ML Miniproject 2 - A Processing Pipeline for the Digits Data Set Classifier

Tanasorn Chindasook and Prateek Kumar Choudhary

Abstract

This report discusses the second Machine Learning mini-project, involving the development of a classifier using ridge regression for the Digits dataset. The classifier is fit with various feature sets, including the complete dataset as well as feature vectors generated from the dataset using Principal Component Analysis, K-Means Clustering and Handcrafted features. The efficiency of the classifier is evaluated through a training, test and cross validation errors. The purpose of this documentation is to discuss our exploration of various methods used to optimise the classifier, as well as its results.

I. Ridge Regression for Classification

The implementation of ridge regression for classification requires one to first convert the output label C_i into binary k-dimensional vector z_i called a class indicator vector, k being the total number of classes, such that the vector contains 0s at all the position except i. The problem now becomes of a problem of functional approximation where we wish to find a decision function w which takes a m-dimensional input vector and returns a k-dimensional output vector.

$$w : \mathbb{R}^m \rightarrow \mathbb{R}^k \tag{1}$$

In order for our decision function to give us correct class indicator vector we want to optimize it to minimize the misclassification rate(MISS):

$$MISS = \frac{1}{N}\Sigma_{i=1}^{N} I(d(f_i) \neq z_i) \tag{2}$$

and the Mean Squared Error(MSE):

$$MSE = \frac{1}{N}\Sigma_{i=1}^{N} (d(f_i) - z_i)^2 \tag{3}$$

Thus the optimal decision function W_{opt} would be:

$$W_{opt} = \underset{W \in \mathbb{R}^{k \times m}}{argmin} \frac{1}{N} \Sigma_{i=1}^{N} ||W(f_i) - z_i||^2 \tag{4}$$

where W_{opt} is a $k \times m$ sized matrix of optimal weights [1].
This W_{opt} can be calculated in a simple manner by a procedure known as linear regression, where given a set of training data $X = [x_1, x_2, ..., x_N], x_i \in \mathbb{R}^m$ and their class labels $Y = [Y_1, Y_2, ..., Y_N]$ which can be converted into class indicator vectors $z_i \in \mathbb{R}^k$ we can then calculate W_{opt} as :

$$W_{opt} = (X^T.X)^{-1}.X^T.Z \tag{5}$$

where $Z = [z_1, z_2, ..., z_N], z_i \in \mathbb{R}^k$.
An improved version of this method is called Ridge regression where we add a ridge to the $X^T.X$ matrix in the form of $\alpha^2.I$ where alpha is a scalar variable and I is the identity matrix of size m. Resulting in the new formula for W_{opt}:

$$W_{opt} = (X^T.X + \alpha^2.I)^{-1}.X^T.Z \tag{6}$$

Ridge regression benefits us in two ways:

1) It helps resolve the numerical instability of the $(X^T.X)^{-1}$ when $X^T.X$ is close to singular.
2) It penalizes the regression coefficient terms such that less significant terms converge towards zero faster than more important terms as we optimise the value of α to avoid over fitting. The convergence of these coefficients according to various alpha values can be seen in Figure 1

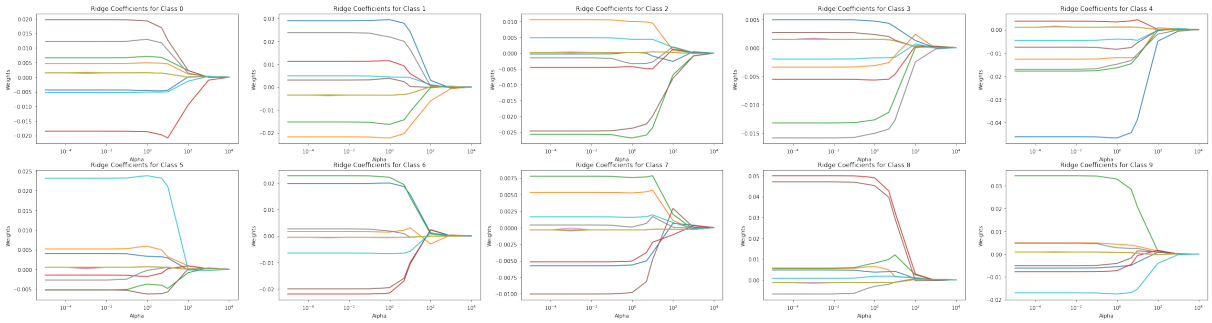

Fig. 1. A visualisation depicting the convergence of the first 10 coefficients for each class. Each coefficient function gets minimised towards 0 at a different rate, denoting its importance in differentiating values for the specific class. Features that converge to 0 slower can be seen as playing a more important role in decision making. The feature set used to create this visualisation is $\mathbf{f}_{(Kmeans_{OPT})}$, which will be discussed later in the paper.

