling the `OptimizeHyperparameters` option in the `fitcsvm` function. We use the default Sequential Minimal Optimization (SMO) routine to obtain an optimal value for KernelScale[6]. Note that KernelScale $= 1/\sqrt{\gamma}$. Table 2 presents the results obtained using this method.

	$\gamma^\star$	**Cross-Validation Error**	**Test Error**
SMO Optimiser	0.002037	0.7070%	3.5714%

Table 2: Optimal SVM parameter selection using SMO Optimisation

Because the optimiser (run for 30 iterations only) returns a value fairly close to the one obtained using the narrow range, this is the method we will be using to choose optimal parameter values in the subsequent sections.

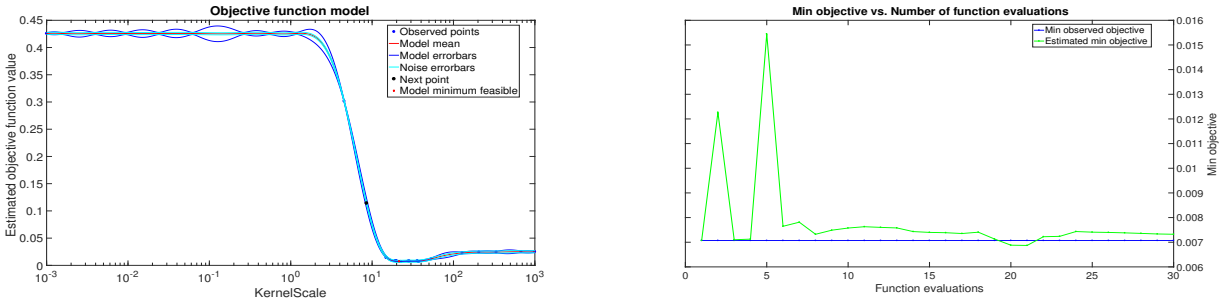

Figure 2: Obtaining $\gamma^\star$ using SVM Parameter Optimisation

[6]Fan, R.-E., P.-H. Chen, and C.-J. Lin. "Working set selection using second order information for training support vector machines." Journal of Machine Learning Research, Vol 6, 2005, pp. 18891918

Part (B)

In this section, we use Principal Component Analysis (PCA) to reduce the dimensionality of the raw training data from $\mathbb{R}^d \rightarrow \mathbb{R}^k$ where $1 \leq k \leq d$. The PCA coefficients are obtained using the `pca` function. We project the original feature vectors onto the space defined by the coefficient matrix and use the resulting data to train a soft-margin SVM classifier. The objective here is to obtain an optimal value for k such that the cross-validation error is minimised.

The training procedure here is similar to the manner in which the SVM method was implemented in Part A. However, we use the `OptimizeHyperparameters` option in the `fitcsvm` function to select **both** the `KernelScale` ($1/\sqrt{\gamma}$) and `BoxConstraint` (C) i.e. the optimiser must attempts to find the pair of these parameter values that minimises the 10-fold cross-validation error.

We iterate over every k value, and record the minimum cross-validation error produced when using the k-dimensional PCA approximation of the raw training data. The optimal $k^\star$ corresponds to the the scheme with the lowest overall cross-validation error. Figure 3 illustrates the training, cross-validation and test-error results as function of k. Table 3 reports the best-case performance. These results can be reproduced by running the `Q4B.m` script.

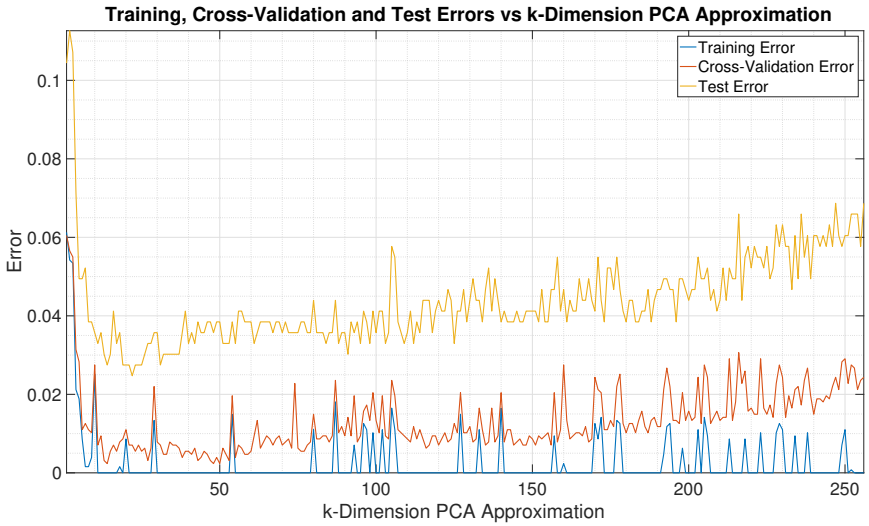

Figure 3: Training, Cross-Val and Test Errors vs k-Dimension PCA Approximation

A **2.7473%** test error improves upon the **3.0220%** recorded in the last section, and represents a worthwhile improvement over the **5.4945%** achieved with the simple perceptron classifier.

