University of Southern Denmark

Statistical Machine Learning F21

Random Forests and KNN

Christian Skov Esbensen
chesb17@student.sdu.dk

Jakob Grøftehauge
jakra17@student.sdu.dk

Handover Date: 31-05-2021

1 Introduction

This exercise will investigate the KNN and random forest algorithm. The algorithms will be applied to the handwritten digit recognition problem, where it is wanted to classify single digits from 0 to 9. The report features a brief summary of the algorithms. Then follows a brief description of the dataset and how it is organised. Afterwards the results of the tests conducted for each algorithm is presented and discussed.

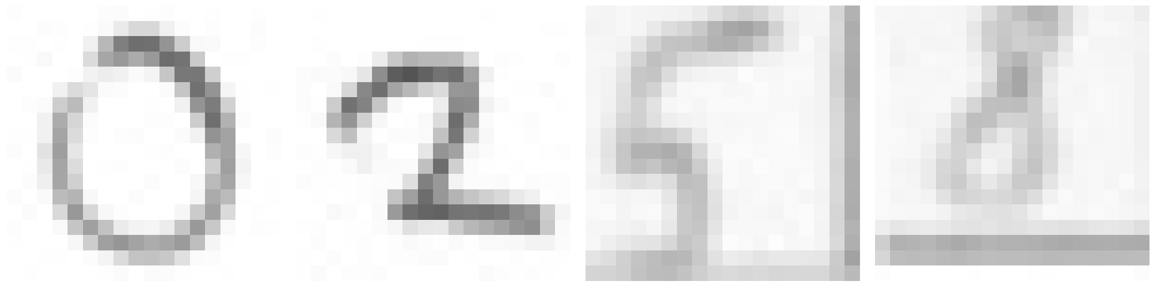
2 Data

The data for this report consist of grayscale images of single handwritten digit form 0 to 9. The complete collection of digits consist of 56000 images, and is made up of data from 38 persons. Each person has supplied 2000 data instances to the complete collection and the instances from a single contributor are uniformly distributed among the 10 digits. A examples of images from the dataset can be seen in figure 1.

To test the performance of the random forest and KNN algorithm a training and test dataset is needed. For the conducted test two different approaches for splitting the complete collection of data is used. For both splits it is tried to achieve a test-train split of 20/80 percent.

The first approach is denoted, *all-persons-in*, and features data from all persons in both the test and training set. It is ensured that each person the same amount of data in the test and training set. This is done by taking a fixed percent of each digit form each person, the used percentage is 21%. This means that 42 images of each digits is taken out for each contributor., to yield the test set.

The second splitting method is denoted, *disjunct*, and characterised by only have data from a single person either in the training or the test set. 8 person has been taken out of the complete collection to form the test set. The remaining 30 persons make up the training set.



(a) (b) (c) (d)

Figure 1: Randomly sampled images from the complete dataset

3 Methods

3.1 KNN

The KNN is a very simple classification algorithm. It works by comparing a data-point in question, to a large collection of previously classified data instances. The collection of previously classified data instances is traversed in order to find the data-instances closest to data point in question. The data-point in question is then classified based on the most frequent class of the K nearest neighbours in the collection of data instances. The distance between data-points can be measured in different ways, and is highly dependent on the application of usage. The KNN algrithm is therefore determined to have the time complexity in Big-O notation for inference [3]:

$$O\left(k \cdot n \cdot d\right), \tag{1}$$

where k is the number of neighbours, n is the number of data points in the training set and d is the dimension of the data, in this case $18^2 = 324$ dimensions.

The general procedure of classifying a data instance can be outlines as follows:

1. Find K record that have similar features, to the data-point in question, in the collection of previously classified data instances.
2. Find out what the majority class is among the set found in 1. Assign this label to the data-point in question.

