COMP 551 - Mini-Project 2

Jessica Helena Chan
William Bouchard
Frederic Beaupre

November 22, 2020

Abstract

In this project, we compare three classification algorithms, Logistic Regression (LR) with mini-batch gradient descent, K-Nearest Neighbours (KNN) and a Convolutional Neural Network (CNN) on three datasets. Momentum is incorporated into the gradient descent in the LR model and visual analysis is conducted on the effects of hyper-parameters. Specifically, the training and validation accuracy of different learning rates, batch sizes and momentum parameters are plotted to visualize the effects. For one dataset, we perform principal component analysis to reduce dimensionality. Optimal hyper-parameters are found using 5-fold cross validation and models are compared based on test accuracy.

1 Introduction

The objective of this project is to familiarize ourselves with gradient-based optimization and gain experience comparing the performance of different models. We implement a Logistic Regression model that employs momentum and mini-batch gradient descent for optimization, and tune for hyper-parameters using 5-fold cross validation. This model is compared to SciKit Learn's K-Nearest Neighbour in terms of test accuracy and CNN implemented using Keras. We conduct further analysis on the effects of learning rate, mini-batch size and momentum on the training and validation accuracy of the Logistic Regression model. We find that K-Nearest Neighbours and CNN generally perform better and run faster than our implementation of Logistic Regression.

We train and test on three datasets for classification: the Digits dataset from sklearn.datasets [1], the Scene dataset from OpenML [2], and the Leaf dataset from OpenML [3]. We find that the models perform better on some datasets over others.

2 Datasets

The first dataset, MNIST Digits from sklearn.datasets, contains vectorized grayscale pixel data of images of handwritten numbers from 0 to 9, labelled by their respective digits. The data is loaded into a Pandas DataFrame and labels are one-hot encoded. The second dataset, Scene from OpenML, contains vectorized coloured pixel data of images of scenery. Each entry is labelled as Beach, Sunset, FallFoliage, Field, Mountain, or Urban through one-hot encoding. Principle component analysis was performed on Digits and Scene in order to reduce dimensions, with the number of components selected based on reconstruction error. The datasets are split into 80/20 train/test splits before being fed into the models. Training data is further split into 5 cross validation folds for hyper-parameter tuning.

To further analyze the regression models and the impact of the quality of datasets on the end results, we decided to use the Leaf dataset from OpenML. This dataset was particularly chosen since it has fewer samples (340 entries), 16 features and 30 possible targets. The idea behind this choice was to explore the impact of the amount of data and number of classes when using our models for classification. The targets are one-hot encoded and the hyper-parameter tuning is done in the same way as in the Digits dataset.

3 Results

Out of all the models, KNN yields the best accuracy with CNN coming in close second and Logistic Regression last. We note that the Leaf dataset yields the lowest accuracies, and we believe that this is due to a high amount of classes (30) and a smaller number of samples (340). The models generally perform better on Digits than on Scene, and this may be due to a higher amount of noise in coloured data. We find that PCA reduction slightly improves performance in both cases.

Model	**Digits**	**Scene**	**Leaf**
`Logistic Regression`	86.67	60.30	8.24
`KNN`	99.56	74.25	18.82
`CNN`	98.46	-	-

Table 1: Test Accuracies (in %) for Logistic Regression, KNN and CNN. LR optimal hyperparameters found using 5-fold cross validation: Digits - [0.01, 0.98, 512] Scene - [0.01, 0.97, 512], Leaf - [0.001, 0.98, 128]

3.1 Principal Component Analysis

Principal Component Analysis (PCA) was performed on the Digits and Scene datasets. The number of principle components were selected using the elbow method after visually analyzing reconstruction error plots (Figure 1). We found that PCA reduction generally increased test accuracy for Digits and Scene (Table 2), but had a somewhat negative impact on validation accuracy (Figure 2). This may suggest that PCA reduction helped to reduce over-fitting to the training set.

Model	**Digits**	**PCA Digits**	**Scene**	**PCA Scene**
`Logistic Regression`	86.67	88.89	60.30	67.78

Table 2: Test Accuracies (in %) for Logistic Regression with original data and PCA reduced data. Logistic regression hyper-parameters are found using 5-fold cross validation: Digits - [0.01, 0.98, 512], Digits PCA - [0.01, 0.98, 256], Scene - [0.01, 0.97, 512], Scene PCA - [0.01, 0.97, 256]