II. Task Introduction

As part of this project we have to use ridge regression to train a classifier on 50% of the data from the Digits dataset for each digit and then test the classifier on the remaining data. In addition to the training and testing split, we also had to come up with our own cross validation implementation in order to better assess model performance. We then had to select the best model and return its parameters, as well as the resulting errors.

A. Data Description

The Digits dataset contains 2000 observations and 10 classes in total. The data is separated so that the first 200 observations belong to the "0" class, the next 200 belong to the "1" class, and so on. Each observation contains 16 x 15 pixel with each pixel having intensity values ranging from 1 to 6, and can be considered as a row vector with 240 dimensions in the format:

$$x \in [0, 6]^{240}$$

Our training set consists of the first 100 observations from each class, 1000 observations in total. Our testing set consists of the other 1000 observations that were not used in the training set.

B. Feature Extraction

As mentioned in the data description above our data is distributed over a 240 dimensional hyperspace, causing the spread of our data points for training to be extremely diluted, this makes it very difficult for learning models to estimate the distribution of input patterns. In order to reduce this complexity we begin by first reducing the 240-dimensional image vectors into lower dimension feature vectors, which are then used to train the classifier. The rule of thumb states that one should have at least 10 times more data points than features [1] . Since we have 1000 observations in the training set, there should be no more than 100 features. Our feature vectors ψ_j will span, at most, a 100-dimensional subspace in $\mathbb{R}^{240}$. The feature extraction techniques that we have used for dimension reduction are principal components analysis(PCA), K-means clustering, hand-designed features and their combinations. The extracted features are then used to fit a ridge regression model and the optimal features are selected using cross-validation. For each selection of features we generate three Mean Squared Error(MSE) and three Misclassification(MISS) error measures to evaluate the effectiveness of the classifiers:

MSE_{train}, MSE_{test}, MSE_{CV}, MISS_{train}, MISS_{test}, MISS_{CV}

The different models are then compared by their errors, and as this is a classification task, the model that minimises the MISS_{CV} is chosen as the final model. We choose MISS_{CV} over MISS_{test} because the testing error is highly dependent on which features are selected in the training and testing sets. Cross validation helps us eliminate this bias in the split by ensuring that every observation appears in the test set once.

1) Hand-Designed Features: We are using image intensity to generate custom features, image intensity are the sum of all pixel values in the image. We generate five intensity values, total image intensity which is the sum of pixel intensity values over the whole image, top half intensity which is the sum of pixel intensity values in just the top half of the image, bottom half intensity, left half intensity and right half intensity which are all calculated similar to top half intensity.

The thought process behind this is that the different intensity values tell us about how an image is filled up. Distribution of the intensities for the digits 0,1,2 and 3 can be seen in Figure 2.

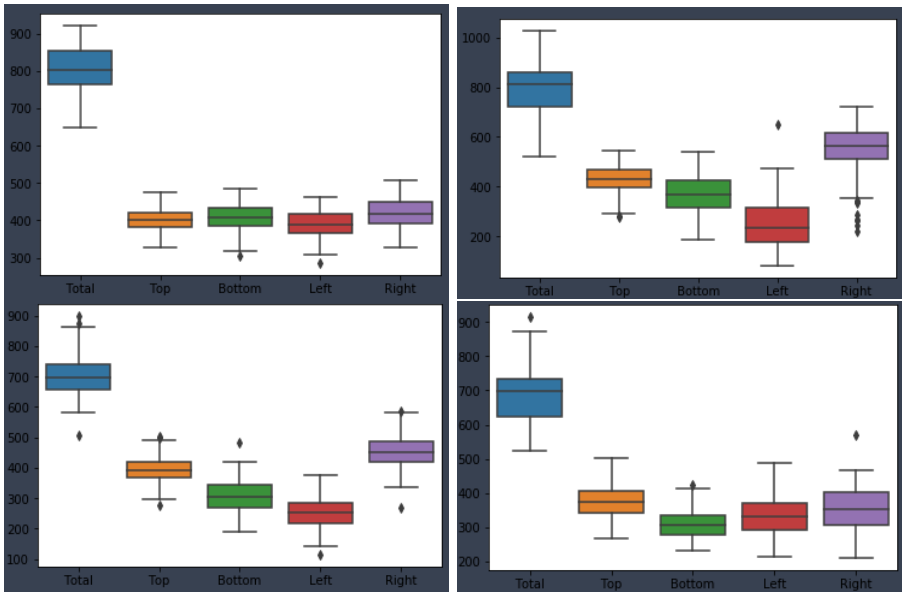

Fig. 2. Spread of intensities for different digits. From the top left going clockwise we have the image intensities for training dataset for digits 0,1,2 and 3 respectively.

