The Application of Logistic Regression and Softmax Regression on KMNIST Dataset

Wenbo Hu, Wenxiao Li, Huaning Liu
University of California, San Diego
Email: w1hu@ucsd.edu, wel032@ucsd.edu, h9liu@ucsd.edu

***Abstract*—The project utilizes logistic regression and softmax regression to detect the classification of KMNIST data. Both binary classification with a subset of the data as well as multiple class classification will be explored separately. For the binary classification between labels 0 and 6, a test accuracy of about 98.9% is finally achieved; while for the binary classification between labels 2 and 6, a test accuracy of about 87.85% is achieved. For multi-class classification, labeling from 0 to 9, the model is trained to have a validation accuracy of about 82.15%, with a test accuracy of 70.1%. It is noticeable and interesting that there exists differences by test accuracy for different subsets in binary classification cases. More in-depth exploration about such disciplines will be discussed in this project.**

***Index Terms*—Logistic Regression, Softmax Regression, Deep Learning, KMNIST**

I. Introduction

The KMNIST dataset explored in this project is a Hiragana character set that is a part of the Japanese writing system. Accordingly, a train set with matching labels and a test set with matching labels are given as data. The target at this time is to train a model that fits for both binary classification (i.e. classify dataset with only two labels) and multi-class classification (i.e. classify dataset with ten labels from 0-9 in this case). In this case, stochastic gradient descent is applied to optimize the model, as well as k-fold cross validation is included for better interpretation of the training process. By building a network of logistic regression and softmax regression implemented by python with its numpy library, we explore what train and test accuracies can be correspondingly achieved; train losses and validation losses are also recorded for further plotting and analyzing. Then, we tried out different combinations of hyperparameters, including learning rate, number of epochs, k-folds, etc, in order to tune and optimize the model to better fit the training data. Finally, by pushing the model into the test set, we check how good our model could perform on the test set. A high test accuracy is achieved on 0 and 6 binary classification by logistic regression; while multi-class classification by softmax regression performs a relatively high test accuracy as well. This shall be an expected output, and we are going to further discuss the relations and reasons behind the outputs.

II. Related Work

Our work focuses on the exploration of KMNIST dataset by logistic regression and softmax regression, with logistic regression based on several subset data of the dataset. With the dataset to be open sources, there are some other works that explore this dataset and try building models based on the data with different approaches, as well as interpreting and comparing the results from different perspectives.

In Harshil Shah and his team's paper **A Comparative Study of Classification Algorithms with Varying Training Dataset Sizes on Cursive Hiragana Characters** [1], four different algorithms, including Support Vector Machine (SVM), Multi-layer Perceptron (MLP) and two different architectures of the Convolutional Neural Network (CNN) on Kuzushiji-MNIST is provided for comparison. In this study, one linear classification algorithm and three non-linear classification algorithms are provided and their performances, in metrics of accuracy, with different percentages of data used, are presented and compared. The author detects that with more data used, the higher accuracy will be achieved, the slower the test accuracy will grow. In our project, from a different perspective, two linear classification algorithms, linear regression and softmax regression are included; and different pairs of binary labels, with their image data, are extracted and trained. In this way, we discuss and compare the performance binary logistic regression for different pairs of labels.

In Tarin Clanuwat and his team's work **Deep Learning for Classical Japanese Literature** [2], Kuzushiji-MNIST (KMNIST) as well as two larger datasets are introduced. The authors explain the configuration of the datasets at the beginning, then do experiments on the datasets by five different models as baselines, including 4-Nearest Neighbour Baselines, Keras Simple CNN Benchmark, etc. Then, the separate training results are compared and analyzed. In our work, only the KMNIST is used for training. Also, compared with Tarin's work that focuses more on the generating process of the dataset, while we explore from the training perspective to achieve an optimal model and compare the model performance on different sub-dataset.

