Machine Learning Coursework 4

Jesus E. Garcia Condado

March 25, 2017

I, Jesus E. Garcia, 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.

Problems chosen

- Problem 1 (10 marks)
  - Part a (6 marks)
  - Part b (4 marks)
- Problem 2 (11 marks)
  - Part a (6 marks)
  - Part b (3 marks)
  - Part c (2 marks)
- Problem 4 (15 marks)
  - Part a (4 marks)
  - Part b (8 marks)
  - Part c (3 marks)

1 Problem 1

1.1 Part a

In the modified version of linear regression the aim is to minimize the weighted empirical error with Tykhonov regularization $J(w)$ defined as:

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

Where $w \in \mathcal{R}^{dxp}$ is the matrix of weights we want to find for a given set $X = (x_1, ..., x_n)^T$ of n data points $x_i \in \mathcal{R}^d$ and its corresponding $Y = (y_1, ..., y_n)^T$

predictions with $x_i \in \mathcal{R}^p$. We can therefore rewrite the error in terms of matrix operations as:

$$J(w) = \sum_{i=1}^{n} \gamma_i \left((x_i^T w) - y_i^T \right)^2 + Tr(ww^T) \tag{2}$$

Obtaining the derivative with respect to w and setting it to 0:

$$0 = \sum_{i=1}^{n} \gamma_i 2x_i((x_i^T w) - y_i) + 2w \tag{3}$$

The summation term corresponds to n weighted sums which can be expressed as:

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

Simple matrix algebra yields the following expression for w:

$$w = (X^T \Gamma X + I)^{-1} X^T \Gamma Y \tag{5}$$

1.2 Part b

Since for all i we have $\gamma_i = \gamma$, then $\Gamma = \gamma I$ where I is the identity matrix of the same dimensions as Γ. We can therefore rewrite our predictor as:

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

To show that a kernelized version is possible we must obtain an expression for a prediction $g(X) = Xw$ which has terms in XX^T. Then applying a linear transformation from X to Z would result in having terms in ZZ^T which can be conveniently computed as $K(X, X') = ZZ^T$. We will use the following equality:

$$(X^T X + \frac{1}{\gamma} I) X^T = (X^T X) X^T + (\frac{1}{\gamma} I) X^T = X^T (XX^T + \frac{1}{\gamma} I) \tag{7}$$

We can rewrite 6 by multiplying in both sides of 7 by the same terms:

$$(X^T X + \frac{1}{\gamma} I)^{-1} (X^T X + \frac{1}{\gamma} I) X^T (XX^T + \frac{1}{\gamma} I)^{-1} Y = (X^T X + \frac{1}{\gamma} I)^{-1} X^T (XX^T + \frac{1}{\gamma} I)(XX^T + \frac{1}{\gamma} I)^{-1} Y \tag{8}$$

$$X^T (XX^T + \frac{1}{\gamma} I)^{-1} Y = (X^T X + \frac{1}{\gamma} I)^{-1} X^T Y \tag{9}$$

$$X^T (XX^T + \frac{1}{\gamma} I)^{-1} Y = w \tag{10}$$

For any prediction $g(X) = Xw$ we have that $g(X) = XX^T (XX^T + \frac{1}{\gamma} I)^{-1} Y$ and applying a non-linear transform from X to Z results in a kernelized version $g(K(X, X')) = K(K + \frac{1}{\gamma} I)^{-1} Y$.

2 Problem 2

Given that g is a maximum margin linear classifier for the d-dimensional data set $\mathcal{X}$, we shall define a new data set $\mathcal{X}^-$, constructed such that the data point $x_i \in \mathcal{X}$, which is not a support vector, is not in $\mathcal{X}^-$. Then the new maximum margin linear classifier shall be defined as g^-.