2) Principal Components Analysis: In order to implement PCA we first center the data and then compute The principal components through the singular value decomposition of the covariance matrix [2]. Since we have 240 dimensions, our covariance matrix is sized 240×240. The principal components are a set of orthonormal, real eigenvectors of this covariance matrix, and the variances that each princical component captures are the corresponding eigenvalues of these eigenvectors. As we cannot have more principal components than we do dimensions, we can select up to a maximum of 240 possible principal components as features. However, according to the rule of thumb, we will keep our maximum number of principal components to 100.

Dimension reduction using PCA is achieved by choosing the m leading principal components as features, this is done by cross-validation and will be discussed a later section. The variances captured by each additional principal component decreases as shown in Figure 3.

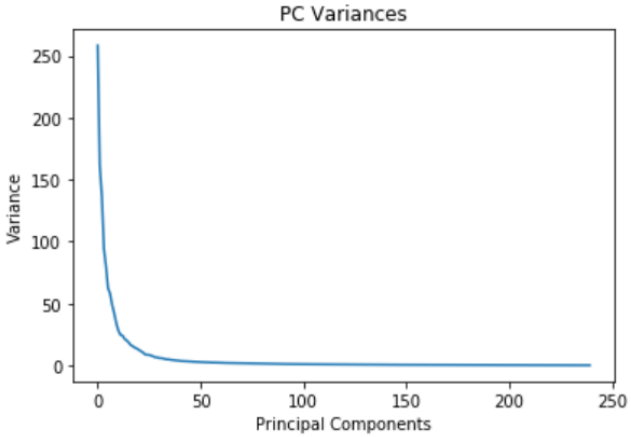

Fig. 3. Variance captured by principal components showing the steep decrease of variance captured in each additional principal component.

Figure 3 clearly shows that after approximately 50 leading principal components, the variance captured by any additional principal component is almost zero, suggesting that the optimal number of principal components will be approximately around 50.

3) K-Means Clustering and Codebook Vectors: Dimension reduction through K-means clustering is achieved by first generating stable set of K centroids for the data cloud which gives us K-codebook vectors. We then compute the

distance of each data point from each of the K codebook vectors [3] . Since we have 1000 data points in our training set, the maximum possible number of K that we can select is 1000. However, again according to the rule of thumb, we will set the maximum number of K for this task to be 100.

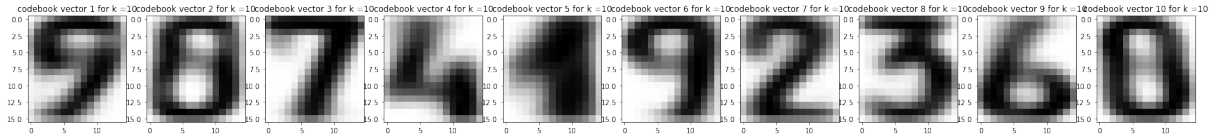

Fig. 4. Codebook Vector Visualisations for K = 10. The visualisation highlights the fact that at K = 10, some of the classes are not well separated.

Figure 4 displays the codebook vectors for $K = 10$, one can clearly see that attempting to assign one cluster per class is insufficient as some of the digits seem to be mixed up, therefore an optimal K will most certainly be larger than 10.

C. Implementation

Our linear regression model adds a bias vector to the existing $N \times m$ dimensional feature matrix, creating a new feature matrix ϕ sized $N \times (m + 1)$. The predictions are then generated by the dot product of the test feature matrix with W_{opt}, resulting in a hypothesis vector containing the degree of "belief" for each class. This hypothesis vector is then converted into a probability vector using the softmax function.

1) Softmax: The softmax function takes two parameters, the hypothesis vector and the α parameter which determines its "sharpness". The effect of the alpha parameter on the softmax function for one predicted probability vector is demonstrated in Figure 5.