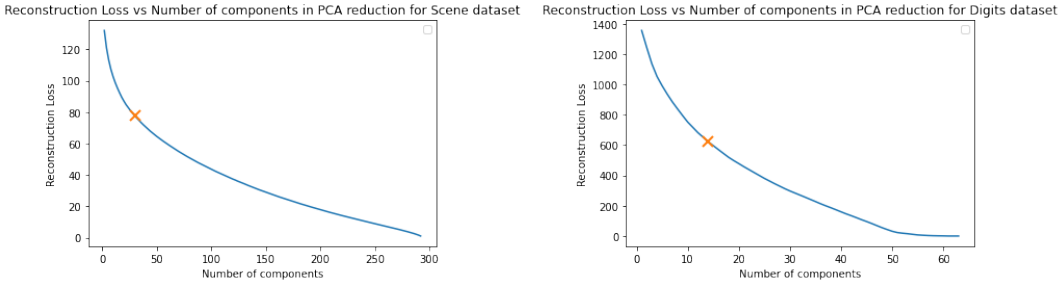

Figure 1: Elbow method for finding best number of components for PCA. Scene: n = 30, Digits: n = 14

3.2 Logistic Regression

The Logistic Regression model utilizes mini-batch gradient descent momentum to minimize soft-max cross entropy loss and to learn weight parameters. The gradient descent terminates if cross entropy loss has not decreased sufficiently after 10 time steps, if validation accuracy has not increased sufficiently after 10 time steps, if the length of the gradient is less than 1e-8, or after 1000 time steps. The model weights with the best validation accuracy are returned after convergence. Grid search cross validation with 5-folds was used to find optimal hyper-parameters. The search space included learning Rate (alpha): [0.001, 0.005, 0.01], momentum (beta): [0.97, 0.98, 0.99], and batch-size: [128, 256, 512]. Note that we narrowed down the lists of hyper-parameters by running multiple individual tests with many values for each hyper-parameter. This effectively reduced the run-time of the grid search. Unique hyper-parameters were found for each dataset and its PCA reduction, and the model weights with the highest validation accuracy were used to predict labels for the test set. We limited our choices of hyper-parameters due to time constraints, however, we believe that higher validation and better test accuracies could have been achieved given more time to run. Test results and tuned hyper-parameters for each dataset are listed in the description of Table 1.

Training and validation accuracy curves were plotted for one cross validation fold for models with the best and worst validation accuracies and run-times for analysis of the hyper-parameters. We found that models with smaller learning rates converged faster, often meeting the cost termination conditions in 10 time steps. We also found that the fastest models tended to be the most inaccurate, whereas the slowest models tended to be the most accurate. After PCA reduction, the generalization error for validation seemed to increase, suggesting that PCA reduction caused under-fitting for the validation set (Figure 2).

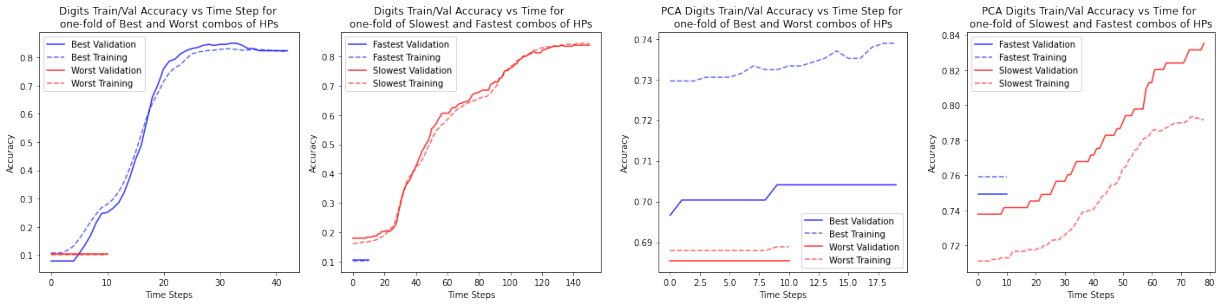

Figure 2: Train/val accuracy curves of final cross validation fold for models with the following hyperparameters; Digits: Best validation accuracy - [0.01, .98, 512], Worst validation accuracy - [0.001, 0.99, 512], Fastest convergence (10 timesteps) - [0.001, 0.99, 512], Slowest convergence (155 timesteps) - [0.001, 0.98, 128]; PCA Digits: Best validation accuracy - [0.01, 0.98, 256], Worst validation accuracy - [0.005, 0.98, 512], Fastest convergence (10 timesteps) - [0.005, 0.99, 512], Slowest convergence (80 timesteps) - [0.01, 0.99, 512]