III. Dataset

As a drop-in replacement for the MNIST dataset, this dataset consists of ten kinds of Hiragana in Japanese, with the corresponding images representing the handwritings for that character. There are 70,000 images (28×28 grayscale) in total, with each of the classes averagely taking 7,000 images. In this case, as Figure 1 shows, for each of the classes, 6,000 of the images are split to be in the train set; 1,000 of which are in the test set. In other words, the train set has 60,000 images

class	English Pronunciation	train size	test size
class 0	o	6000	1000
class 1	ki	6000	1000
class 2	su	6000	1000
class 3	tsu	6000	1000
class 4	na	6000	1000
class 5	ha	6000	1000
class 6	ma	6000	1000
class 7	ya	6000	1000
class 8	re	6000	1000
class 9	wo	6000	1000

Figure 1. Count of observations for train set and test set

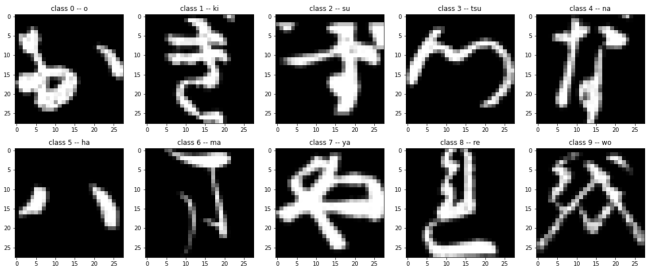

Figure 2. Sample handwriting image for each class

and the test set takes 10,000. Therefore, for statistics for the different splits of the dataset, counts of observations in the train set and test set are evenly distributed among the classes.

Figure 2 shows a randomly sampled example for each class, with the matching labels as well as numerical labels from 0 to 9 encoded in the training/test sets with its character's English pronunciation appended above. Then, to visualize and get a rough idea about the difference of images inside each classes, another 10 randomly samples images are presented in Figure 3. Furthermore, to better interpret the whole dataset for each class, the average images for the 10 classes are presented in Figure 4. We detect that unlike single MNIST or Chinese Hand-writing characters recognition, Japanese hand-writings varies more and therefore potentially more difficult to train and classify in further steps.

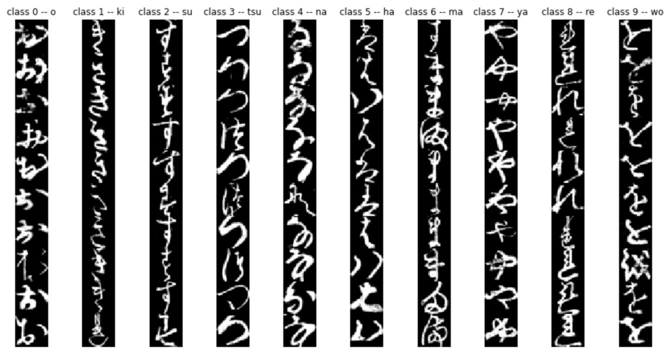

Figure 3. 10 sample images for each class

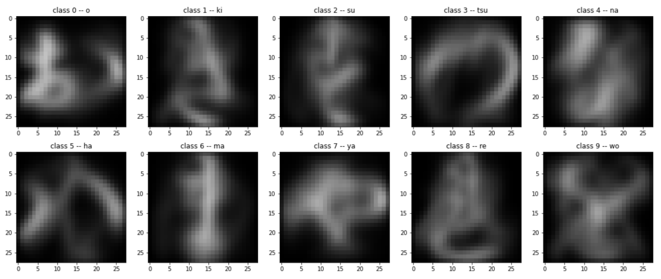

Figure 4. Average image for each class

Based on this dataset, our goal, as mentioned above, is training models to classify the images. Hence, proper data preprocessing is implemented before building the model. First, in further softmax regression, we one-hot encode the labels to make each of them a vector of length 10. Then, for the training set, we give an attempt on either doing min-max normalization or doing z-score normalization. The z-score normalization is implemented following the rule $f(x) = \frac{x-\mu}{\sigma}$; the min-max normalization is implemented following the rule $f(x) = \frac{(x-min(x))}{(max(x)-min(x))}$. On the other hand, some random biases are appended by the end of each vector for us to necessarily build the logistic regression as well as the softmax regression.

IV. Logistic Regression

Here, we use the Logistic Regression model to discern between two classes of data, namely C_0 and C_1. The logistic regression model consists of trained weights w, and an activation function $g(a) = \frac{1}{1+\exp(-a)}$. For any input vector x, the model will first insert a leading entry 1 in x to match the bias weight, then compute $a = w^T x$ using its trained weights. After that, the model will apply the activation function to output $g(a)$, where $0 < g(a) < 1$. If the output value is closer to 0, the model will identify the input x as a member of class 0, otherwise a member of class 1. To train the Logistic Regression model to discern between two classes, we first do pre-processing of the data to make them fit into each fold. Then, for each epoch, we shuffle the data, and use SGD with a fixed batch size over the loss function $E = -\sum_{n=1}^{N} \{t^n \ln(y^n) + (1-t^n)\ln(1-y^n)\}$, to update the weights along the way. Eventually, we keep the best weight amongst the k-folds.