3.2 Random Forrest

The random forest algorithm utilises an ensemble of binary decision trees. To construct the trees the concept of bagging is used. The bagging algorithms uses a subset of the data to construct the tree. The subset is randomly sampled with replacement form the complete set. This re-sampling is repeated for every tree in the ensemble[4]. The time complexity is determined to be in Big-O notation [3]

$$\text{Training :} \qquad O\left(n \cdot \log\left(n\right) \cdot d \cdot k\right), \tag{2}$$

$$\text{Test :} \qquad O\left(d_t \cdot k\right), \tag{3}$$

where n is the number of points in the data set, $d = 324$ is the dimensionality of the data, k is the number of trees in the forest and d_t is the depth of the decision trees.

In order to understand the concept random forest it is first needed to understand the concept of decision trees. A decision tree basically implements a set of "if-else" rules dividing the data into smaller and smaller subsets based on specific feature values. The tree is made up of several binary nodes. A binary node splits the data into two buckets based on the value of a features. When these binary nodes are combined in a tree structure they make up a binary decision tree[2].

Several methods can be considered to evaluate the performance of a split. In this paper only entropy is considered. Entropy is a measure of chaos in a population. Minimising one of these measures would yield to a high performing tree. A query data-point can be classified from an ensemble of trained decision trees, by letting the decision trees vote on the class that best match the query data-point[1]. Unlike KNN, binary tree is able to discover hidden patterns in the data introduced by complex interactions in the data. Though the challenge exists in partitioning the data well in reasonable time while not over-fitting to the training data.

3.2.1 Principal Component Analysis

For the random forest the PCA algorithm will be explored for prepossessing the data, in an attempt to make it easier for the random forest algorithm to divide the data into cluster of equal class. The PCA algorithm is used to determine the transformation of the data, so that the most variance is expressed along the principal axis. The transformation to determined is given by the eigenvectors of the covariance matrix. The transformation is structured so that the most variance in the data is expressed along the first principal axis, the second most along the second axis and so on. All new principal axis are perpendicular to each other.

The PCA algorithm is not explored for the KNN algorithm as it yields a transformation of data, which preserves the distance between individual data points. This means that it will not make any difference in the accuracy of model. For KNN, PCA is only meaning to full to use for reduction of data dimnesionality, but this will not be explored in this report.

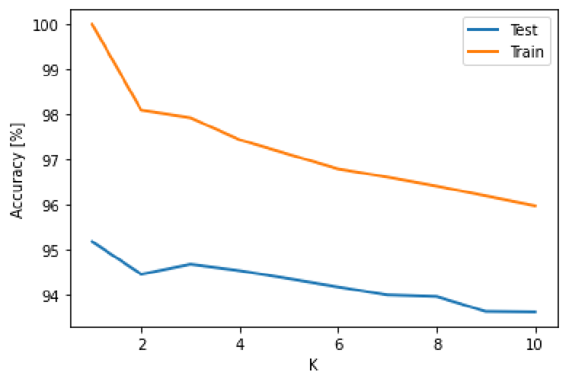
4 Results

In this section the KNN and random forest algorithm will be tested. It is experimented with different preproccesing steps. Furthermore an investigation into the influence of the hyperparameters of each algorithm is conducted.