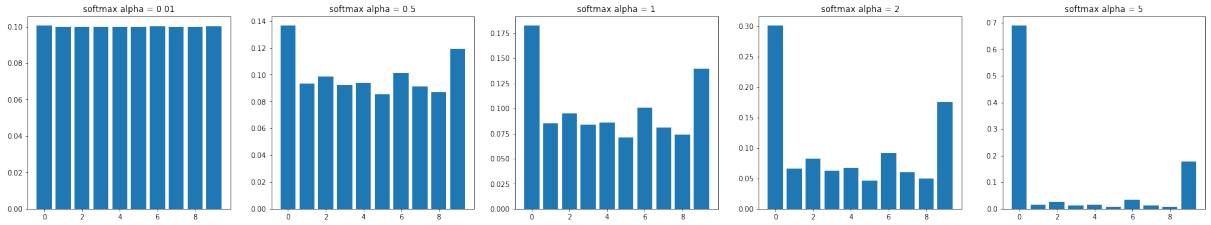

Fig. 5. Visualisations of the probability vectors for the first test observation belonging to class "0". When alpha is close to 0, the probabilities exhibit an almost uniform-like distribution, and when alpha is large, the differences in the distribution are magnified.

It should be noted that the alpha parameter does not affect the final results as the final prediction for each observation is made towards the class with the highest probability. Therefore, it does not matter whether this probability is significantly higher or slightly higher.

D. Model Fitting

We have decided to systematically explore five different kinds of feature sets. Our first feature set $\mathbf{f}_{(raw)}$ is modelling using the raw data as given, this was done to help us establish a base line for what we should aim to improve on. Our second feature set $\mathbf{f}_{(PCA)}$ comprises of features derived from performing principal component analysis on the data. Our third feature set $\mathbf{f}_{(Kmeans)}$ is created by computing the distance of each data point from each of the K codebook vectors. The fourth and fifth feature set $\mathbf{f}_{(PCA+F)}$ and $\mathbf{f}_{(Kmeans+F)}$ consists of the optimal number of chosen features from the previous two sets combined with the five extra hand designed features.

The approach evaluates optimal combinations of α and the number of features selected simultaneously for $\mathbf{f}_{(PCA)}$ and $\mathbf{f}_{(Kmeans)}$, but optimises α for $\mathbf{f}_{(raw)}$, $\mathbf{f}_{(PCA+F)}$ and $\mathbf{f}_{(Kmeans+F)}$. The process is implemented in the following way in order to prevent over fitting.

E. Cross Validation

We are using k-fold cross validation as our cross validation strategy, in order to implement this, the full data set is shuffled and then k-folds are generated to hold the indices of the shuffled data. Due to the way that the data is structured, shuffling the full set of data before dividing the train test split is very important as we could end up having certain folds with only

one type of digit for a larger k if we did not shuffle the data beforehand.

The algorithms loops through the k folds, leaving the i^{th} fold out for testing and trains the model on the remaining data for that loop. The predictions from the trained model on the test observations are then evaluated using MSE and the misclassification rate. The MSEs and MISSs are then averaged across all the k-folds and these are then the overall error rates for the currently selected features or parameter values.

F. Feature Selection

Using the aforementioned validation scheme, optimal feature selection is performed on the features obtained through K-means clustering and principal components analysis to select the optimal number of K and principal components to use respectively.

The effect on the model error subject to the increase of number of features used in a simple linear regression model $(\alpha = 0)$ can be seen in Figures 6 and 7.

