MNIST digit recognition analysis
Lab report

Auer Thomas

September 13, 2022

Contents

1. Introduction

The following report describes various setups for experimentation on the MNIST database of handwritten digits using support vector machines, principal component analysis, and neural networks. The goal is to analyze the results and runtime of the various algorithms, given multiple configurations. For this purpose the libraries SKLearn and Keras of the programming language Python are used.

2. Methods and Resources

The imported dataset for all computations is provided by the keras.datasets package. All python scripts include the datetime and time packages for timing purposes, while matplotlib offers some visualization functionalities for datasets. With the implementation of 4 python programs, each script uses its own respective set of packages.

1. SKLearn support vector machine (SVM)
2. SKLearn support vector machine with principal component analysis (PCA)
3. SKLearn multi-layer perceptron classification, with and without PCA
4. Keras Sequential model

The programs implementing the SKLearn-Kit library further allow for hyperparameter search using GridSearchCV, which allows automated customization and inspection of each algorithm over various pre-defined parameters in search of an optimal score for specific criteria, here "accuracy". The script using the Keras library has additional information on which images were especially difficult to classify.

Approach

After fetching the data from the Keras database, it is necessary to reshape and normalize the data for machine learning purposes. The reshaping of the data is the transformation of the original three-dimensional dataset into a two-dimensional training set. We reshape the data so that we have access to every pixel of the image. The reason to access every pixel is that only then we can apply deep learning paradigms, and , furthermore, we can assign color code to every pixel. We can utilize RGB color codes, where different values express various colors, each with a maximum of 255. Hence we convert all available pixel values from the range 0 to 255 to the range of 0 to 1, by simple division. Lastly and optionally, we reduce the size of the training and testing data to speed up the experimentation. Afterward, the model fitting begins. In the context of this research, the accuracy and the execution time are used to rate each algorithm.

3. Results

The following results were created over one-fourth of the original test data set size and using default parameters, which are shuffled with several predefined values during hyperparameter search. Notably, the displayed values are one of many produced results and may vary from execution to execution. Given the length restriction of the report, one set for exemplary display was deemed sufficient, but please view the latest entries of each respective file in "log-win" for additional aggregated results.

Support Vector Machines

Kernel	Accuracy	Time [s]
Linear	0.8935574229691877	2.556283473968506
Poly	0.938375350140056	2.7583091259002686
RBF	0.9487795118047219	6.743616104125977
Sigmoid	0.7911164465786314	4.9489216804504395

Table 3.1: Algorithms and their results via default parametrization.

The Hyperparameter search, via GridSearch, uses 2 parameters in addition to the kernel, namely C and Gamma. The C parameter is a trade-off between correctly classifying training examples and maximizing the margin of the decision function. A low C value allows larger margins and thus eases restrictions on the decision function but at the cost of accuracy. A small margin, by using a large C value, can produce better results if the decision function classifies all training points. Gamma can be described as a radius of influence around training samples. The higher the value, the smaller the radius for all samples. For more details on these parameters, please this documentation.

Kernel	C	Gamma	Score
Linear	1	0.01	0.9147267755918639
Poly	100	0.005	0.9616640213404468
RBF	50	0.01	0.9683311325997555
Sigmoid	100	0.0005	0.9311945537401357

Table 3.2: Algorithms and their hyperparameter evaluation score.

Principal Component Analysis and Support Vector Machines

The following results utilize principal component analysis in conjunction with support vector machines. PCA is the decomposition of singular values of the data to project it to a lower dimensional space using variance, to improve execution time and accuracy.

Kernel	Accuracy	Time [s]
Linear	0.8935574229691877	2.47495698928833
Poly	0.9555822328931572	4.203923940658569
RBF	0.9551820728291317	7.085856199264526
Sigmoid	0.8563425370148059	3.6313655376434326

Table 3.3: Algorithms and their results via default parametrization on PCA-transformed datasets.

Kernel	C	Gamma	Score
Linear	1	0.01	0.9147267755918639
Poly	50	0.01	0.9719982216294321
RBF	50	0.01	0.9683311325997555
Sigmoid	100	0.0005	0.9313278870734688

Table 3.4: Algorithms and their hyperparameter evaluation score on PCA-transformed datasets.

Neural Networks

The following results were produced by neural network algorithms, using SKLearn and Keras, respectively.

Estimator	Accuracy	Time [s]
SKLearn-MLP	0.9507803121248499	0.002245187759399414s
SKLearn-MLP-PCA	0.9307723089235694	0.006018638610839844s
Keras-Sequential	0.9347739219665527	0.14106178283691406

Table 3.5: Algorithms and their results via default parametrization.

Additionally, the Hyperparameter search for the SKLearn-MLP algorithm suggested the following optimal choice of parameters. For details on the parameters, please refer to the official documentation.

Estimator	Act.Fun.	Alpha	Layer	Learn Rate	Solver	Score
SKLearn-MLP	'relu'	0.05	(784,)	'adaptive'	'adam'	0.9655973324441481
SKLearn-MLP-PCA	'relu'	0.05	(784,)	'constant'	'adam'	0.9673976436589975

Table 3.6: Algorithms and their hyperparameter evaluation score on PCA-transformed datasets.

4. Discussion

The programs and their results are sufficiently satisfactory.

The differences of each kernel are made clear thanks to the hyperparameter evaluation, albeit the outcome of the principal component analysis-supported datasets were surprising. In comparison with the simple-reshaped datasets, the accuracy has risen while the time needed has worsened for most cases. Given that the main idea of PCA is to explain variances between different data points and dimensions, it may not be optimal to use on the MNIST-Digit dataset which consists only of two-dimensional pixel images. On the other hand, the neural networks have performed excellently, with accuracies rivaling that of the SVMs at far faster execution times.

The results indicate that both, SVMs and neural networks, with and without PCA, are extraordinarily useful algorithms to recognize hand-written digits. Overall the neural network performed better in terms of time (SKLearn-MLP: 0.0022s), while SVMs allowed for the highest possible accuracy (Poly-PCA: 0.9719). It is noteworthy that there are perhaps far better results achievable, depending on the experimentation of parameters.

5. Conclusion

As a layman in training, it was a difficult but entertaining experimentation. The most remarkable knowledge I have gained during this research is that neural networks can be effectively rivaled by other algorithms, if the interpretation and use of provided data is done sensitively. In practice, I believe that neural networks are a good choice for initial training setups, given how easy and fast they're programmed. SVMs on the other hand can pave a road to maximize specific goals, such as accuracy or loss.