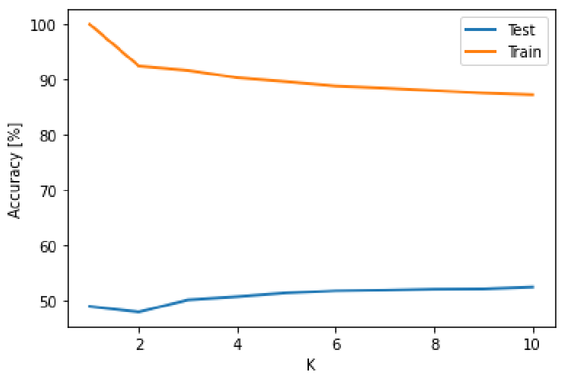
4.1 KNN

Firstly KNN is evaluated on the unmodified data and tested on both the disjunct and all person in datasets. The result for the disjunct dataset can be seen in figure 2a and the result for the all person in dataset can be seen in figure 2b. As seen from the two figures the KNN algorithms is better at generalising to the test set for the all persons in data than the disjunct dataset. This is expected behaviour as is has already seen data from every person and therefore can easier fit the the specific handwriting for each person, rather than generalising to the general description of each digit. From both figures it can be observed that the training set for $K = 1$ achieves an accuracy of 100%. This originates form the fact that the model contains a copy of all data points in the training set, therefore when finding the nearest neighbour in the model, it will always match to a copy of itself, and thereby yield the correct prediction every-time. When examining the two figures, it is evident that the two models suffer from overfitting. Most noticeable is it for the disjunct model, though it is still present for the All-person-in model. The KNN algorithm itself does not provide any way of counteracting this overfitting, therefore it has not been pursued to decrease it, and it will not be investigated further. From figure 2a and 2b it can be seen that the maximum achieved test accuracy for the all-person-in is 95.2% and 66.3% for the disjunct dataset. Both dataset achieved the highest accuracy with a k value of 1.

Next it is wanted to test how it affect performance, if a prepossessing algorithm is applied to the input data. It will be tested how normalisation of data effect the performance of the KNN algorithm. The normalisation of data means that instead of measuring how big the response each input feature is, the magnitude of each data feature can be view as equivalent to measure of high far from average the response is. The images are pixels-wise normalised using the mean and standard deviation found for the training set. Figure 3a and 3b show the performance of the KNN algorithm with normalization of data. As seen in figure 3a



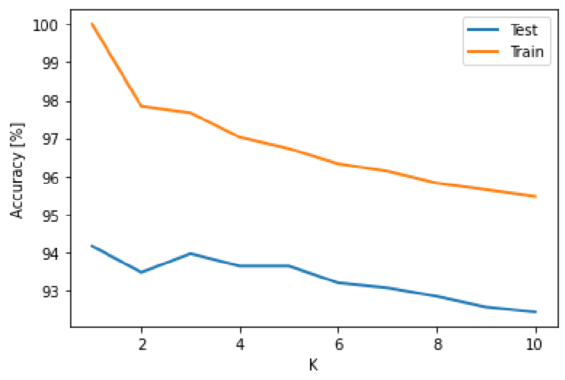
(a) All Person In



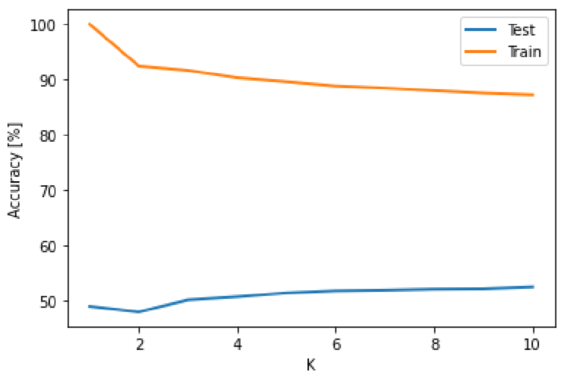
(b) Disjunct

Figure 2: Accuracy curve for both disjunt and all person in dataset.

the maximum achieved test accuracy for the all-person-in data is 94.2% with using $k = 1$. For the disjunct the maximum accuracy is achieved using $k = 10$ and is measured to 52.5%



