Machine Learning, Spring 2019: Miniproject 2

Konrad Burchardt Margaret Sant

21 April 2019, 2019

1 Introduction

In this report we will go through the steps we took to train our digits classifier. We will start with a brief explanation of the dataset, then we will explain how we preprocessed the data and created our testing and training datasets. After this we will go in detail into our 3 functions we created, K-means for dimension reduction, Linear regression to find our W_{opt} and cross validation to find the right number of k . After this we will explain our optimal number of k and our finding. We will also include different case scenarios with different parameters.

2 The Digits Dataset

The input patterns for our main dataset are 15 x 16 grayscale images of hand-written digits 0—9, which were normalized in size and ratio to fill the image space. The dataset is comprised of two-dimensional array of size 15 x 16, making vector x_i lengths of 240. The values of the vector components correspond to the greyscale values [0 *to* 6] The rows consist of 200 instances of each digit, making 200 instances of first "0" images, then "1" images, until "9".

3 Objectives

The objective is to train model that will classify digits patterns based on a sample of hand-written digits. We use vector quantization with K-Means clustering for image reduction, and Linear Regression for our classifier. We need to also search for the optimal hyper parameter k (number of clusters) by incorporating K-Fold Cross Validation on our regressor, repeating for various feature dimensions (various values of k). We need to get a miss classification rate below 5%

4 Data Preprocessing – Creating the Train and Test Set

Our first preprocessing step was to create a column to indicate the class (digit) of our training pattern. We would go from 0 to 9 and change the number every 200 rows. The first 200 would be class = 0 for "0" patterns and so on until we have class = 9 for "9" patterns. Then, we used one-hot encoding to create a binary 10-dimensional target vector z_i, in which a one corresponds to the class of pattern x_i and a zero everywhere else.

Second, we equally divide our main dataset into x^{train} and x^{test} so that 1000 patterns that correspond to 200 of each class belong to both each. This way, our final model is trained and tested equally for each class.

5 Dimension Reduction - Creating functions for feature extraction

For dimension reduction, we use K-Means Clustering to extract a low dimensional feature representation of the data points:

$$x \in \mathbb{R}^n | n = 240 \rightarrow f(x) \in \mathbb{R}^m | m = k$$

This is an unsupervised machine learning technique that finds similarities in the data-points and clusters them based on closeness (Euclidian distance) so each cluster $C_1, \ldots, C_k$ represents a feature. First, an initiation function randomly assigns each data point in the training set to one of the k clusters. The first step of the algorithm re-assigns each data-point to the closest cluster centroid. The second step recalculates the location of the centroids based on the mean of the data-points in the respective clusters. Steps one and two repeat until no reassignment takes place. This equation describes the formula for computing the mean of each cluster:

$$\mu_j = \frac{\sum x \in {s_j}^x}{|S_j|}$$

μ_j s the centroid, aka the codebook vector. The minimal distance between the data-point and centroid is thus calculated as distance $\alpha_i = \|x_i - \mu_j\|$. This is the Euclidian distance. Our K-Means algorithm employs distances as a distance vector on both the train and the test set to obtain the reduced x to our feature vector $f(x) = (a_1 \ldots, a_k)$ which will be fed into our Linear Regression classifier. We later explain how we split x^{train} into validation and training folds to evaluate the models for various numbers of k to find the optimal k_{opt} . We will also use K-Means on x^{test} using k_{opt} for feature reduction before using our final Linear Regression classifier.

6 Linear Regression - Creating functions for Linear Regression

In this stage, we are using Linear Regression as a mechanism to transform the feature vectors $f(x)$ into a hypothesis vector $h(x) \in \mathbb{R}^k$. What we want is a decision function d: $\mathbb{R}^m \to \mathbb{R}^k$ which returns ideal class hypothesis vectors.

We use the machine learning method called Linear Regression as our d decision function to find the optimal weights W_{opt} for our features $f(x_i)$ to create the hypothesis vector, which is an approximate our target z_i, our binary encoding for the digit classes explained above.