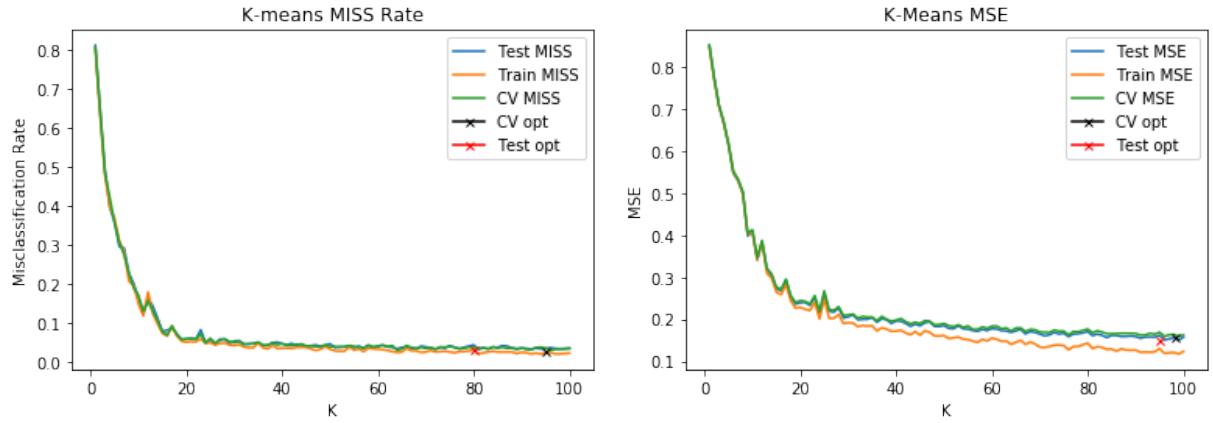

Fig. 6. Misclassification Rates (left) and MSEs (right) for a linear regression fitted model, with features obtained through K-means clustering.

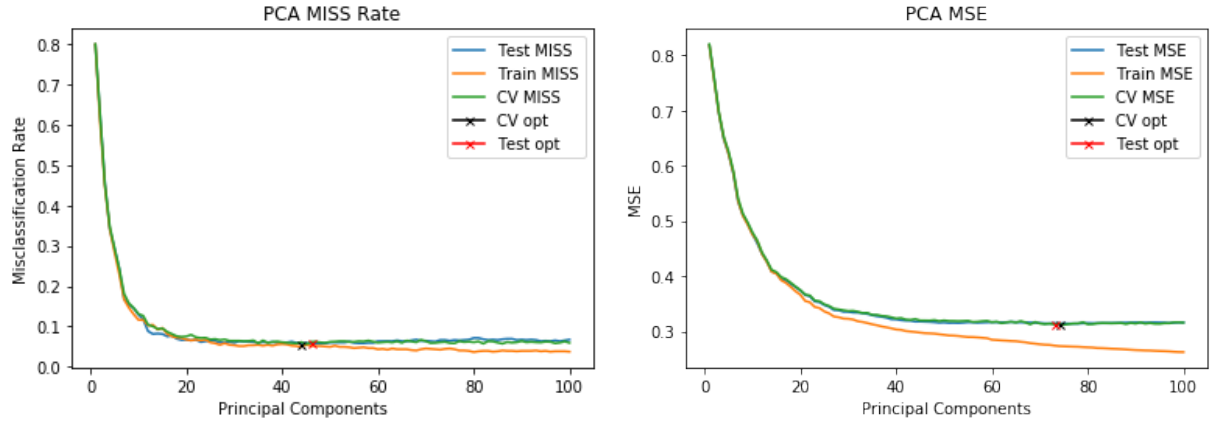

Fig. 7. Misclassification Rates (left) and MSEs (right) for a linear regression fitted model, with features obtained through PCA.

It is evident that the training error continues to decrease when one uses more features in the model. However, the test error and the cross validation error for the PCA obtained features reach a local minimum and begin to increase with the addition of more features from that point. This behaviour is due to over fitting and leads to a common problem faced by machine learners, the bias-variance dilemma. By selecting an optimal number of features as opposed to simply using all possible features, over fitting is prevented.

An interesting observation is that the test error and cross validation error for the features obtained through K-means clustering

continues exhibit a decreasing behaviour even at the maximum number of features, suggesting that a selection of K larger than 100 may yield even more improvements in the results.

In addition to optimising the number of features, one can also optimise the α parameter to further improve the results of the model. The α parameter as explained in Section 1, penalizes the features which are less useful in calculating the class as compared to the more useful parameters, hence by choosing an optimal value for α we can reduce overfitting in our model and improve it's generalization. An visual example of how α affects model output for various number of features can be seen in Figures 8 and 9.