$k^\star$	$\gamma^\star$	$C^\star$	**Cross-Validation Error**	**Test Error**
14	0.060331	959.627	0.2357%	2.7473%

Table 3: Best-Case soft-margin SVM Performance using PCA approximation

We make the following observations on the PCA-Approximation scheme. Very small values of $k << k^\star$ (e.g. 1 or 2) show poor performance since we have discarded the information contained within all the other PCA components. Secondly, increasing $k > k^\star$, increases the VC-Dimension and leads to worsening generalisation performance; VC analysis tells us that

we need a higher number of training data points in order to maintain, or even improve, the generalisation to out-of-sample data.

Part (C)

In this section, we wish to classify the digit 1 from all the other digits in the training data set. We will implement, analyse and compare two different approaches which differ in the construction of the in-sample data set. Namely, we will train a soft-margin SVM on (i) 2-Dimensional PCA Approximation of the raw data and (ii) Hand-crafted feature vectors.

As was the case in Part B, we use the `OptimizeHyperparameters` option in the `fitcsvm` function to select **both** the `KernelScale` ($1/\sqrt{\gamma}$) and `BoxConstraint` (C) parameters. The table and the scatter plots below present the results that were obtained using this approach.

	$\gamma^\star$	$C^\star$	**Training Error**	**Cross-Va Error**	**Test Error**
2-D PCA Projection	2.8368	11.9152	0.301742%	0.329173%	0.647733%
Hand-Crafted Features	0.4097	0.1691	1.042381%	1.042381%	1.893373%

Table 4: Best-Case soft-margin SVM Performance using different Train Data Construction

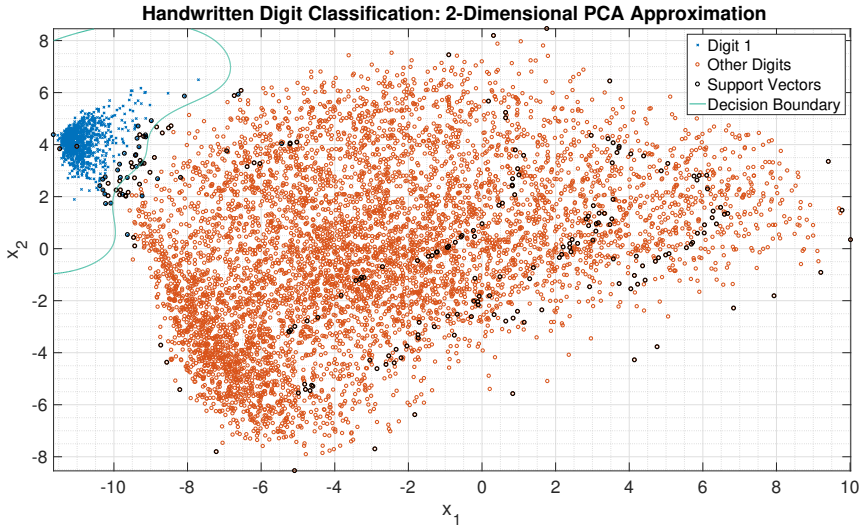

Figure 4: Handwritten Digit Classification: 2-Dimensional PCA Approximation

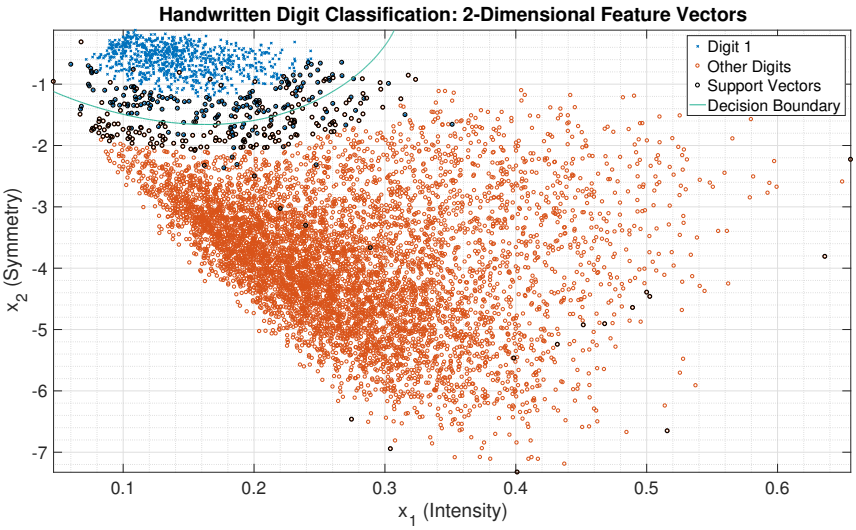

Figure 5: Handwritten Digit Classification: 2-Dimensional Feature Vectors

These results show that using the 2-D PCA approximation on the raw-data training data set produces superior test error probability and generalisation performance. The higher training error associated with the hand-crafted features indicates that the {Intensity, Symmetry} construction does not discriminate the data-set well. In comparison, the PCA approach uses the raw-data projection onto the first two principal components with the highest variance. Lastly, note that the PCA approach also produces an SVM solution with fewer support vectors (227 i 347) and is therefore expected to outperform the hand-crafted features for the same number of in-sample points.