Linear and Gaussian Classification of Data and Tikhonov Regularization of a Signal

Part A) Linear Classification Aggregated Data

Abstract

Purpose: The aim behind the first section of the assignment is to see how classifying data can be done with a Support Vector Machine (SVM). This is a powerful method in data analytics which provides a separating hyperplane for the data. For part A a linear classifier will be used.

Methods: As many of the algorithms and techniques in use for this part of the assignment have been extensively covered and self-implemented in the past, MATLAB's built-in functions are explored instead and the possibilities they provide to those interested in data analysis and machine learning.

The functions of interest are:

- $fitcsvm$
- $crossval$
- $kfoldloss$
- $perfcurve$
- $predict$
- $confusionmat$

Prior to performing the analysis it is necessary to reduce the dimensionality of the dataset as the problem begins as a 777-dimensional problem. Dimensionality reduction is accomplished by Principal Component Analysis (PCA).

Results: The MATLAB script, "a4q1_20051918.m", produces two figures, the accuracy of the classification along with the corresponding confusion matrix. The figures show the separating hyperplane and ROC curve. These are found in further detail in the **Results** section.

Discussion: Evidently with a linear classifier it is difficult to perfectly classify data as it can't fully account for outliers which will induce misclassifications. This is a large reason for any inaccuracy produced by the test sets. Although in the tests using cross validation the loss from 5-fold, the accuracy was still very high, approximately 94%.

This classification method produces an accuracy of 93.35%. This vastly exceeds what a random classification should do as mentioned in the assignment statement for this question. Findings are explored in-depth in the corresponding section.

Introduction

Classification is an ever-growing topic in data analysis and encompasses many important techniques. In this question, the problem is to predict if an institute is labelled private or public. For programming purposes these labels are switched to -1 or +1 respectively. This dataset is from "An Introduction to Statistical Learning" By James et al., 2013.

The act of classification is reliant on finding some way to best divide the data so for when predicting future data the chance of a correct classification is higher. There exist multiple techniques for classification but for this question an SVM is implemented and used.

SVM separates the two classes of data by finding two support vectors which maximize the margin of separation. As obvious misclassifications, or non-linearly separable data, will cause issues with the SVM the SVM introduces slackness as a way of managing these misclassifications. Slackness provides means of stopping misclassifications from being support vectors and overtly influencing the model.

Methods

The MATLAB script begins by first loading in the dataset. As it is a csv file the first row and column are discarded.

The data is then separated to two variables, X which contains the independent data and $yvec$ which contains the labels. X is scaled by 0.001 given per the professor's instructions so that it works with the plotting functions provided.

As data becomes easier to classify when it has less dimensions, the pca function from MATLAB is called. As it is suggested to work in 2-D, the first two columns from the PCA are selected. The first principal component accounts for as much of the variability of the data as possible and the second as much of the remaining variability as possible.

$$X_2 = X \cdot pca(:,1:2);$$

Now with reduced data an SVM model with a linear classifier is fitted. This uses the MATLAB function $fitcsvm$ with the following parameters:

$$linmodel = fitcsvm(X_2, \vec{y}, 'Standardize', true, 'Solver', 'L1QP');$$

This fits a linear SVM classifier using the dimensionally reduced data with labels given by $\vec{y}$. Furthermore, the data is standardized, and also indicated to the function call is that L1 soft-margin minimization by quadratic programming is the the solver to be implemented. Then by the returned values contained in the structure $linmodel$, the SVM hyperplane $\vec{w}$ can be obtained.

Next, the accuracy of this model is computed. This provides an insightful measure to how well the SVM is performing. Pulling from MATLAB's extensive selection of built-in functions this is easily accomplished. Calling the $crossval$ function a 5-fold cross-validation model is implemented for testing.

$$cvmodel = crossval(linmodel, 'kfold', 5);$$

Then with the kfoldloss function the accuracy can be computed.

$$loss = kfoldloss(cvmodel);$$

$$acc = 100 - loss * 100;$$

Note that loss is given as a decimal less than one, so it has to be multiplied by 100 to obtain accuracy as a percentage.

