Mini-Project 2

Multi-Class Logistic Regression and Gradient Descent

COMP 551, fall 2020, McGill University

Elaheh Astanehparast (elaheh.astanehparast@polymtl.ca), Sameen Mahtab (sameen.mahtab@ mail.mcgill.ca), Chris He (yinliang.he@mail.mcgill.ca)

Abstract – This project aims to compare the performance of two different classifiers (Softmax with mini batch gradient decent with momentum and KNN) on two different datasets (digits and vehicles). After data normalization, hyperparameters of both models are tuned and the training and validation error for five-fold cross validation is reported. Finally, the accuracy of the model using unseen data is calculated. We observed that Softmax takes much time than KNN while is more accurate. Also, we found that by changing hyperparameters, the performance of the model changes.

1. Introduction

In this project, we have compared the performance of Softmax classifier using mini-batch gradient decent approach vs KNN classifier on two datasets. The first dataset contains images of digits and the second one is designed for classifying four types of vehicles by looking at the image features of their silhouette. At first, we have normalized the data and tuned the hyper parameters. We found that hyperparameters have pivotal role in time complexity of the solution. Then we fitted the models using a five-fold cross validation approach on 80% of the data and calculated the training error and validation error and cost for each fold and acquired the mean of training and validation error. Finally, using the remaining 20% of the data as test data and the fitted models, the accuracy of the models for each dataset is derived. The result shows for digits dataset, KNN works slightly better than Softmax while for vehicles dataset, Softmax does a more accurate job. Also, Softmax classifier has higher time complexity as it is using gradient decent optimizer.

2. Datasets

We used two datasets for this project that will be described in this section.

2.1. First Dataset: Digits [1]

This dataset contains 1797 labeled instances each an 8x8 gray-scale image of a digit. By putting all the pixel information of an image in a row, we have 64 features. The class label obviously declares the digit appearing in the image. The dataset does not have any missing value and is already scaled (all the values for features are between 0 and 16 and have the same importance). We just encode the label using one-hot encoding while classifying by Softmax.

2.2. Second Dataset: Vehicles [2]

The second dataset was chosen from OpenML and contains features describing how a car looks like from the silhouettes. These features are extracted by the HIPS (Hierarchical Image Processing System) extension BINATTS. The dataset does not have any missing value. The only preprocessing was scaling the features to be between 0 and 1 and encoding the label using one-hot encoding while classifying by Softmax.

3. Results

In this section we will briefly go through the process of classification for each classifier and will provide the result of methods deployed per classifier and per dataset.

3.1. Softmax Classifier

The main steps here are creating cross validation folds, tuning hyper parameters, fitting the model using early stop approach, testing the model.

3.1.1. Cross validation

A five-fold cross validation is deployed on the training data to estimate the mean of validation error. The strategy of splitting data into training and validation is to select 20% of ordered training data each time as validation set and move the validation window for the next fold (Figure 1).

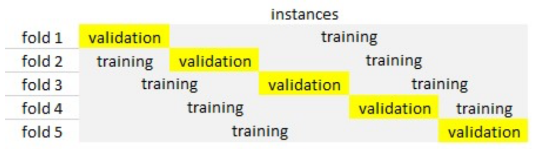

Figure 1 - cross validation approach

3.1.2. Tuning hyperparameters

As we are using mini-batch gradient decent for optimizing Softmax classifier, before using the model, we need to tune the hyperparameters including learning rate, momentum and batch size. For this purpose, we defined a parameter grid in which learning rate can be {0.1, 0.01, 0.001}, momentum can be {0.9, 0.8, 0.7}, and batch size can be {8, 16, 32, 64}. Using the 80% of the data as training, we acquired the training error, validation error and time spent for each combination of hyperparameters. Although time and training error are important features to be considered, we chose our best combination based on the lowest validation error. The tuned hyperparameters are shown in Table 1 while the full detailed information will be generated in the code.

Dataset	*Learning rate*	*Momentum*	*Batch size*	*Validation Error*
digits	0.001	0.7	16	8.02
vehicles	0.1	0.9	64	19.27

Table 1 - Best hyperparameter combinations that minimizes the validation error

Due to the fact that we did not have enough resources to test many different values for hyperparameters, we could not extract certain relations between error and each of hyperparameters. However, we will briefly discuss our main findings in the following.

Figure 2 is showing how mean of validation error changes over learning rate. One could say by increasing learning rate, the probability of having higher validation error increases.

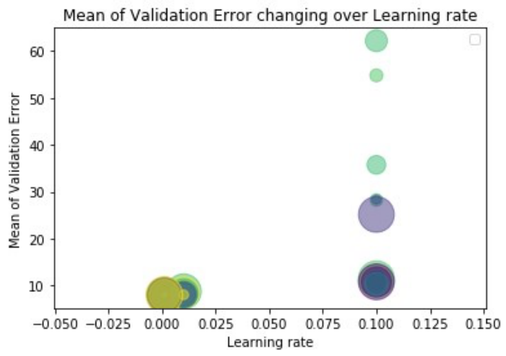

Figure 2- Changing mean of validation error over learning rate

For batch size, by decerasing batch size less than 10, the probability of having higher validation error increases (Figure 3).