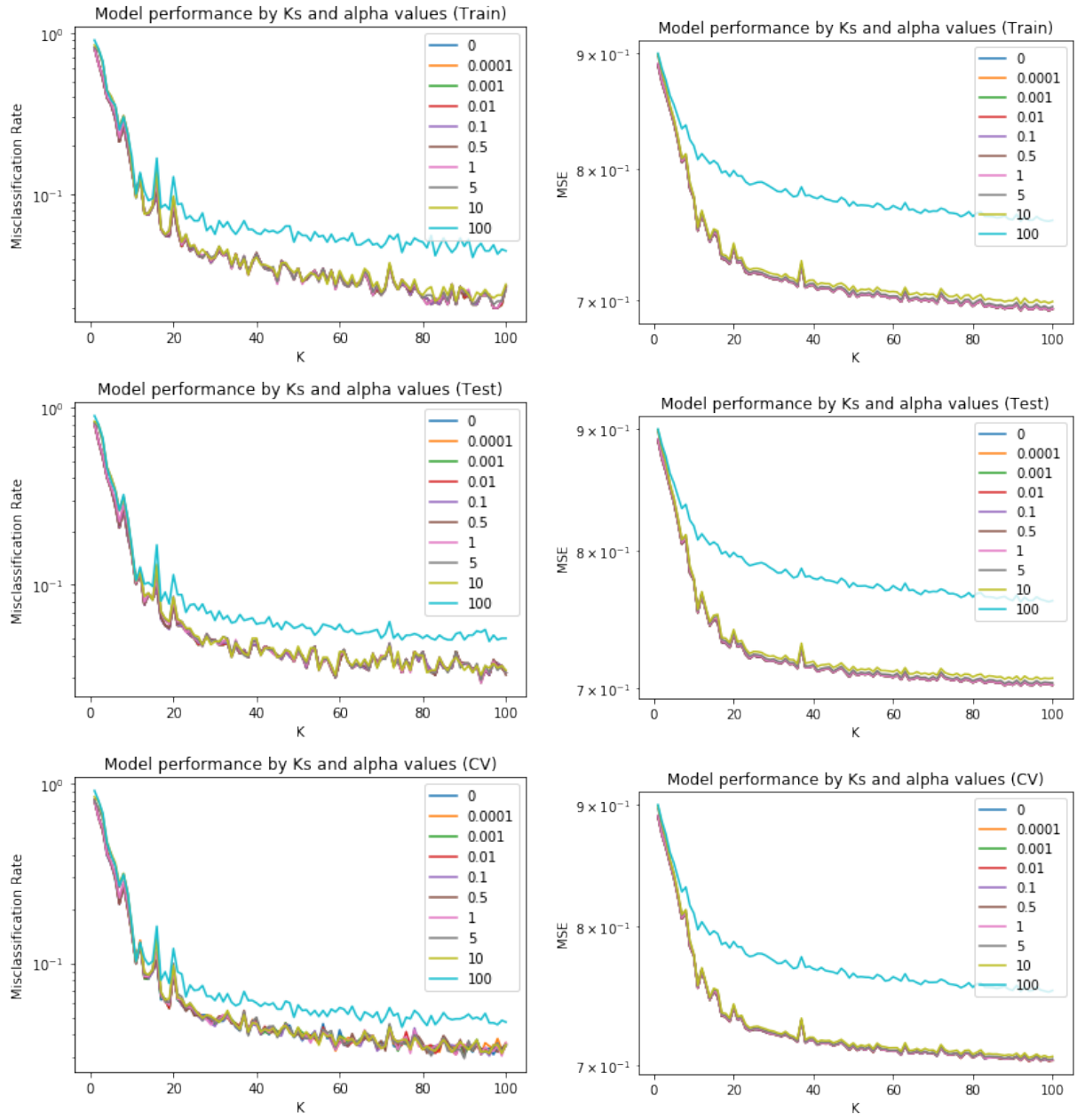

Fig. 8. A comparision of various misclassification Rates (left) and MSEs (right) for ridge regression fitted models, with features obtained through K-means clustering and different values of alpha. The top row depicts the training errors, the middle row the test errors an the bottom row the cross validation errors. It is clear that an extreme value of alpha makes the model much worse.

