A paper on models performance on MNIST dataset

Performance Of Different Image Classification Models with digit recognition

Anders Binder and Joachim Frank

KEA - Københavns Erhvervsakademi

December 16, 2021

Abstract

In this paper we explore the possibility that different image recognition classifiers have distinct levels of performance. We take a look at how several different models perform when handling the MNIST data set downloaded from sklearn. We set up and fit the different classifiers and take a look at their training time, prediction time and their Youden's index, and calculate the average prediction time and Youden's index. We want to explore the effect that PCA may have on the models performance. Look at the possibility that digits may get confused for very specific other digits. Using the data, we create some confusion matrix graphs to get an easier view of how they may confuse digits. We finish the paper by concluding that there indeed is a difference in performance between the models, and how PCA affected their time and Youden's index.

1 Introduction

In the modern world you may come to experience that people like to optimize and automatise certain tasks. One example is that YouTube, one of the biggest video platforms out there, uses AI to recommend its users content that keeps them engaged[1]. Another is for detecting and filtering unwarranted or illegal content from websites[2]. AI is even used in the medical department for delivery of healthcare[3]. So it can be seen that AI is used in a wide variety of fields. In fact, as the amount of content to review and filter through grew exponentially, human review systems have been quickly replaced with AI instead[4].
In this paper we want to explore how some different AI models recognise and classify image data. We hypothesise that the different AI models may be suited for different tasks, and as such we seek to explore the potentially different results they produce and the quality thereof. We will take a look at how good they are at distinguishing images that look similar. We also seek to potentially enhance their training and prediction time through PCA, and see if the trade-off on Youden's index is worth it. We will train these different AI models on the MNIST datasets.
The focus of the paper will be to answer the following question: **How well do different image classification models perform?**
Along the process we would also like to look at the following questions: **Do some digits get more easily confused as certain specific digits more than others?** and **Can model speed be enhanced with PCA reliably?**

2 Methods

From Nunamakers model "multimethological approach to Information Systems research"[5], We will use theory building to develop our questions that will be the focus of our research. The downloading of the AI models we use and the MNIST dataset[6] will all be done in Python using scikit-learn, also called sklearn[7]. We will also be using the libraries matplotlib, numpy and time in the notebook.
The models we will use are:
- Random Forest model.
- MLP model.
- NaiveBase/GaussianNB.
- K-neighbors.
- K-Means

With the tools that Scikit-learn gives us we can make confusion matrices, evaluation metrics (Youden's index) and classifiers.
Youden's index[8] which is a classifier for a models predictive power. We will divide the MNIST dataset into 3 groups: a training-set, a validation-set and a test-set.
We will make confusion matrices to get a look

at what numbers that the different classifiers may confuse the most.
The way that we will calculate the performance of the models is by looking at the time it takes for them to be trained on the training set, make their predictions, and their Youden's index, both before and after PCA applied to see the effect. We will then be comparing the numbers we get. The average of time and Youden's index both follow the math equation: (val1+val2+val3)/3 = average.

3 Analysis

Starting with Theory crafting, as we know that different AI models exist, we believe that they could produce several results that differ when given the same MNIST dataset.
In order to get the results, we will have to train each of the models on the MNIST data set. Each models will trained on the training set and then tested on each of the 3 sets, training set, validation set and test set.
We start up a virtual environment and download the libraries that we are going to use. We startup a python notebook, import sklearn and the required libraries: time, numpy, matplotlib and the MNIST from sklearn. We then import evaluation metrics, the confusion matrix and the confusion matrix display. We import the classifiers(the AI models) we are going to use and the tools for PCA.
We check the dataset to ensure that it contains the expected amount of 70000 digits and 784 features. Having confirmed it, we move onto separating the digits into 3 sets.
We create the variables X and Y, Where X holds an array of the features for each MNIST number, and Y holds an array of the ground truth for them. We set a shuffling index with seed 19 and a permutation of 70000 so that the process can be repeated later with the same shuffled sets, so that if the models are rerun, then one should get nearly the same results.
To ensure that the features converts to the pictures as expected, we will attempt to print one of them out. We convert the X array into type of int and then, Using a random int number within the bounds of the X array, print out the value at that point from the Y array and generate an image of the same point in the x array.
We take the X and Y arrays and split them into 3 different sets: A training set consisting of 50.000 value and a validation and test set each consisting of 10000 digits. Checking the shape of the x array training set, we see that it contains the 50000 variables and still retains the 784 features.
Training the random forest
We start of by giving the random forest classifier 50 estimators, 2 jobs and a random state 19.
We fit the model with the training set, then we have it predict the first 10000 from the same training set. Before running the models we put up timers to see how long it will take for each function to go through. We also take this chance to calculate the Youden's index.
For each of the other models we then do the same. Train them on the training set then have it predict on the training set. Following, we have it make predictions on the validation set. After checking the Youden's index, we then have the model make prediction on the test set. With this we consider the model to be done.
Using the ground truth from the Y array and the outcome of predictions from the test set, we generate a confusion matrix, which we visualise with a graph. These confusion matrix graphs are the findings we can look at to see any patterns for when models confuse digits for other digits.
Training the naive bayes classifier
Training naive bayes runs largely the same as the random forest, we however find it appropriate to make a change in the hyper parameters that naive bayes makes use of due to what we judge to be rather poor performance at this time. We test the naive bayes on the validation and test set as well, getting the Youden's index for each, before creating the confusion matrix and moving onto the next classifier.
The rest of the models are run through the same process as the former one, there however a few differences that we will describe before moving on. All models will be trained on the same sets, tested on all 3 sets, have their training time and prediction time measured. We will also make confusion matrix graphs for each.
Training the KNeighbors classifier: The KNeighbors classifier is made with 5 neighbors, uniform weights and a brute algorithm.
Training MLP classifier: The MLP model is set to have 2 hidden layers, with layer sizes as 75 and 50. The amount of hidden layers is good enough with 2[9].
Using cluster analysis: We use a K-Means cluster analysis model as well. We give it 10 clusters, an init of 10, set the random state to 19, have a max iter of 300 and put the algorithm to auto.
prepare data for principal component analysis: Having gone through all the models and gathered their performance data, we now move on to apply PCA, to see whether it could