As the model created is trained on 2-D data rather than the original 17-D data, it is necessary to see how this model will classify the labels for each institution from the 2-D data. For this process the $predict$ function available in MATLAB is useful. This will generate predicted labels for each institution as well as scores.

$$[lbls, scores] = predict(linmodel, X_2)$$

$lbls$ are, as mentioned, the predicted class labels given by the classification of the data. $scores$ is a two column-vector matrix of classification scores that MATLAB routinely computes, the following line shows the equation MATLAB implements. This is similar to that seen in lecture.

$$z(\vec{x}_j) = \sum_{i \in \mathbb{N}_s} \left(\alpha_i y_j \kappa(\vec{x}_i, \vec{x}_j)\right) + b$$

The second column vector is the negation of the first. Graphically this is that one plots 1-Sensitivity versus Specificity and the latter plots the reverse. For the plotting function given the latter is used thus…

$$scores = scores(:,2);$$

With the newly obtained predicted labels the confusion matrix is computed using the $confusionmat$ function of MATLAB. As the confusion matrices viewed in lecture differ in order from MATLAB's default an additional parameter 'Order' will be passed to correct for this slight difference.

$$cmat = confusionamt(\vec{y}, lbls, 'Order', [1\ -1]);$$

This confusion matrix can be found in the **Results** section.

Lastly, the Receiver Operating Characteristic (ROC) curve is found then plotted. Found along with the ROC curve is the Area Under the Curve(AUC), a measure of how well a binary classification model operates. These can be obtained using the $perfcurve$ function of MATLAB.

$$[xroc, yroc, \sim, auc] = perfcurve(\vec{y}, scores, 1);$$

The variables used to plot are $xroc$ and $yroc$.

Results

The accuracy obtained through 5-fold cross validation and the use of the $kfoldloss$ MATLAB function is 93.56%. (This will change due to the randomness of cross validation.)

Table 1: Displays the confusion matrix for the predicted labels.

True Positive (TP)	False Negative (FN)
550	15
False Positive (FP)	True Negative (TN)
32	180
Accuracy	
93.95%	

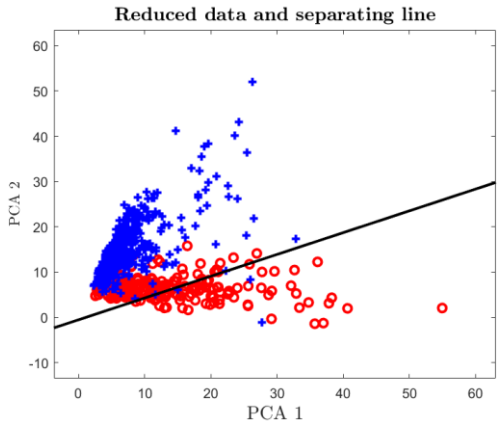

Figure 1: Displays the SVM hyperplane overlaid on the reduced data. This is the hyperplane for 'Standardize' set to true.

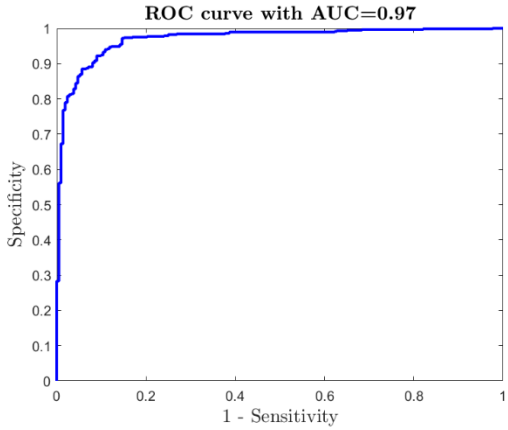

Figure 2: The ROC curve generated using the $perfcurve$ function in MATLAB.

Discussion

Through the use of numerous built-in MATLAB functions we were able to build a classification model using SVM that outperforms a random classification, that is as there are 777 total institutes and 565 are private so a random classification of an institute being private should be correct approximately 73% of the time. Obviously from the confusion matrix, *Table 1*, in the **Results** section our model beats this with a 21.23% increase in classification accuracy. As the 5-fold cross validation model takes random subsets its actual accuracy will slightly change each run of the script. I found it to be approximately the same as the accuracy given by the confusion matrix.

This increase is due to the use of SVM in particular a linear SVM and the reduction of data through Principal Component Analysis. PCA allows us to reduce our data to 2-D where SVM can be more effective as it attempts to find a hyperplane which best separates the data based on a number of criteria. Although this is possible for higher dimensional data such as 3-D and so on, it becomes more difficult to visualize the results hence why we've chosen to work in 2-D. *Fig. 1* shows the data in 2-D with the resulting hyperplane.