(a) All Person In

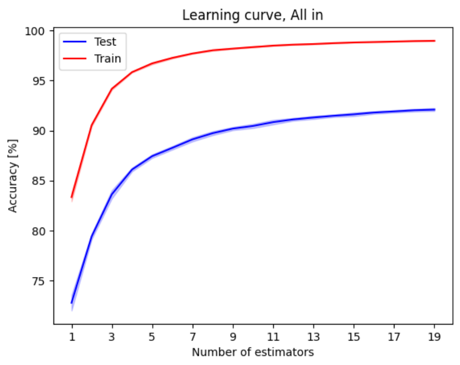


(b) Disjunct

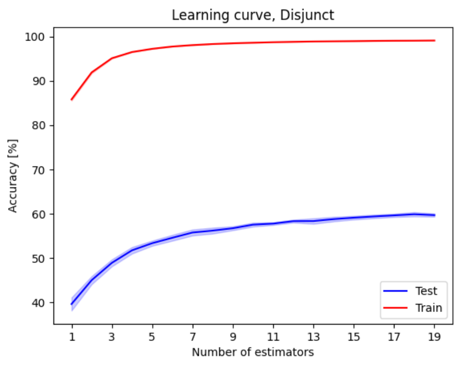
Figure 3: Accuracy curve for both disjunt and all person in dataset with the use of PCA as preprocessing.

4.1.1 Random Forrest

The performance of the Random Forest is evaluated by training the decision trees on both data-sets while varying the number of trees in the forest, from 1 to 19. This test is conducted 10 times for each number of trees in the forest, where the classification accuracy is recorded. The same approach on the principal components of the two data sets. Where the test set transformed to the feature space by the training sets principal component coefficients. The result of the tests without PCA are shown in figure 4 result using the PCA as preprocessing can be seen in figure 5.



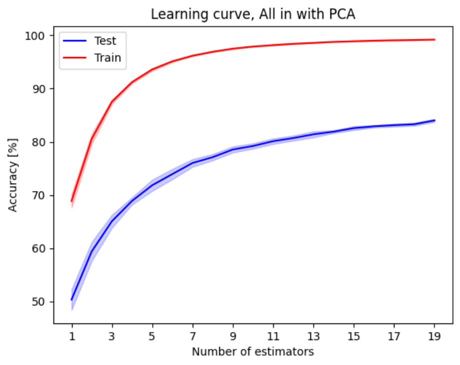
(a) Learning curve for *all-persons-in*



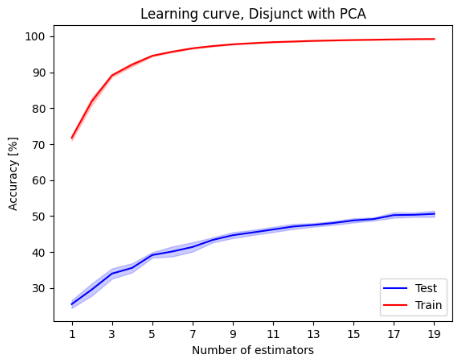
(b) Learning curve for *disjunct* data set

Figure 4: Learning curves for Random Forest applied on both data set

It is seen from figure 4 that both 4a and 4b shows that the random forest obtains a higher accuracy when the number of trees in the forest is increased. Additionally, it is seen that the random forest has a higher accuracy when it is trained and compared using the *All-persons-in* set, where a 91.2 % accuracy were obtained, compared to the 59.5 % accuracy that were obtained from the *disjunct* data set. Furthermore, it is seen that the random forest is prone to overfit for both data sets. Comparing figure 4 to 5 shows a worse performance for the random forest when classifying on Principal Components. The method applied on the *disjunct* data set reaches approximately 50 % accuracy on the test set. It is however seen that the accuracy increases while increasing the number of trees in the forest.



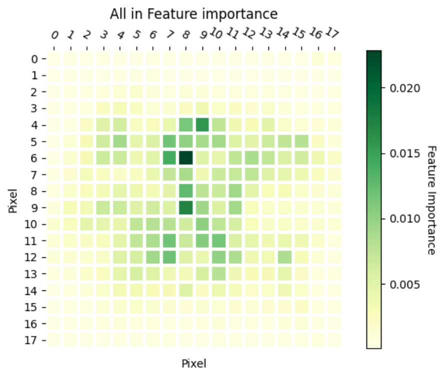
(a) Learning curve for *all-persons-in* with PCA