To start, we create a feature vector of "1"'s as a padding to deal with bias. So, $f(x_i)$ has size $k+1$, and W_{opt} is a matrix of size 10 x $(1+k)$. Then ,we create our linear regressor with the following equation:

$$W_{opt} = \left(\frac{1}{N}XX' + \alpha^2 I_{nxn}\right)^{-1} \frac{1}{N}XY$$

where X is our $f(x)$, and Y is our binary encoding z_i. For our model, we set α to 0.5. And, N is the dimensions of our raw data. We fit the above equation to x^{train} with $f(x)$ and z_i to create our hypothesis vector $h(x)$, which we will use to test the (1) misclassification $MISS$ and (2) mean squared error MSE scores of our train set using cross validation, described in the following section.

7 Cross Validation

In order to boost our model performance, we run the cross-validation scheme described above for various values of k. We do this to show whether the model is under- or overfitting. We only have x_{train} available to us at this time, so we split the training data set $S = (x_i, y_i)_{(i=,\ldots,N)}$ into two subsets, a $\backslash train$" $T = (x_i, y_i)_{i \in I}$ and a "validation" $V = (x'_j, y'_i)_{i \in I'}$.

This way we train the model and get a training error metric, while also simulating "new data" in $x^{validate}$to get a validation (or test) error metric. We use two different error score metrics, mean-squared-error MSE and misclassification $MISS$ to see how our model performs:

$$MSE_{train} = \frac{1}{N}\sum_{i=1}^{N} \left\| z_i - d(f_i^{train}) \right\|^2$$

$$MSE_{validate} = \frac{1}{N}\sum_{i=1}^{N} \left\| z_i - d(f_i^{validate}) \right\|^2$$

$$MISS_{train} = \frac{1}{N} \left| \left\{ i | maxInd\ d(f_i^{train}) \neq c_i^{train} \right\} \right|$$

$$MISS_{validate} = \frac{1}{N} \left| \left\{ i | maxInd\ d(f_i^{validate}) \neq c_i^{validate} \right\} \right|$$

where d is our linear regression decision function training on the training folds, z_j our given encoded target vector, and c_i is the hypothesis vectors.

8 Using Cross Validation to find optimal K

We use Cross validation on various numbers of k_i, and graph our MSE and $MISS$ results. For our problem, we use 8 folds for our cross validation, results shown on the graphs below:

Given k=[1,10,20,30,40,50,60,70,80,90, 100, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240]:

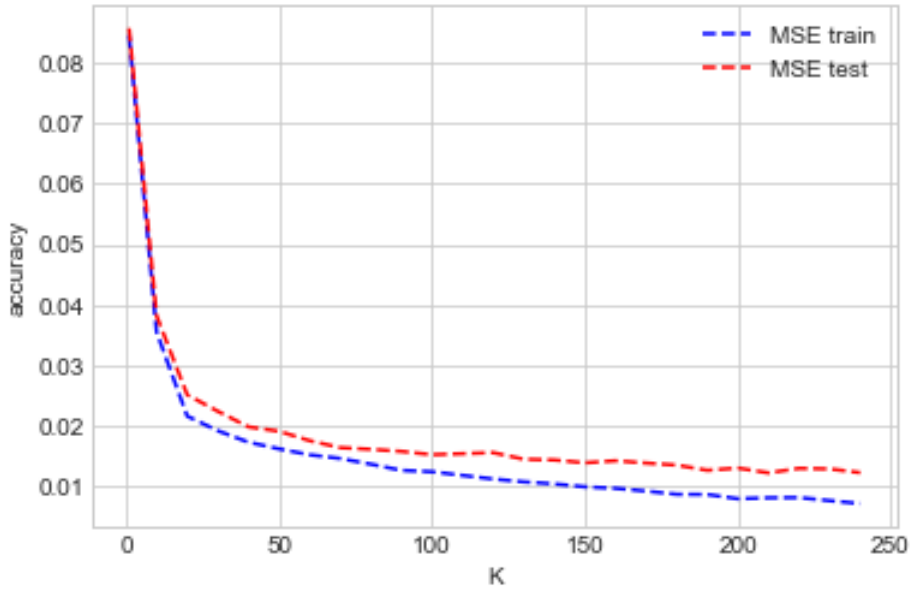

Figure 1: MSE score with 8 folds

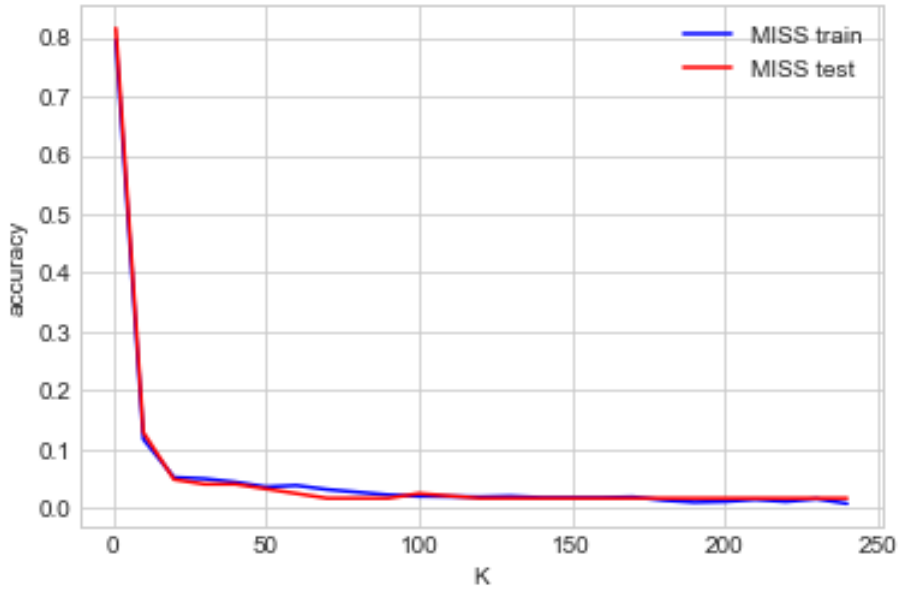

Figure 2: MISS score with 8 folds

The optimal number of clusters is somewhere between 80 and 110. On this range, the MSE and $MISS$ scores begin to level-off. If you look at the $MSE_{validation}$ score on the graph to the right, between 80 and 110 clusters, the line begins to increase and move away from the MSE_{train}score. So, we decided to test our classification model using various numbers of clusters between 80 and 110.

9 Results

To measure the accuracy of our model, we use 1 – $MISS$, which gives us a percentage of correct classification. We ran K-Means for values of k in range 80 to 110 on the entire training set, and then fit our classifier and obtained the following results for the test data:

$Number\ of\ clusters$	MSE^{test}	$MISS^{test}$	$Accuracy$
80	1.6	3.8%	96.2 %
90	1.6	3.6%	96.4 %
100	1.5	2.8%	97.2 %
110	1.5	3.1%	96.8 %

So the best K-Means reduction technique for our linear classifier model uses 100 clusters. This means that we when we reduce our data $x \in \mathbb{R}^{240}| \rightarrow f(x) \in \mathbb{R}^{100}$, we obtain the optimal weights for our linear classifier.