In *Fig. 2* the computed ROC curve, a plot of the true positive rate versus the false positive rate, is shown along with the AUC. In the context of classification the AUC is representative of how capable a model is at distinguishing between two classes. In our case the AUC is 0.97 (97%). This makes sense as it corresponds well to the calculated accuracies from the confusion matrix and 5-fold cross validation.

The confusion matrix is how I decided whether I should standardize the data or not. Without standardizing the confusion matrix ends up containing more false positives and negatives indicating slightly less clarity in the model. This stood out to me as an important reason to standardize the data even if the difference is small in this case.

Part B) Gaussian Classification of Small Data

Abstract

Purpose: The purpose of question two is to use an unscaled Gaussian kernel with the SVM in an attempt to properly classify a small dataset.

Methods: To accomplish this goal, the dimensionality of the data has to be reduced this will be done with Singular Value Decomposition (SVD). Following that a hyperplane will be found with an SVM that implements an unscaled Gaussian Kernel, from this hyperplane the data will be separated, and the classifications will be predicted for the reduced data.

Similar to the previous question, built-in functions of MATLAB will be exploited, specifically:

- $fitcsvm$
- $predict$

Results: Shown in the figure appended in the relevant section it is clear the SVM with an unscaled Gaussian Kernel performs exceptionally well after the data has been reduced to 2-D.

Discussion: As seen in **Results** there are no misclassifications and so the classification ratio is 100%. This just highlights the power of non-linear hyperplanes in classification and will covered further in this section.

Introduction

For data that is easily separable just not linearly, a linear classifier immediately provides complications. This is where kernels and soft-margin techniques such as the Gaussian kernel provide beneficial results. As well they continue to work in N dimensions due to a particular property of their Taylor Series expansion, that is it has infinite terms.

The problem pertains to a 5-D dataset. This can be reduced to 2-D using SVD and taking the dot product of the first two columns of both the resulting decomposed U and S matrices.

Once reduced, a classification prediction routine is easily implementable, similar to the way it was done in Part A).

This question provides insight into how kernels can be used with classification methods and the boost they provide to non-linearly separable data.

Methods

The script for this question is found in the MATLAB file a4q2_20051918.m. As usual the script begins by loading in the dataset and separating the independent data from the label vector.

Immediately the built-in SVD function of MATLAB is called on the independent data.

$$[U, S, V] = svd(X, 'econ');$$

$$X_2 = U(:, 1:2) \cdot S(:, 1:2)^T;$$

$'econ'$ returns a size reduced decomposition. What SVD essentially does is finds the singular values of a matrix, in our case X, and orders the singular values from greatest to least where the first two singular values correspond to data vectors which contain the most information about the labels. These two are selected for the 2-D data.

With the data reduced, it is standardized then scaled by a factor of ten. The scale factor lets the data be plotted using the plot functions provided by the instructor.

The skeleton script provides a conditional branch with a flag that dictates whether a linear classifier will be used or a Gaussian kernel. For this question the flag is set to true (1) as the focus is on the effect of using a Gaussian kernel.

The $fitcsvm$ function in MATLAB allows the use of an SVM with an unscaled Gaussian kernel. This call looks like…

$$svmClassStruct = fitcsvm(X_2, \vec{y}, 'KernelFunction', 'RBF', \ldots$$

$$'Standardize', false, 'KernelScale', 1, 'Solver', 'L1QP');$$

After obtaining the SVM structure, the script makes a call to the $svmgausscheck$ function that was supplied by the instructor albeit missing a few lines that were completed. This function uses a loop to compute scores and then compares against their actual values. After all these comparisons the classification ratio is computed which is:

$$Classification\ Ratio = \frac{Properly\ Classified}{Total}$$

This ratio is then printed to the console but is also included in a statement in the **Results** section.

Results

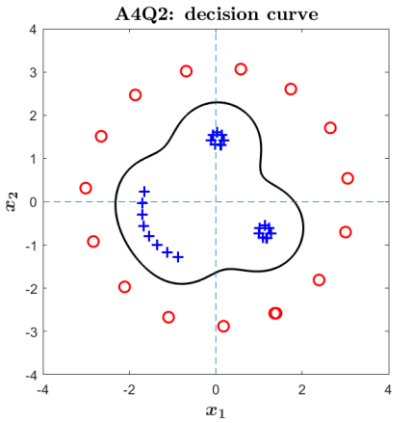

Figure 3: Blue crosses (-1) and red circles (+1) are 5-D data that was reduced to 2-D. The black line is the hyperplane given by a Gaussian SVM which attempts to classify the data.