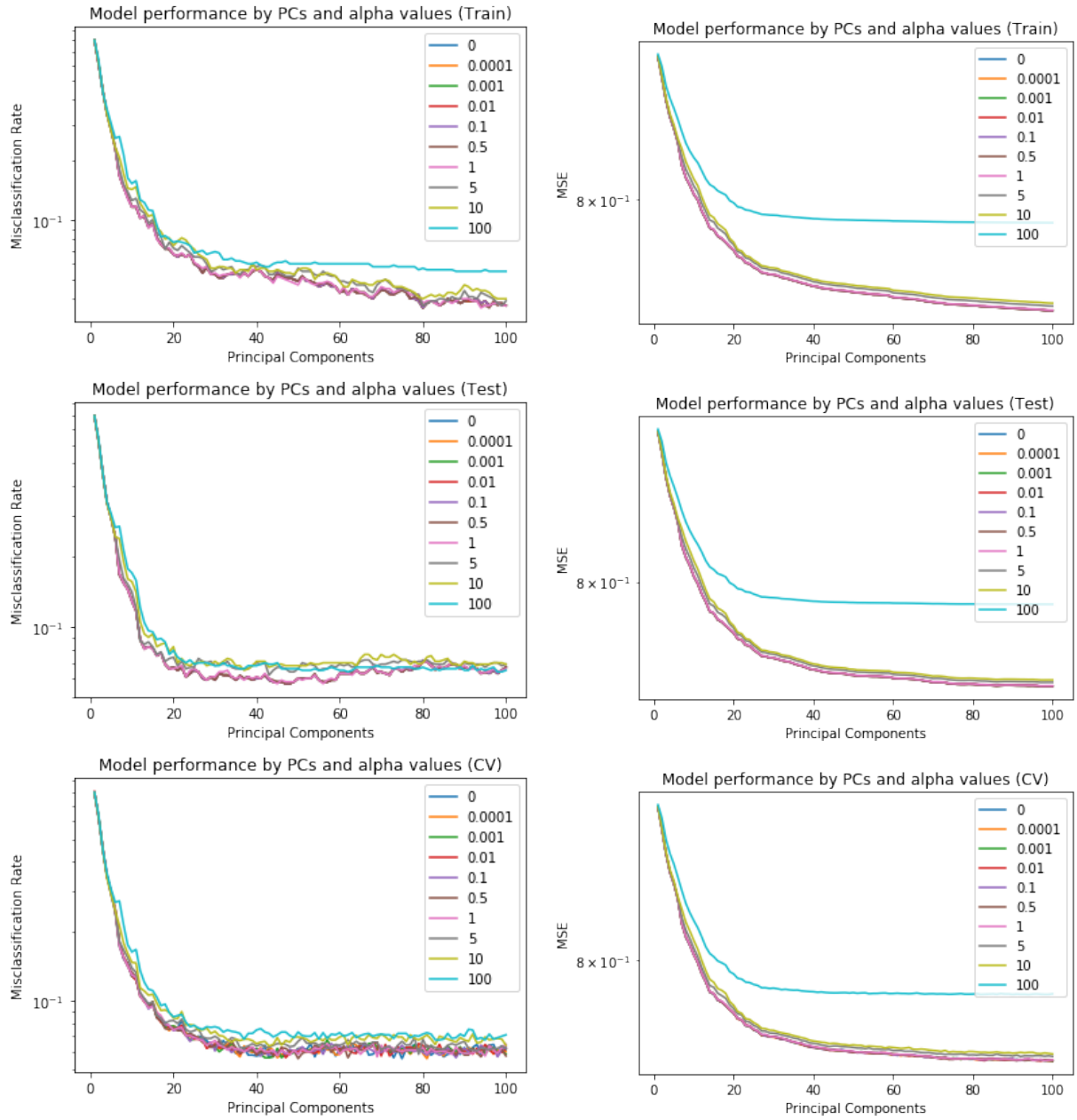

Fig. 9. A comparision of various misclassification Rates (left) and MSEs (right) for ridge regression fitted models, with features obtained through PCA and different values of alpha. The top row depicts the training errors, the middle row the test errors an the bottom row the cross validation errors. It is clear that an extreme value of alpha makes the model much worse.

	alpha	K	cv_MISS	cv_MSE	test_MISS	test_MSE	train_MISS	train_MSE
0	0	97	0.0310	0.703006	0.029	0.702096	0.020	0.693019
1	0.0001	94	0.0310	0.702631	0.029	0.702096	0.020	0.693019
2	0.001	90	0.0310	0.702809	0.029	0.702096	0.020	0.693019
3	0.01	88	0.0310	0.702744	0.029	0.702096	0.020	0.693019
4	0.1	96	0.0305	0.703049	0.029	0.702090	0.020	0.693033
5	0.5	81	0.0315	0.702863	0.028	0.702065	0.020	0.693138
6	1	99	0.0305	0.702815	0.028	0.702090	0.020	0.693213
7	5	90	0.0295	0.703460	0.029	0.703110	0.021	0.694544
8	10	96	0.0315	0.705049	0.031	0.705700	0.022	0.697801
9	100	75	0.0440	0.751143	0.048	0.761200	0.041	0.758879
10	1000	99	0.1330	0.858966	0.140	0.868167	0.145	0.867722
11	10000	74	0.6165	0.899123	0.504	0.899398	0.498	0.899389