In order to obtain the maximum margin linear separator for linearly separable data we must solve the Lagrangian formulation given in the notes as $\mathcal{L}(\mathrm{w}, \mathrm{b}, \alpha)$ by minimizing w and b while maximizing α:

$$\mathcal{L}(\mathrm{w}, \mathrm{b}, \alpha) = \frac{1}{2}|\mathrm{w}|^2 + \sum_{i=1}^{n} \alpha_i [1 - y_i(\mathrm{w}^T x_i + \mathrm{b})] \tag{11}$$

The KKT conditions then define the w at the solution as:

$$\mathrm{w} = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{12}$$

with the additional conditions of $\alpha_i = 0$ or $y_i(w^T x_i + b) = 1$. The values that satisfy the second condition will be defined as the support vectors (SV) and those that don't as non-support vectors. Therefore $x_i \in NSV \to \alpha_i = 0$ and we can then rewrite the summation from the non-zero terms as:

$$\mathrm{w} = \sum_{x_i \in SV} \alpha_i y_i x_i \tag{13}$$

From this we can conclude that adding or removing a NSV point does not change our prediction w.

Then if $\mathcal{L}(\mathrm{w}, \mathrm{b}, \alpha)$ is a convex function we have that the maximum margin separator is unique, since local minimum of convex functions are global minimums. The sum of 2 convex functions is convex if its summation terms are convex. $\frac{1}{2}\|\mathrm{w}\|^2$ is convex since its first derivative set to zero gives a single solution $w = 0$ which is a a global maximum since the second derivative is equal to a positive 1. The second term $\sum_{i=1}^{n} \alpha_i [1 - y_i(\mathrm{w}^T x_i + \mathrm{b})]$ is a convex in terms of the w and b since it is a linear combination of such variables. Since both summation terms are convex we can conclude that $\mathcal{L}(\mathrm{w}, \mathrm{b}, \alpha)$ is convex in w and b and that the maximum margin classifier is unique.

Since the minimum for the Lagrangian is unique we can conclude that if the new margin classifier g^- produces a larger margin than g, then it would be the solution w as seen in equation 13. However this contradicts the fact that the difference in the training set for the two cases is a NSV, since given that all the terms in equation 13 involve SV only g and g^- should remain the same. For the case where a new point is introduced that does not violate the current margin then we have that for such point x the condition $y_i(w^T x + b) = 1$ does not hold and therefore the α for such x must be 0. This point is threfore NSV and will not change the maximum margin classifier when introduced as seen in equation 13.

2.1 Part b

Part a showed that if the classifier w is a maximum margin classifier then it does not change for changes in points that are not support vectors. Since the predictions are of the form $y = sign(w^T x + b)$ with $x \in \mathcal{R}^d$, we can find a unique classifier w by solving the system $Y = sign(\mathrm{W}^T X)$. From part a we found that only SV modify the prediction. Therefor in such equation X is the matrices of support vectors with the additional dimension for the b term in the sum $X \in \mathcal{R}^{(d+1)\times n}$. The classifying vectors w' include the b term by being defined as $w' = b, w_1, w_2, \dots, w_d$. The equivalence of this 2 equations can be observed from the terms in the matrix multiplication:

$$y = sign(w^T x + b) = sign\left(\begin{bmatrix} b & w_1 & w_2 & \dots & w_d \end{bmatrix} \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{d,1} & x_{d,2} & \dots & x_{d,n} \end{bmatrix} \right) \tag{14}$$

For this solution to exist and be unique we need X to be a full column rank matrix. This implies the system does not have free variables. For this to be satisfied we must have that $n \leq d + 1$. Since n has been defined as the number of support vectors that change the prediction w, we can conclude that only $d+1$ of such vectors can change the prediction.

2.2 Part c

The eross validation error has been proven in the notes to be the unbiased estimator of the test error. This test error R for the case where we are performing leave one out cross validation in a set of n is given as:

$$\mathbb{E}_{\mathcal{D}}[R] = R_{cv} = \frac{1}{n} \sum_{k=1}^{n} \mathbb{I}\big(y, sign(w_k^T x_k)\big) \tag{15}$$

where x_k is the point left out with the additional 0th dimension and hence $x_k \in \mathcal{R}^{d+1}$ and w_k is the maximum margin separator trained on the set obtained by removing the x_k point. Since the classifier w_k is only defined by the support vectors, as seen in part a this w_k will be the same as long as x_k is not a support vector. For all this cases since the data is linearly separable and the prediction is the same for the full training set and the leave one out cross validation set the left out point is correctly predicted. When the point left out is a support vector w_k changes and the left out point might be misclassified. Hence:

$$\mathbb{E}_{\mathcal{D}}[R] = R_{cv} = \frac{1}{n} \sum_{x_k \in SV} \mathbb{I}\big(y, sign(w_k^T x_k)\big) + \frac{1}{n} \sum_{x_k \in NSV} 0 \leq \frac{1}{n} \sum_{x_k \in SV} 1 = \frac{|SV|}{n} \tag{16}$$