Clearly the classification ratio is 100%.

Discussion

From *Fig. 3* it is evident that the Gaussian kernel is superior to a linear classifier as it is capable of handling non-linearly separable data. The SVM with a Gaussian kernel had a 100% classification ratio this further supports the ability of a Gaussian kernel.

Question two also shows once again, the perks of using dimensionality reduction in classification problems. The first time running the script provided by the instructor immediately results in data that seems non-separable. After reducing to 2-D (selecting the two largest singular values) the data becomes separable as the plot makes a smiley face.

As a Gaussian SVM allows for soft-margin hyperplanes the 2-D data is easily separated by a circle-like boundary in between the two classes. This provides a simple visualization which helps understand the different capabilities between a linear and Gaussian classifier. This idea can now easily be transferred to more complicated problems that may appear in the future.

Part C) Tikhonov Regularization of a 1-D Audio Signal

Abstract

Purpose: In question three the goal is to use Tikhonov Regularization and its hyperparameter λ to improve the quality of a provided audio clip.

Methods: Aside from Tikhonov Regularization, for this question it is necessary to form a bidiagonal matrix and explore ways of finding a suitable hyperparameter λ. To find a suitable hyperparameter a trial and error approach was used to first gather some insight to the effect λ has on the audio. As it is difficult to score denoising of audio the hyperparameter is selected off intuition and testing.

To create a bidiagonal matrix of sufficient size for the 1-D signal would require a surplus of memory which is not available. Therefore a sparse matrix must be used to be memory efficient. Similarly a sparse identity matrix is required. In MATLAB this can be accomplished with:

- $spdiags$
- $speye$

Results: The lambda value selected for optimal noise reduction is 52.

Discussion: With the optimally chosen λ value it is clear it has some effect on the regularized audio. Whether it truly denoises it is difficult to tell but some of the background static does seem to be mitigated at a trade-off of the instructor's voice sounding slightly muffled. The best balance of reducing these two disturbances was attempted to be found.

Introduction

For the final question the problem is how Tikhonov Regularization can be used for reducing noise in a 1-D audio signal. This requires finding an optimal vector $\vec{w}$ which is dependant on the value of λ.

This poses an interesting situation whereas without background information on the problem and no input, λ must be found through other means.

The noise in the audio is a Gaussian-distributed random variate. This appears as a static noise in the background of the audio.

Audio denoising is a popular topic in both academia and industry. It has many applications and the methods used translate to similar problems. This question will explore some of these methods and provide insight to this type of topic.

Methods

The script for this question is found a4q3_20051918.m, it begins by reading in the audio file and assigning it to an array variable $svec$. $svec$ is then passed along with the lambda value to the $totvar$ function.

The $totvar$ function first constructs the regularization matrix R which is bidiagonal and has form…

$$R = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ 0 & 0 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \cdots & 1 & -1 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \text{ or}$$

$$\sum_{i=1}^{n} (w_{i+1} - w_i)^2 = \|R\vec{w}\|^2$$

With R formed, solving for $\vec{w}$ is straightforward and given by…

$$\vec{w} = [I + \lambda R^T R]^{-1}\vec{y}$$

This regularized vector is returned to the main function where the original audio sample is played proceeded by the regularized audio.

In order to find the optimal λ value for denoising the audio signal a first approach of trial and error was used to get an idea of the effect λ has on the denoising. Once numerous λ values were tested, and a suitable range was discovered the λ values were varied with smaller increments. This allowed for a fine-tuned λ value to be selected based on a personal assessment.

Results

The final λ to be selected was…

$$\lambda = 52$$

Discussion

The first point from this question to discuss is the selection of the λ value. As the default was 1 my first instinct was to check $\lambda < 1$ to ensure these values were non-sensical. This idea was correct as it either had no effect on the audio if not making it worse.

Following this I wanted to find some upper bound to λ. To do this I checked $\lambda = 1000$, obviously when listening to the regularized recording this removes the background static, we hear but at the same time makes the voice sound muffled and very quiet. This provides a reasonable range to work within.

After finicking with λ I finally got it to a value, 52 that I thought provided the best blend of reduced background noise and not too noticeable of a muffling effect to the voice.

In all, this question has helped demonstrate how Tikhonov Regularization with a bidiagonal matrix R and hyperparameter λ can give way to reducing noise in a 1-D audio signal, noticeably improving the quality.