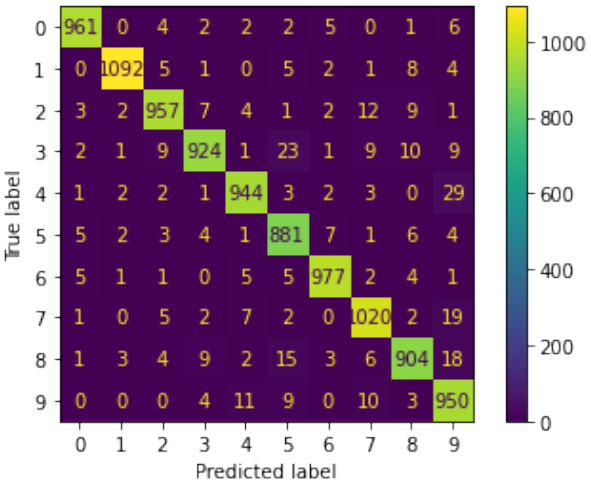
potentially improve upon their training time and prediction time without, affecting the Youden's index too much negatively.
We do this by initializing a standardscaler and fitting it with the X array training set. We now apply the fitted scaler and have it transform the three X arrays: The training set, the validation set and the test set. The scale of features in a dataset affects PCA, therefore we apply the fitted standardscaler to our three sets to ensure all of our features follow the same unit scale[10]. We now create a new PCA object and fit it with the x training set. The PCA object is then used to transform each of the 3 sets. The goal is to reduce the dimentionality of our dataset, which in turn should make it easier to interpret and improve our models speed[10, 11]. Looking at the x array training set we can see that the features has been reduced from 784 to a mere 328.
So for each of the 5 models: random forest, naive bayes, Kneighbors, MLP and K-Means, we train them on the now PCA applied sets, check their time, their Youden's index and finally create a confusion matrix for each. Beside now being PCA applied, every single step is the same to account for consistency. This will make the result easier to compare later.