The number of support vectors $|SV|$ has already been bounded to be smaller than $d + 1$. Moreover this equation is the expected test error for a training set of $n - 1$, since leave one out cross validation was performed. Therefore for a randomly sampled training set of n we have that the test error $R < \frac{d+1}{n+1}$

3 Problem 4

3.1 Part a

A support vector machine (SVM) with Gaussian (RBF) kernel learns a large margin separator by applying a non-linear transform without having to compute the non-linear transform for each training point. This can be done by finding a kernel which computes the dot product of two data points **x** and **x'** of dimension *d*. In the case of the RBF kernel used this is computed as:

$$K(x, x') = e^{-\gamma \|x - x'\|^2} \tag{17}$$

This requires the parameter γ to be chosen. This can be done through cross validation to choose the γ that minimizes the expectation of the test error, without having to use the set reserved for testing. Since in Assignment 1 it was determined that this data is linearly separable, a hard margin SVM can be used. In order to perform such training and cross validation the Python library `sklearn.svm` can be used. This tools do not include however a hard margin separator. However, the soft margin trainer class SVC for C-Support Vector Classification can be used by choosing the appropriate extra C parameter. This parameter defines the penalty for the error in classification allowed. A high C would therefore prevent any point to be misclassified and would achieve the same result as a hard margin separator. Setting C to any value greater than or equal to the default value of 1 was found to provide the same test error, and was therefore concluded that any value above 1 would ensure correct separation of the data for this particular training set.

The library also provides a function to perform k-fold cross validation. Giving this function the SVC class set with the RBF kernel and a certain γ parameter returns the errors after training the class on the appropriate subsets of the provided training set and testing with the data left out for testing in each k-fold run. Figure 1 shows the average error of the 30-fold cross validation for each gamma. The graph also shows the minimum cross validation error found to be 0.6% when gamma is 0.012. Training the classifier on the full training set with this gamma value results in an expected training error of 0% since the data is linearly separable, and in a test error of 3.57%. This is a substantial improvement with respect to the 4.9% error obtained by the perceptron algorithm in assignment 1. In assignment 1 the benefits in the test error of a large margin in the separator where explored. Since the SVM maximizes the margin it can outperform the perceptron algorithm which just obtains a separator without considering the margin.

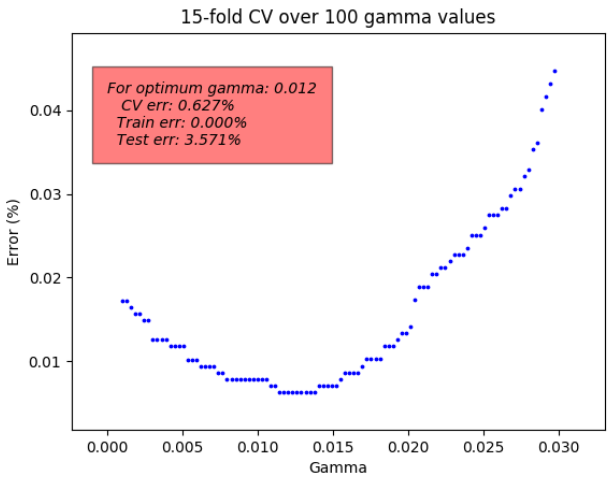


Figure 1: Cross validation for gamma parameter of SVC with RBF kernel

3.2 Part b

The principal component directions of our data can be determined using the tool `sklearn.decomposition.PCA`, which produces the desired number of component directions. Since having reduced our data to a lower dimensional space might have made it not linearly separable, a soft margin SVM is required. This can be achieved by setting the already mentioned C parameter in the SVC trainer to penalize the allowed errors. How much such errors must be penalized must therefore be chosen without using the test set. The cross validation was therefore performed not only on γ but also on C.

