AMATH 482: Home Work 2

Avinash Joshi
bwaseous@uw.edu
AMATH 482 Wi 22
11 February, 2022

Abstract

Classifying objects and testing algorithmic success are integral parts of modern data analysis. This assignment focuses on training a bi-numeral classifier based on the famous handwritten numbers of the MNIST data set and testing the classifier's success rate in classifying pairs of numbers - (1,8), (3,8), and (2,7). In order to build the classifier, the structure of the training set was investigated using Principal Component Analysis and properties of the Frobenius norm. Using sixteen (16) of the PCA modes, Figure 1 resulted in more than 90% of data variance retention, Table 1. Then, by projecting sets of extracted number pairs onto the 16 most energy dense PCA modes, ridge regression can then employed to create said classifiers for the number pairs. These classifiers are then tested against the actual labels of the extracted data sets and evaluated using the mean squared error (MSE), the results of which are found in Table 2. The classifiers, however, are rudimentary and have decreased accuracy when the digit pairs have similar geometry, as in pair (3,8).

1 Introduction and Overview

The purpose of this assignment is to train a classifier to distinguish between pairs of handwritten numbers in the MNIST data set - (1,8), (3,8), (2,7) - and then test the classifier on a subset of untouched testing data; thus, two data sets have been provided - the training data set with labels of the appropriate handwritten numbers and the testing data set, which likewise has labels. Each handwritten number is 16 by 16 pixels, 256 pixels in total, with the training set being comprised of 2000 of these labelled numbers and the testing set having only 500. The classifier must be trained on the training set, its accuracy evaluated, and then employed and evaluated on the testing data set.

Before the classifiers are trained, the data is combed through and the pairs of digits and their labels, which are used for training and testing the classifiers, are extracted from the data sets into $X_{(\text{digit 1, digit 2})}$, e.g., $X_{1,8}$, for the pictures and $Y_{(\text{digit 1, digit 2})}$ for their labels. In these bi-numeral classifying sets, called b_{train} or b_{test}, the first number is given the label -1, and the second, +1. In order to train the classifier, Principal Component Analysis must be employed only on the training data set. This decomposes the training data into an orthnormal eigenvector basis which describes the shape[1] of maximal variance of the numbers 0 through 9 with the appropriate scaling eigenvalues. To approximate how much each PCA mode contributes to the variance of the data, an approximation of the Frobenius norm can be computed up to some desired percentage of data preservation. From this, only a select number of PCA modes will be used for generating the classifiers.

This classifier, called $\hat{\beta}$, is generated by first projecting the extracted data, $X_{(\text{digit 1, digit 2})}$, onto a set number of PCA modes that correspond to the highest amount of data variation. This intermediary matrix, A_{train}, condenses the 256 pixels of information into only a select number reducing computational intensity. This new, lower dimension space contains the features most necessary for describing the data set. From here, ridge regression is preformed on this A_{train} and b_{train} in order to find a linear approximation between A and b - the classifier $\hat{\beta}$. With the classifiers generated, the accuracy can be tested by finding the mean squared error (MSE) between the classifier approximating the labels b using the compressed pictures, A, and the actual labels, b. A value close to zero for MSE correlates to a better approximation of the labels and, thus, a more accurate classifier.

2 Theoretical Background

Supervised Learning, the task of predicting and classifying new data given labelled training data, attempts to find a classifier that, given a new data point, x_j, will accurately predict the associated correct label, y_j. Before, the classifier is trained, however, the inherent structure of the data must be investigated in order to cut unnecessary information from the data points.

This data is investigated through Principal Component Analysis (PCA) which decomposes a data matrix, X, into an optimal basis which the different data points can be best represented by. The PCA is computed on the covariance matrix, C_X, - the matrix that defines the variation between the data points - defined by

$$C_X = Q\Lambda Q^T,$$

where Q are the eigenvectors of the associated eigenvalues Λ. This can be further decomposed using Singular Value Decomposition (SVD) which decomposes X into a basis for the column space, U, and row space, V^T, of the matrix along with the eigenvalues, Σ, of eigenvectors U. This decomposition simplifies PCA into

$$C_X = \frac{1}{N-1} U\Sigma^2 U^T$$

where U are called the PCA modes or left singular vectors of X and Σ are the principal components or singular values of X.