3.3 K-Nearest Neighbour

We fit a K-Nearest Neighbours model to the three datasets using Scikit-Learn's KNeighboursClassifier. For hyper-parameter tuning, we investigate the number of neighbors $K \in [2, 8]$ and distance metrics (Euclidean, Manhattan and Minkowski, see Appendix for details). We performed grid-search cross validation with 5-folds to find the optimal hyper-parameters as we did for the Logistic Regression model. We then plotted the accuracies over the number of dimensions for each distance metric, for each dataset:

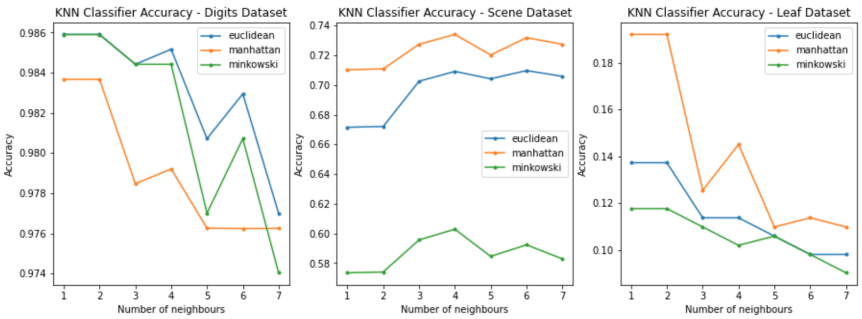

Figure 3: KNN classfier accuracies per distance metric for the three datasets.

While the distance metric's influence on run-time is similar for all three datasets, with Minkowski yielding the highest run-times, we observe that the performance and optimal distance metrics are not the same across different datasets. From tuning, we find that the best hyper-parameters for each dataset are: **Digits Dataset**: $K = 4$; metric: Euclidean, **Scene Dataset**: $K = 4$; metric: Manhattan, **Leaf Dataset**: $K = 2$; metric: Manhattan. We then predict the classes of the test data using these hyper-parameters for each model and obtain the accuracies described in Table 2.

3.4 Convolutional Neural Network

Because the selected datasets represent pixel data, we wanted to find out how a Convolutional Neural Network (CNN) would perform in comparison to the other classifiers. 2D convolutional networks are especially well structured for processing image data, utilizing kernels that pool and process surrounding pixel data to learn features of the image, such as vertical edges. We implemented a 2D conv-net whose convolution layers have 32 filters and strides of 3 pixels along both dimensions, one 2D max pooling layer and 2 dense layers with relu and softmax activation functions. The CNN model was implemented using Keras modules and the MNIST dataset was provided by Keras as well. We did 5-fold cross validation on the conv-net, where we fit 4 of the 5 folds using 10 epochs and a mini-batch size of 32, and we validated with the last fold. The average accuracy obtained from the 5-fold cross validation was 98.46, as shown in Table 2.
Note that we were not able to find the 2D counterparts for the Scene and Leaf datasets.

4 Discussion and Conclusion

In this project, we implemented a Logistic Regression model with mini-batch gradient descent momentum from scratch and familiarized ourselves with hyper-parameter tuning using cross validation and grid search. We compared our implementation with 2 other classifiers, namely K-Nearest Neighbours and 2D Convolutional Network, and gained experience in comparing classifiers based on accuracy. Beyond this, we practiced dimensionality reduction techniques seen in class to improve our results.

In terms of improvements, we would perhaps try to increase the accuracy of the models when predicting the Scene dataset by applying different normalization techniques and testing their impact on the accuracy achieved. In addition, we could also implement L2 regularization in the cost function of our Gradient Descent optimizer and compare the results yielded by this modification with the ones already obtained. We believe that the resulting test accuracy should be better, especially for the Scene dataset which happens to be over-fitting on some occasions. Finally, as mentioned already, we used a very short list of combinations of hyper-parameters to meet the time constraints. In fact, we are only testing for 27 combinations. This number could be brought to multiple hundreds if given more time and more computational resources. We estimate that the accuracy could probably be brought up over 90% given the right hyper-parameters.

5 Statement of Contributions

The work was distributed evenly.

References

[1] Pen-based recognition of handwritten digits data set.

[2] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification, 2004.

[3] Pedro F. B. Silva, André R. S. Marçal, and Rubim M. Almeida da Silva. Evaluation of features for leaf discrimination, 2013.

6 Appendix

6.1 Distance Metrics for KNN:

$$\text{Euclidean: } \sqrt{\sum (x-y)^2} \tag{1}$$

$$\text{Manhattan: } \sum |x-y| \tag{2}$$

$$\text{Minkowski: } (\sum |x-y|^p)^{\frac{1}{p}} \tag{3}$$

where we use $p = 5$.