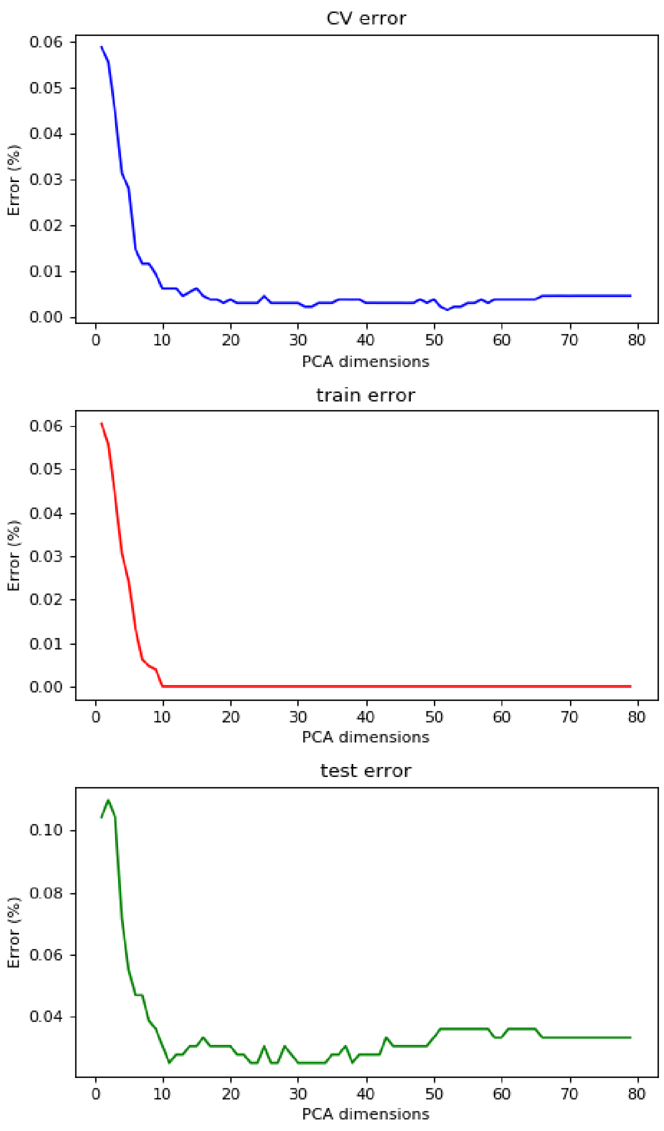


Figure 2: Errors in training and testing for different k number of PCA dimensions with cross validation over γ and C

The cross validation, training and test errors for using PCA to obtain k dimensions can be observed in Figure 2. As k increases the errors tend towards the observed values in part a with the original 256 dimensions. This can be explained by the relevant number of support vectors.

Problem 2b proves that there can only be $d + 1$ support vectors such that the maximum margin classifier changes if the support vector is removed, where d is the dimensions of the training data. Although the train samples have k dimensions, the RBF kernel is equivalent to computing the dot product of the infinite dimension non-linear transformation of two data points as can be observed in its Taylors series expansion:

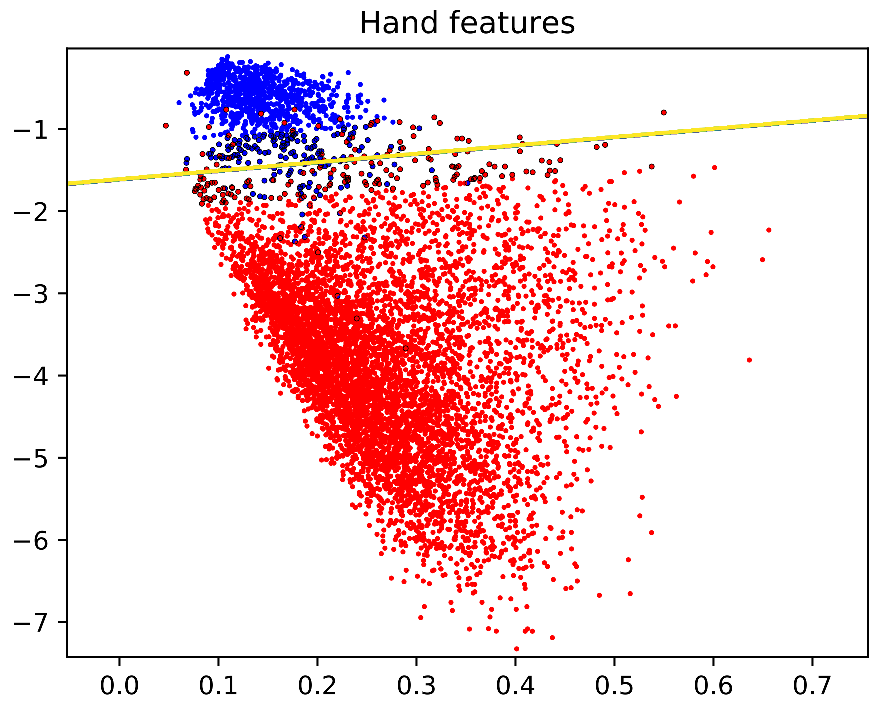
$$K(x,x') = e^{-x^2} e^{-x'^2} \sum_{n=0}^{\infty} \frac{2^n x^n x'^n}{n!} \tag{18}$$