[1]The PCA modes of 2D pictorial data have corresponding representations of shape, but other data won't be as intuitively described as such.

From this decomposition, the density of information of the basis can be computed using properties of the Frobenius norm - $\|X\|_F = \sqrt{\sum_{j=0}^{min\{m,n\}} \sigma_j(X)^2}$. This density of information can be determined using n singular values to approximate up to some percentage of the Frobenius norm.

After investigating the dimensionality, or information preservation, of n singular values, the classifier can now be trained. By projecting the training subset data matrix onto the PCA modes, the data matrix is condensed into a lower dimensionality representation of itself, A. Onto A, a column of ones must be appended at the beginning of the lower dimension representation - the reason will be discussed later. By approximating the relationship between A and its labels b[2] (the labels are replaced with either -1 if they correspond to digit 1 or $+1$ if they correspond to digit 2) through a linear relationship, $A\beta = b$ or, in terms of β, $f(x) = \beta_0 + \sum_{j=0}^{N-1} \beta_{j+1} x_j$, the classifier, $\hat{\beta}$, is approximated through ridge-regression. The column of ones of the A matrix was necessary for including the intercept β_0. With ridge-regression, $\hat{\beta} = \min\limits_{\beta \in R^n} \|A\beta - Y\|^2 + \frac{\lambda}{2}\|\beta\|^2$, or more simply,

$$\hat{\beta} = (A^T A + \lambda I)^{-1} A^T Y.$$

The value of λ must be tailored to the data set to prevent overfitting or underfitting.

The classifier, $\hat{\beta}$, can now be used to approximate the N labels, Y, for both the training and testing data sets. To determine the error or accuracy of the classifier, mean squared error (MSE) is employed, where

$$MSE(\hat{\beta}) = \frac{1}{N}\|A\hat{\beta} - Y\|^2.$$

3 Algorithm Implementation and Development

After using "np.load" to load the data into the train and test data sets, the data is further divided into the X and Y matrices for each using ".get('features')" or ".get('labels')" tag. A function called "digitextractor" can be called on the X and Y data matrices with the specified digit pair. "digitextractor" will produce the bi-numeral X, Y, and b. "digitextractor" does this by copying the pictures onto the bi-numeral X matrix if the corresponding label is either digit 1 or digit 2 using a *for* loop. If the current iteration of the *for* loop corresponds to digit 1, a -1 will be placed in most current position of b. If the label corresponds to digit 2, a $+1$ will be placed in the most current position of b instead. This task is repeated for both the training and testing sets with the digits (1,8), (3,8), and (2,7).

To investigate the dimensionality of X, sklearn's "PCA()" is called creating a new instance of the Principal Component Analysis class. sklearn utilizes SVD in order to generate

[2]The b and y matrices will be used interchangeably during ridge regression discussions

U and Σ. To do this, the instance of PCA is called upon the X matrix and then its principal components are extracted using "pca.singular_values_". Next, a function called "fcutoff" can be called using a cutoff value ($0 \leq$ cutoff ≤ 1) and the singular values of the PCA to determine the number of singular values needed to approximate the percent equivalent of the cutoff value of the Frobenius norm. "fcutoff" first calculates the Frobenius norm for the σ's, then, while the Frobenius norm approximation is less than the cutoff value, the function calculates the Frobenius norm approximation using incrementally more singular values. The function stops when the cutoff threshold has been exceeded and prints the number of singular values and the fractional approximation out of 1.

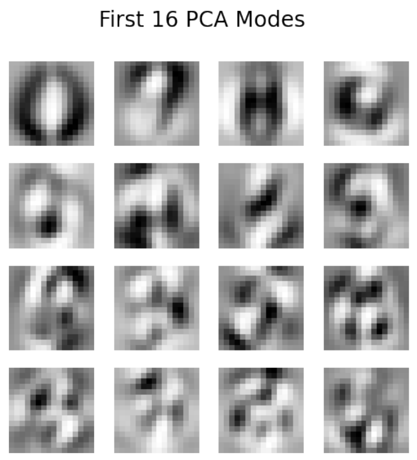

Figure 1: The modes are graphed in decreasing amount of data variance retention, i.e., the first mode captures the highest amount of data variance while the last picture captures the 16th most variation.