V. Logistic Regression Discussion

1) **Class 0 vs. Class 6**
The best group of hyperparameters that we derived from our training model is learning rate = 0.01, and batch size = 128. Under these hyperparameters, we are able to achieve a validation accuracy of 0.9954, and a testing accuracy of 0.989.
The learning rate, if selected in common sense, does not affect the overall result much. For instance, fixing batch size = 128, and changing the learning rate to 0.001, the validation accuracy is 0.9918, while the testing accuracy

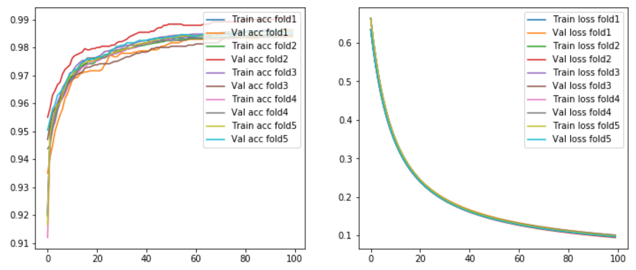

Figure 5. accuracy plot (Left) and loss plot (Right) for classification with class 0 and class 6

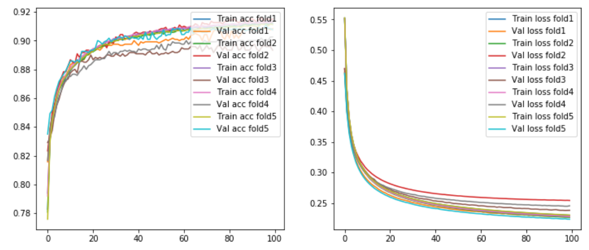

Figure 6. accuracy plot (Left) and loss plot (Right) for classification with class 2 and class 6

is 0.987. These results are way less accurate.
For different batch sizes, we get slightly different results. For instance, fixing a learning rate = 0.001, when the batch size is 512, we get validation accuracy being 0.9889, with the testing accuracy being 0.9865. When the batch size is 1, we get validation accuracy being 0.9941, with the testing accuracy being 0.984. Therefore, we pick batch size = 128 as the best parameter among the three. The related visualization is covered in Figure 5.

2) **Class 2 vs. Class 6**
The best group of hyperparameters that we derived from our training model is learning rate = 0.01, and batch size = 512. Under these hyperparameters, we are able to achieve a validation accuracy of 0.9125, and a testing accuracy of 0.8785.
The learning rate cannot be too small, otherwise the update is not significant enough to approach such good results. For instance, fixing batch size = 512, and changing the learning rate to 0.001, the validation accuracy is 0.8683, while the testing accuracy is 0.853. These results are way less accurate.
If the batch size is smaller, we still cannot get such good results. For instance, fixing a learning rate = 0.01, when the batch size is 128, we get validation accuracy being 0.8817, with the testing accuracy being 0.8775. When the batch size is 1, we get validation accuracy being 0.8783, with the testing accuracy being 0.831. Therefore, we pick batch size = 512 as the best parameter among the three. The related visualization is covered in Figure 6.

3) **Overall Observations for Logistic Regression**
The performance gap between the "Class 0 vs Class 6" group and the "Class 2 vs Class 6" group is mainly due to the similarities between Japanese characters. From graph Figure[3], we can clearly see that "Class 2, Class 6" have many similar patterns, while "Class 0, Class 6" are very different. Therefore, our model will perform better when classifying between Class 0 and Class 6.

VI. Softmax Regression

Here, we use the Softmax Regression model to discern between multiple classes of data. In our case, we have 10 classes in total, namely C_0, C_1, .. , C_9. The logistic regression model consists of trained weights w, and an activation function $y_k^n = \frac{exp(a_k^n)}{\sum_{k'} exp(a_{k'}^n)}$. For any input vector x, the model will first insert a leading entry 1 in x to match the bias weight, then compute $a = w^T x$ using its trained weights. After that, the model will apply the activation function to output a vector y, where all the entries in y are between 0 and 1, the sum of all entries is 1. From the output value, the model will identify the input vector x as a member of class k, if the kth entry in x is the largest.
To train the Softmax Regression model to discern between two classes, we first do pre-processing of the data to make them fit into each fold. Then, for each epoch, we shuffle the data, and use SGD with a fixed batch size over the loss function $E = -\sum_n \sum_{k=1}^c t_k^n \ln y_k^n$, to update the weights along the way. Eventually, we keep the best weight amongst the k-folds. r

VII. Softmax Regression Discussion

The best group of hyperparameters that we derived from our training model is learning rate = 0.075, and batch size = 512, and epoch = 50. The epoch is set to be 50 due to almost no observable changes being reflected after epoch 40 under this learning rate. From these hyperparameters, we are able to achieve a validation accuracy of 0.8215, and a testing accuracy of 0.701.