Although there are non-zero terms for infinite dimensions, such terms decay as the dimension increases. As this happens the terms can be considered insignificant. This sets a limit in the maximum number of relevant support vectors $d + 1$. We can therefore conclude that for values of k were this number of relevant support vectors has been achieved, more training data would not improve the expected test error. Since the cross validation error is the expectation of the real test error, this conclusion can also be obtained during training.

3.3 Part c

The training methods applied in part b can be used to classify the digit 1 from the rest of the digits as seen in 3. As in part b, a soft margin SVM has been trained on non linearly separable data by cross validating the γ and C parameters. This was performed on both 2-dimensional features provided and on those obtained through PCA.

This training results in a test error of 2.09% for the handcrafted features and of 0.89% for the PCA obtained features. The better performance of PCA is also reflected in the graph by a large presence of support vectors in the handcrafted classifier, since the expected test error is proportional to the number of support vectors.



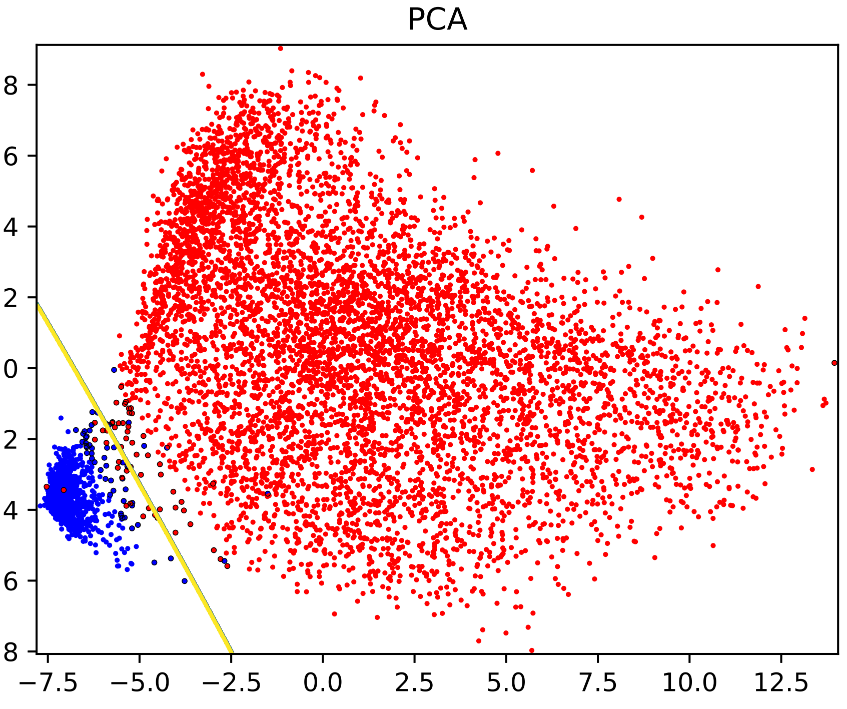


Figure 3: RBF kernel soft margin SVM on 2-dimensional data of digit 1 (blue) and the rest of the digits (red) obtained through PCA (top) and the handcrafted features (bottom). Support vectors' data points are circled in black