With the investigation of dimensionality completed, sklearn's PCA is reinitialized with the parameters "PCA(16)" which only keeps the 16 most variant PCA modes. From the PCA class, "pca.components_" is called to extract the 16 PCA modes and sklearn's ridge regression class, "Ridge(alpha = 1)", is initialized for later use. Using these 16 PCA modes and the extracted bi-numeral X matrix from earlier, the A matrix is generated by multiplying the X with the transpose of the 16 PCA modes, then inserting a column of ones before that. To preform this, the matrices are multiplied together using "@" then the column is inserted using "numpy.insert" along axis one at the zeroth position.

Calling the ridge class with A and b generates $\hat{\beta}$ as described in Section 2. $\hat{\beta}$ is then extracted from the ridge class by "ridge.coef_" and reshaped into a column vector. The first index of the classifier is replaced with the intercept coefficient β_0 with "ridge.intercept_".

With the classifier trained, the labels are then predicted by multiplying A with $\hat{\beta}$. It's accuracy is then tested using sklearn's "mean_squared_error" command with the correct b first and the predicted labels second. The entire process is repeated for all bi-numeral pairs.

4 Computational Results

Using sklearn's PCA to investigate the dimensionality of the pictures resulted in Figure 1. The most variation of the data was found in generating zeros and ones with subsequent PCA modes defining regions or lobes that correspond to or rule out certain numbers. After the first sixteen modes, less and less information is preserved - the results of the investigation are found in Table 1. To retain 60% of the information, only the first three PCA modes are required; for 80%, the first seven; for 90%, the first fourteen. Using sixteen PCA modes retains more than 90% of the data set's variance and information.

Retention (%)	Singular Values (n)	Actual Retention (%)
60.0	3	66.7
80.0	7	81.3
90.0	14	90.5

Table 1: Approximation of the Frobenius Norm using n singular values. There are diminishing returns on the number of singular values used and the increase of actual data retention.

Following the steps outlined in Section 3, the mean squared error for the different pairs of numbers and their training and testing sets are found in Table 2.

Digits	Training MSE	Testing MSE
(1,8)	0.07461129582068486	0.08329273567721955
(3,8)	0.1804095297340279	0.2581845525715514
(2,7)	0.0917899948271311	0.13649452373815243

Table 2: Mean squared error of the classifiers trained on the training data set and tested on both.

5 Summary and Conclusions

By using properties of the Frobenius norm to approximate data retention of the PCA modes, 16 PCA modes were determined to retain more than 90% of the data variance. Other data retention percentages are found in Table 1 and the 16 PCA modes themselves are found in Figure 1. These 16 PCA modes focus on the major regions, spaces, or lobes of the numbers 0 through 9. They are, thus, the most dominant features of the entire data set.

Using a linear approximation between the A matrix and the labels Y and ridge-regression, the classifiers $\hat{\beta}$ for each digit pair were generated and tested. The accuracy of the classifiers, found in Table 2, falls in line with conventional thinking. The Training MSE is closer to 0, a perfect score, given that there are more training points which the classifier is trained on, with the Testing MSE having higher error than the Training for all digit pairs as the classifier has had more training on the larger Training set. The digits (3,8) had the worst performance of the digit pairs as 3, when mirrored on itself, turns into an 8, so the classifier would have a higher chance of misreading the lobes of the 3 as an 8 or the lobes of the 8 as a 3. The digits (2,7) had the second best results but a marked decrease in performance in testing as 2 and 7 share approximately $\frac{2}{3}$ of their features: an upper feature and a diagonal feature. This classifier, however, is better in comparison to (3,8) as the 2 has one more feature which the classifier can distinguish it from the 7. The (1,8) preformed the best as the two numbers share little to no distinct features.

To improve the classifier, Cross-Validation should be employed to empirically find the best λ for training the classifier and different algorithms to compute $\hat{\beta}$ besides ridge-regression should be tested.

6 Acknowledgements

The author would like to graciously thank his classmates for allowing him to discuss the problem with them as a means to help them and test his own understanding of the material.