The learning rate will affect the overall result to some extent. For instance, fixing batch size = 512 and changing the learning rate to 0.1, the validation accuracy is 0.8198, and the testing accuracy is 0.6988, while changing the learning rate to 0.08 gives validation accuracy of 0.8208, and testing accuracy 0.699. Batch size will also impact the performance. For instance, fixing a learning rate = 0.075, when the batch size is 128, we get validation accuracy being 0.8112, with the testing accuracy being 0.6906.The related loss plot and accuracy plot are covered in Figure 7. On the other hand, the visualization of the weights of training model is included in Figure 8. From the weight patterns, we can visualize some implicit shapes of these original characters at places where weights are relatively smooth. Although it is not clear, some rough trends could still be detected following the shape of the characters, referring to the image sample and average image visualization in the dataset part as well. Hence, we conclude a relatively successful model training process from this visualization.

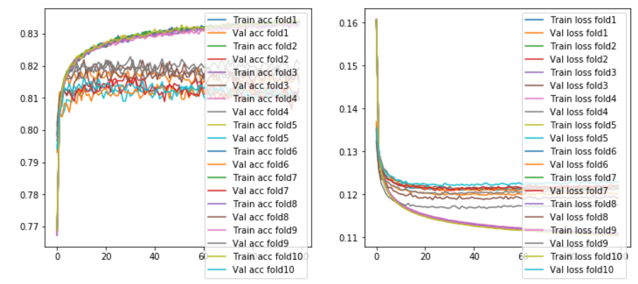

Figure 7. accuracy plot (Left) and loss plot (Right) for Softmax Regression

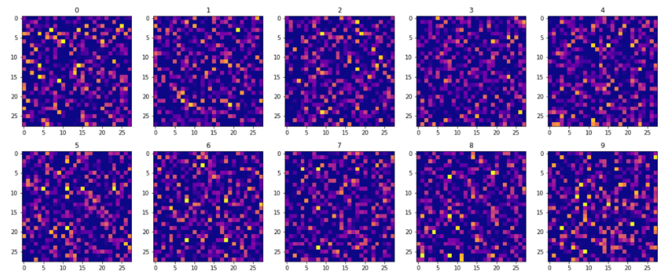

Figure 8. Visualization of the weights of training model

VIII. Conclusion

In this project, we explore the potential application of logistic regression and softmax regression on KMNIST dataset. The intuitive goal is to train models to predict the class that an image belongs to. In this case, we developed a logistic regression model to solve binary classification problems, by training models for both "class 0 vs. class 6" and "class 2 vs. class 6" for comparison. At the end, there exists a performance gap between the two sets, which might potentially be due to the property and shape of the Japanese character itself. On the other hand, a softmax regression model that deals with multiple class classification is trained, with a test accuracy of about 70% reported. A good take-away for this project will be the comparison and possible gap of different sets of training and the character-related reason behind that, and the potentially best performance of the models we are able to achieve.

IX. contribution

Wenbo: Implemented normalizations, cross validation, early stop, loss function, gradient descent and train network. Plotted training graphs and choosing hyperparameters.
Wenxiao: Implemented the networks for Logistic and Softmax Regression: connecting data-preprocessing, k-folds, epochs, SGD together. Testing the performances of weights. Use hyperparameter tuning to optimize the accuracies.
Huaning: Implemented data preparation and pre-processing file, did exploratory data analysis and visualization, organized and configure the report, wrote abstract, introduction, related work and dataset part of the report.

References

[1] H. Shah, R. Rajkumar, and J. Masih, "A comparative study of classification algorithms with varying training dataset sizes on cursive hiragana characters," *Xi'an Jianzhu Keji Daxue Xuebao/Journal of Xi'an University of Architecture Technology*, pp. 1388–1396, 08 2020.

[2] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," 12 2018.