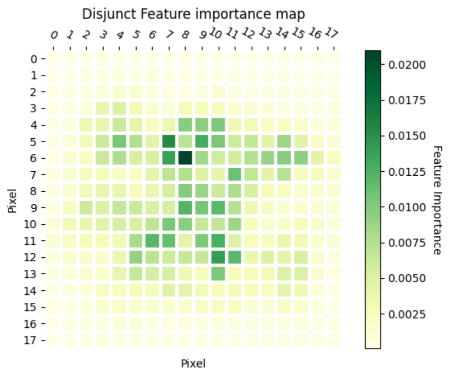
(b) Learning curve for *disjunct* data set with PCA

Figure 5: Learning curves for Random Forest applied on Principal Components of both data set

Figure 4 and 5 shows that the *disjunct* data set obtains a lower accuracy than the *all-persons-in* data set. To determine the importance of each pixels an importance feature map are derived from the random forests. The importance feature maps are shown in figure 6. It is seen that the feature importance in figure 6a is similar to the feature importance in figure 6b, however the importance of the features are different. Additionally, the relative importance of the features are more sparse in figure 6b to the more dense feature map in figure 6a, but both weights the center pixels more important.



(a) Feature importance for All persons in



(b) Feature importance for Disjunct data

Figure 6: Feature importance of for both data sets for a random forest with 40 decision trees

5 Discussion

The data from the individual persons has not been scanned using the same devices. Some digits is digitised using photocopies while others is digitise using digital pens. It is expected that way the data was digitised has an impact on the resulting data. When partitioning the data into the disjunct data set no effort was put into ensuring consistency of the training and test set in terms of how the data was digitised. When examining the result for the disjunct dataset, it can be seen that it consistenly performns worst than the all-person-in, and it is expected that a part of the perofrmance gap could orginates from the potential unconsistency between test and training data. For the all-all-person in it is not expected to be a problem as it features data from all person, and thereby also data from all used digitisation methods.

The decreased performance on the normalised data, could be a result of the normalisation method itself. By normalising pixel-wise across images could result in a noise gain, since multiple pixels in the edge of the picture were unused e.g. gray values close to 0. Additionally, the decreased accuracy of the PCA data, could be caused by the different qualities of the data set. If the training set includes all the hand-scanned pictures and none of these were present in the test set. This would result in a skew PCA transformation on the teat data, and thus resulting in a test set, which didn't completely corresponds to the training set.

6 Conclusion

In this report different strategies for tackling the problem of classifying handwritten digits has been explored. First the KNN algorithms was examined, here we experimented with applying the algorithm to the unprocessed data, but also applying the algorithm to normalised data. It was found that KNN algorithm achieved a relative high performance on the all-persons-in dataset, while achieving a medicore performance of 66.3 % on the disjunct dataset. It was found that applying normalisation to the data did not improve the accuracy of the KNN algorithm.

Secondly the random forest algorithm was used to classify the ciphers, both on the raw data but also the datas principals components. The random forest showed a worse porformance of the PCA transformed data, a suggestion to the cause were examined in the discussion. The random forest achieved a relative high accuracy at 91.2 % for the non-PCA transformed *all-persons-in* data set.

7 Contributions

A detail overview of individual contributions for the report of each group member can be found in table 1. The code written for each method has been produced in a joint effort. Furthermore the review of the complete report has been equally distributed among the two group members

Contributors	Contributions
Christian	Introduction, Method - KNN, Result - Random forest, Discussion
Jakob	Data, Method - Random Forest, Methods - PCA, Results - KNN, Conclusion

Table 1: Individual Contributions.

References

[1] L Breiman. "Random Forests". In: *Machine Learning* 45 (2001), pp. 5–32.

[2] Richard Johnson and Dean Wichern. *Applied Multivariate statistical analysis*. 6th ed. Pearson Education Limited. Chap. 11. ISBN: 978-1-292-02494-3.

[3] Paritosh Kumar. *Computational Complexity of ML Models*. Dec. 14, 2019. URL: `https://medium.com/analytics-vidhya/time-complexity-of-ml-models-4ec39fad2770` (visited on 05/26/2021).

[4] Andre Bruce & Peter Gedeck Peter Bruce. *Practical Statistics for Data Scientists*. second. O'Reilly, pp. 260–270.