Preliminary Report: Assorted Algorithms on MNIST

Dylan Rodriquez
drodriqu@purdue.edu

Javad Raheel
jraheel@purdue.edu

April 10, 2020

1 Introduction

Recognizing handwritten digits is an important problem in several domains of computer science which include optical character recognition, computer vision, and machine learning. MNIST, a dataset consisting of handwritten digits, is a padded version of the NIST handwritten images dataset [Deng (2012)] is widely used for these tasks. Images form this dataset are centered by a bounding box and are 28×28 in dimension (figure 1). This project is about implementing three algorithms, namely K-nearest neighbors (KNN), support vector machine (SVM), and neural network (NN) and studying their performance the MNIST dataset. The focus is on tuning the models by employing k-fold cross-validation followed by statistical analysis to select the hyperparameters for each model.

2 Methodology

2.1 Data

The data are from the MNIST dataset in idx format. The data are portioned into four files that contain the training features, testing features, training labels and testing labels [LeCun, Cortes, and Burges (2010)].

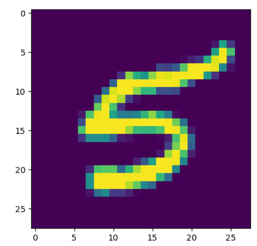

figure 1: Sample from MNIST Dataset

Each dataset was processed according to the provided offsets, types, and values and cast to a numpy array.
The samples are represented as a $n \times 784$ matrix, where n is the number of samples in the training or testing features. Labels consist of a single column vector with the same number of as the number of samples in its respective feature matrix.
Data are shuffled each time upon loading and training data are split into 80% training and 20% validation.

2.2 K-nearest neighbors (KNN)

The KNN classifier was implemented using scikit-learn. If a GPU is present, the model is implemented in cuML. Hyperparameter for KNN is the number of neighbors (K).

2.3 Support vector machine (SVM)

The SVM has not yet been implemented. We intend to implement it using again the scikit-learn library and explore GPU option as well as the CPU one. We intend to tune the kernel (Gaussian, polynomial), and the parameter C as the hyperparameters for SVM. This should be completed by April 16 2020.

2.4 Neural network (NN)

The neural network has not yet been implemented. This would also be implemented using scikit-learn. The hyperparameters to be tuned for NN are the number of hidden layers (2 or 3), and the number of neurons in each hidden layer. We intend to complete it by April 16 2020.

2.5 Platform Specifications

Experiments were conducted on a 2.60-GHz Intel hexa-core i7-9750H with 16 GB memory and an NVIDIA 1660 Ti 6 GB GPU in a Python 3.7.6 environment with scikit-learn Pedregosa et al. (2011) and cuML (RapidsAI) Raschka, Patterson, and Nolet (2020)

2.6 Cross validation

5-folds cross-validation was performed by partitioning the training data into five subsets. For each run, four of the subsets were used to train the model while testing was done on the fifth partition. The algorithm index partition provides k different orderings of indices, where the first index is the excluded partition for validation.

Cross validation performs training and testing in a leave one out manner on the validation data and for each k folds produces the mean and variance for each fold.

THIS ALGORITHM IS THE CURRENT kfold FUNCTION:

Algorithm 1 cross validation

```
$k \leftarrow$ number of folds
$S \leftarrow \{S_1, S_2, ...S_k\}$, k partitions of the training set
**for** i to k **do**
    train with sets$S_1, ..., S_{i-1}, S_{i+1}, ..., S_k$
    test on set $S_i$
    let $M_i$be the Mathew's correlation coefficient when the model is tested on $S_i$
**end for**
return the mean and variance of the set $\{M_i\}_{i=1}^k$
```

2.7 Parameter tuning

Each run of cross validation for a model's settings (i.e. the values of hyperparamters) provides us with a mean and variance of the Mathew's correlation coefficient (MCC) of the model. We would henceforth get n means and variances of the model's Mathew's coefficient for n values of the hyperparameters. To determine the *best* model settings, we use statistical analysis of these means and variances.

ANOVA is used to determine if there is a statistically significant difference in mean MCC of the samples. If there is no statistical significance, one is chosen at random, if the means are different, then post-hoc testing begins.

During the post-hoc analysis we currently use a T-Test with all pairs of folds. the null hypothesis being that there is no difference in means between folds. The alternative hypothesis is that there is a difference in means. We assume that the mean MCC is normally distributed with mean and standard deviation of the population being unknown. Because of the small sample size and an unknown population mean and standard deviation we use a two tailed T-test.

If the null hypothesis is rejected one tailed T-tests with the same assumption are performed to determine if the mean of one fold is greater than another. if the null hypothesis is not rejected then other pairwise tests continue.

Additionally, we plan to implement Tukey's test in substitute of pair-wise t tests due to family-wise error and the lack of independence between means after performing ANOVA.

3 Results

4 Discussion and Conclusions

References

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, *29*(6), 141–142.

LeCun, Y., Cortes, C., & Burges, C. (2010). Mnist handwritten digit database. 2010. *URL http://yann. lecun. com/exdb/mnist*, *7*, 23.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *arXiv preprint arXiv:2002.04803*.