	alpha	PCs	cv_MISS	cv_MSE	test_MISS	test_MSE	train_MISS	train_MSE
0	0	75	0.0560	0.749706	0.057	0.746394	0.036	0.740557
1	0.0001	81	0.0560	0.749454	0.057	0.746394	0.036	0.740557
2	0.001	42	0.0565	0.749576	0.057	0.746394	0.036	0.740557
3	0.01	47	0.0565	0.749555	0.057	0.746394	0.036	0.740557
4	0.1	88	0.0565	0.749695	0.057	0.746396	0.036	0.740560
5	0.5	50	0.0560	0.749626	0.057	0.746450	0.036	0.740627
6	1	69	0.0570	0.749671	0.057	0.746605	0.036	0.740819
7	5	79	0.0585	0.751737	0.062	0.748303	0.037	0.742922
8	10	43	0.0635	0.752941	0.065	0.749489	0.040	0.744430
9	100	84	0.0675	0.782549	0.064	0.788087	0.055	0.787241
10	1000	81	0.1100	0.886437	0.108	0.890427	0.104	0.890401
11	10000	63	0.3000	0.899831	0.250	0.899888	0.250	0.899887

Fig. 10. A comparison of best models per feature combination (K-means on the left, PCA on the right)

III. Results and Conclusions

After our systematic approach of optimising the features for modelling, the results for the best number of principal components to use is 50 PCs with $\alpha = 0.5$. The optimal number of K to use is $K = 90$ with $\alpha = 5$. Finally, after running and comparing all the combinations of features we have achieved the following best errors per feature set:

Model	MISS Error	MSE Error
$\mathbf{f}_{(raw)}$	0.6596	0.7460
$\mathbf{f}_{(PCA)}$	0.0560	0.7495
$\mathbf{f}_{(PCA+F)}$	0.0575	0.7504
$\mathbf{f}_{(Kmeans)}$	0.0295	0.7026
$\mathbf{f}_{(Kmeans+F)}$	0.0355	0.7044

The cross validation errors displayed in the table above are visualised as a function of regularisation in Figure 11.

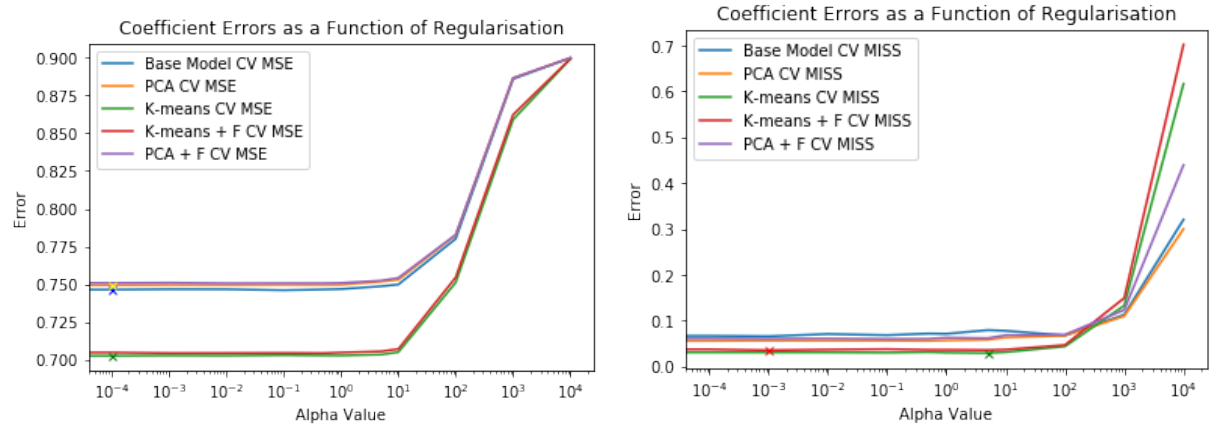

Fig. 11. Final comparision of different combinations of features and their MSE and MISS error rates as a function of regularisation. The best model overall is shown to be the model comprised of K-means created features only.

In conclusion we can see that K means by itself is the best feature set for classification of the Digits dataset with a misclassification rate of 2.95% followed closely by K means and handmade features(3.55%). The handmade features seem to actually detract from the accuracy of both K means and PCA, hence our final classification model reduces the initial 240 dimentional input vector to a 90 dimenstion feature vector by calculating 90 codebook vectors and then calculating the distances between the input and each of the codebook vectors and uses that to classify the digits.

References

[1] Jaeger, H. (2019). "Machine Learning Lecture Notes".
[2] Pearson, K. (1901). "On lines and planes of closest fit to systems of points in space", Philosophical Magazine, Series 6, vol. 2, no. 11, pp. 559-572.
[3] MacQueen, J. (1967, June). "Some methods for classification and analysis of multivariate observations." In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).