We also wanted to see what number of clusters we would use according to the MISS and MSE results for 2-fold cross validation, as a comparison. Judging from this graph, we would choose our number of clusters from range 60 to 100. We obtained the following results on our test set for these clusters:

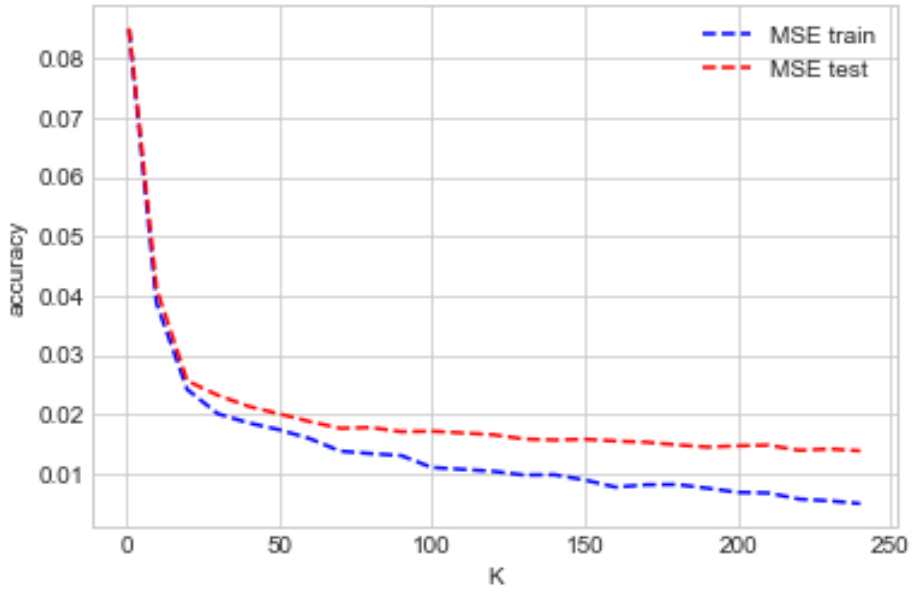

Figure 3: MSE score with 2 folds

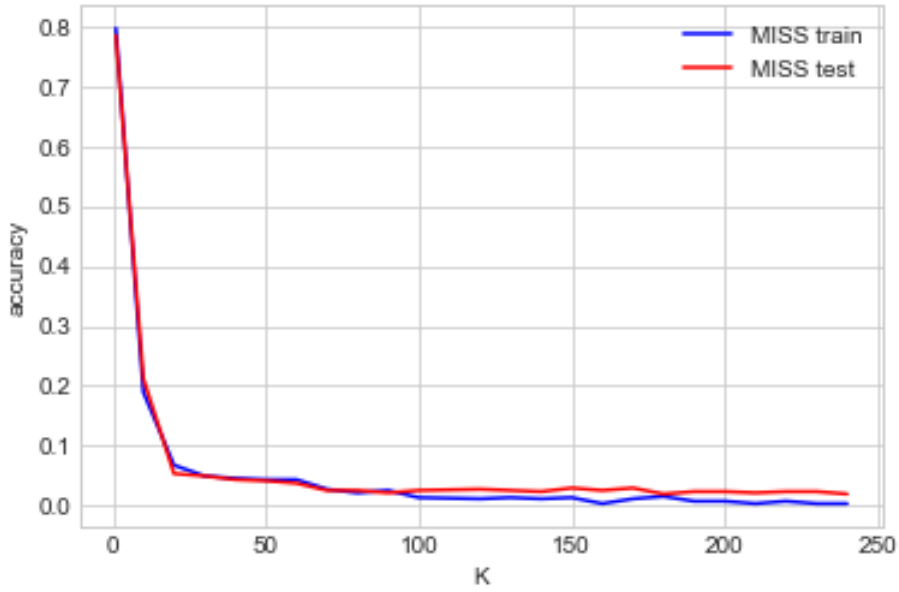

Figure 4: MISS score with 2 folds

Number of clusters	MSE^{test}	$MISS^{test}$	*Accuracy*
60	1.7	3.8%	96.2 %
70	1.7	3.5%	96.5 %
80	1.6	3.6%	96.7 %
100	1.5	3.0%	97 %

100 is also the optimal number of clusters, but it appears are the end of our range. Using 8-folds was ideal for us, because it shows that the accuracy decreases after 100 clusters.