4 Findings

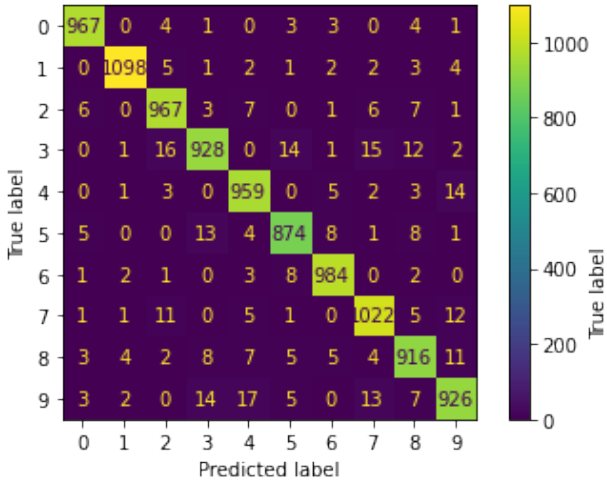
In the analysis section we utilised several classifiers to find their performance data, the data included training time, prediction time for each set and the Youden's index of the models. Here we talk about our findings for these experiments. Values for time are ordered from left to right like this: training set - validation set - test set. We here show our time and Youden's index.
Random forest
Training time: 10.216
Time: 0.135 | 0.152 | 0.154
Index: 1.000 | 0.933 | 0.927
Average Time = 0.147 seconds
Average Youden's index = 0.953
Naive Bayes
Training Time: 0.784
Time: 0.561 | 0.548 | 0.566
Index: 0.595 | 0.593 | 0.603
Average time = 0.558 seconds
Average Youden's index = 0.597
KNeighbors
Training Time: 0.064
Time: 16.929 | 16.747 | 16.031
Index: 0.962 | 0.942 | 0.944
Average time = 16.569 seconds
Average Youden's index = 0.949
MLP
Training Time: 70.950
Time: 0.070 | 0.061 | 0.075
Index: 0.989 | 0.923 | 0.921
Average time = 0.068 seconds
Average Youden's index = 0.944
K-Means
Training Time: 33.175
Time: 0.128 | 0.060 | 0.064
Index: -0.713 | -0.726 | -0.717
Average time = 0.084 seconds
Average Youden's index = -0.718
PCA random forest
Training Time: 28.319
Time: 0.102 | 0.103 | 0.129
Index: 1.000 | 0.862 | 0.851
Average time = 0.111 seconds
Average Youden's index = 0.904
PCA Naives bayes
Training Time: 0.326
Time: 0.234 | 0.219 | 0.219
Index: -0.285 | -0.286 | -0.283
Average time = 0.224 seconds
Average Youden's index = -0.284
PCA KNeighbors
Training Time: 0.064
Time: 12.814 | 11.885 | 12.524
Index: 0.929 | 0.898 | 0.898
Average time = 12.407 seconds
Average Youden's index = 0.908
PCA MLP
Training Time: 18.930
Time: 0.025 | 0.024 | 0.031
Index: 1.000 | 0.937 | 0.937
Average time = 0.026 seconds
Average Youden's index = 0.958
PCA K-Means
Training Time: 24.884
Time: 0.015 | 0.015 | 0.017
Index: -0.817 | -0.813 | -0.814
Average time = 0,015 seconds
Average Youden's index = -0.814
Our findings show that models using the PCA applied dataset had tendency to have shorter training time as well as prediction time. At the same time the models using PCA dataset would also generally have somewhat lower index score. This is to be expected as PCA reduces the features and therefore the data that the models would have to process.
What sticks out here is that PCA did not improve upon Random forest in any capacity; with a longer training time, prediction time, as well as a lower index score. Though it still manages to predict the training set well enough to get an index of 1, both before and after PCA.
Th MLP model only show positive results, with a higher index score as well as shorter time on

training and prediction. This is also withing expectation as MLP is a model built to use ratio data.
Naive bayes however, did get a shorter training and prediction time, but the cut to Youden's index was rather significant, putting it far below it's former index, whereas it was at a rather high point before using PCA.
In terms of performance K-Means by far did the worst getting Youden's indexes close to -1 with and without PCA, and while Youden's index's theoretical range is from -1 to 1, values under 0 won't be meaningful, which means that our K-Means model won't yield any useful results[12].

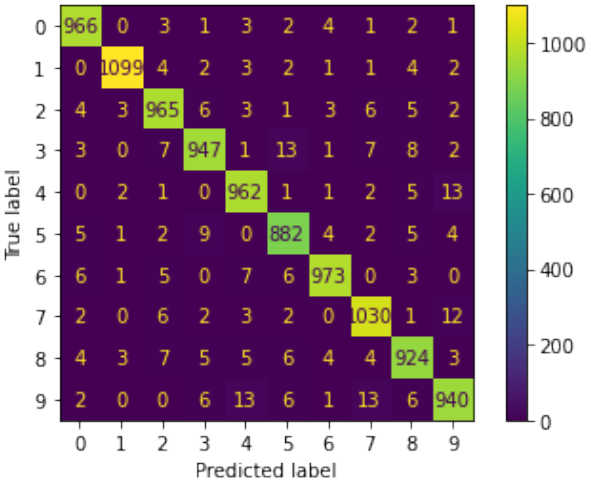
MLP before PCA



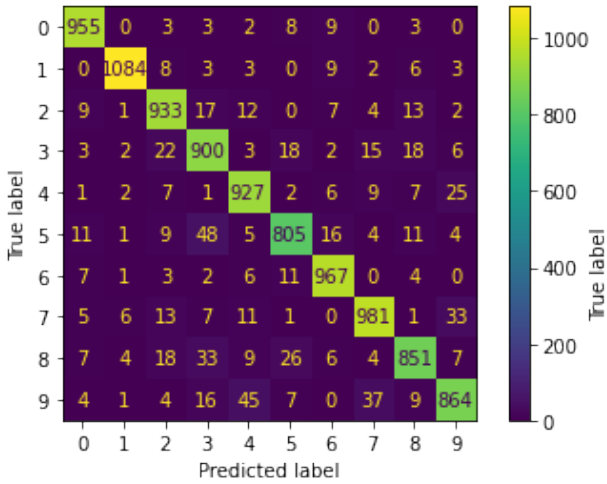
Random forest before PCA



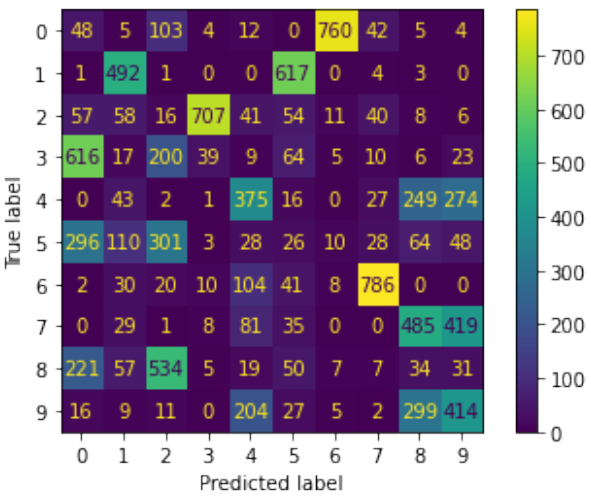
MLP after PCA



Random forest after PCA



K-Means before PCA



Looking at our confusion matrix's, we start to see some consistencies in what numbers get confused, and as to what they get confused as.
In the case of both random forest, MLP and Kneighbors; 4 is consistently getting confused for a 9 more often than an 8, and 3 is often getting confused for a 5 but very rarely as a 1.
This happens consistently on all the models that contain a high Youden's index. On the other hand, for models that has a rather low index, such as K-Means, we see numbers such as a 6 getting confused for a 7 extremely often, which we otherwise can't see in the models that

has a high Youden's index.

5 Conclusions

Our research question was how well different image classification models perform, as such we in our analysis set up several different models in order to create comparable data. We split an MNIST dataset into 3 sets and trained our models with the largest of them, before having them predict on the training set, and the 2 other sets. We found Youden's index for each of their predictions and the time it took them to make predictions, repeating the same after applying PCA.
We can see that regardless of PCA, KNeighbors had the shortest training time. The model with the shortest average prediction time is K-Means after PCA, though PCA applied MLP sits really close.
The best average Youden's index sits at PCA applied MLP, though it is only marginally better than random forest before PCA, and with a worse training time.
We could see in our findings that if we took every model into account both before and after PCA, then PCA applied MLP model had the lowest time and the highest Youden's index.
In terms of the effectiveness of the PCA: In our findings we saw a marginal improvement in the MLP model on all fronts, and a general improvement for other models on training and prediction time, though at the cost of a worsened Youden's index for them. This is to be expected given that features from the dataset were reduced. Random forest did not improve in any way, neither training, prediction nor index score. Meaning PCA most often shorten training and prediction time, but does not necessarily do so for all models. This could be helpful should one have an enourmos dataset that could take hours, or if one is using the MLP model. But may not be worth it for smaller datasets on other models, and not at all on naive bayes, as far as can be seen from the empirical evidence.
Finally, we believe that in terms of image recognition for classifiers for certain digits, there is indeed a pattern in what digits get mixed together. As could be seen by the confusion matrix's whenever it had to deal with the digit 4, consistently confusing it for a 9 more than an 8. Or a 3 getting most often confused as a 5. This is most prominent at higher Youden's index however, and such patterns get disrupted at lower index scores.

References

[1] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[2] Niva Elkin-Koren. Contesting algorithms: Restoring the public interest in content filtering by artificial intelligence. *Big Data & Society*, 7(2):2053951720932296, 2020.

[3] Sandeep Reddy, Sonia Allan, Simon Coghlan, and Paul Cooper. A governance model for the application of ai in health care. *Journal of the American Medical Informatics Association*, 27(3):491–497, 2020.

[4] Cambridge Consultants Tim Winchcomb. Use of ai in online content moderation, 2019.

[5] Jay F Nunamaker Jr, Minder Chen, and Titus DM Purdin. Systems development in information systems research. *Journal of management information systems*, 7(3):89–106, 1990.

[6] Aurélien Géron. *Hands-On Machine Learning With Scikit-Learn Tensorflow*. 2017.

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[8] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.

[9] An introduction to computing with neural nets.

[10] Michael Galarnyk. Pca using python (scikit-learn). *Towards Data Science*, 2017.

[11] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[12] Guogen Shan. Improved confidence intervals for the youden index. *PloS one*, 10(7):e0127272, 2015.