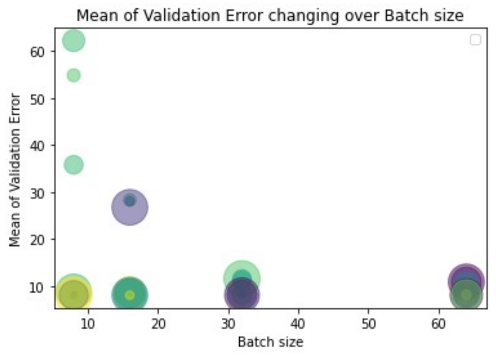

Figure 3- Changing mean of validation error over batch size

Also, looking at standard deviation of the time (8.3), one can conclude changing hyperparameters could change the convergence time noticably.

As it is declared, the batch size in both datasets is the same because batch size is a hyperparameter that depends more on the hardware architecture than dataset specifications.

3.1.3. Fitting the model using early stop approach

Now that all three hyperparameters are tuned, it is time to find the best value of weight matrix (w). To do this, for each fold, we create a new model using tuned hyperparameters and plot the training and validation error as well as calculate the mean of training error and validation error for each epoch (when all the batches in the data are used once). An example is shown in Figure 4. As it is shown the first error for training data is much less than the error for validation. This is because we store errors after each epoch when w has been updated for a couple of times using training data. As we move forward, w is more accurate so that could also predict validation data accurately.

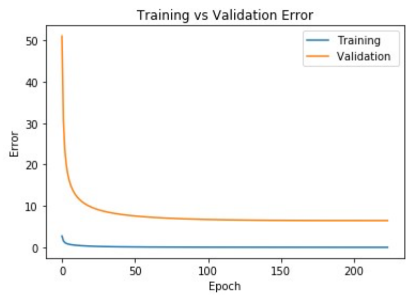

Figure 4- Example of training and validation error diagram for one single fold

The single value for error of the model will be calculated based on the following equation:

$$Error_{i,j} = \frac{1}{2}\left(y_{i,j}^{predicted} - y_{i,j}^{real}\right)^2$$

$$where\ y_{i,j}^{predicted} = \frac{e^{w^T_j x_i}}{\sum_{C_j} e^{w^T_j x_i}},\ error = \sum_i \sum_j Error_{i,j}$$

Also, the best w is reported per fold. Finally, best overall w will be calculated based on the following equation:

$$w_{i,j} = \frac{\sum_f w_{i,j}^{fold\ f}}{\#folds = 5}$$

As we normally see a huge number of iterations, in order to have more clear diagrams, we decided to calculate error after one complete epoch has been seen.

The loop for calculating gradient decent and updating the values of w will stop if one of these three criteria happens:

1- np.linalg.norm(grad) $\leq$ self.epsilon

(norm of gradient is very small)

2- $t \geq$ self.max_iters
(we have updated w many times)

3- self.validation_err_decreasing is false (for the last 20 epochs the validation error has not decreased. In this case the updated w related to the lowest validation error during the last 20 epochs will be returned as the optimal w)

The third condition helps not to overfit the model when the training error is decreasing but the validation error is not decreasing anymore.

It should be mentioned that in some inappropriate combinations of hyperparameter values, we observed that the norm of gradient will be very small before we have seen an entire epoch. In these cases, the most recent w will be returned as the best w.

3.1.4. Testing the model

For evaluating how the model behaves using unseen data, we use the remained 20% of the data to test the model. The result of testing using unseen data is provided in Table 2:

Dataset	***Mean of Training Error***	***Mean of Validation Error***	***Time***	***Test Accuracy***
digits	0.05	8.03	8.26	0.97
vehicles	8.07	19.27	17.43	0.71

Table 2- Tuned Softmax Results on different datasets

3.2. KNN classifier

The main steps here are creating cross validation folds, tuning hyper parameter, fitting the model and testing it.

Cross validation and testing the model is same as what is described in Softmax classifier. Here, there is only one hyperparameter K that should be tuned. We have tested different k between 1 to 50 and compared the accuracy of the model. The best K is the one that produces the least error (Figure 5). For calculating the error, we used Mean Absolute Error (MAE).

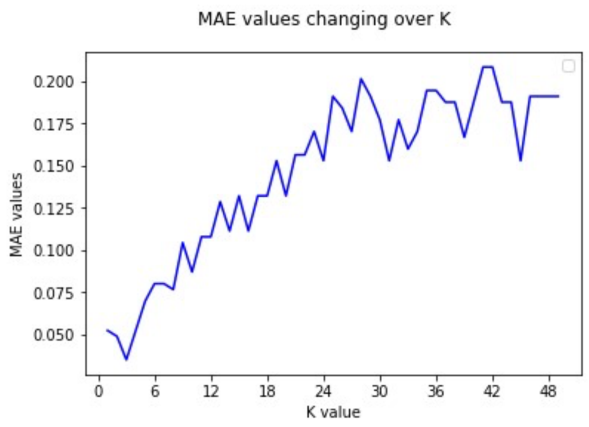

Figure 5- Selecting the best K that has lowest MAE

In the fitting process, we have used the five folds and reported accuracy for each fold. The mean of validation accuracy of the folds is reported as model accuracy. Finally, we have tested the model with unseen data and calculate the accuracy. The accuracy of test and validation data is provided in Table 3

Dataset	***Tuned K***	***Mean of Validation Accuracy***	***Test Accuracy***
digits	3	0.985	0.983
vehicles	1	0.712	0.688

Table 3- Comparison of accuracy of KNN

4. Discussion and Conclusion

A summary of the results of two classifiers on two datasets is provided in Table 4. As it is shown, one could say that Softmax could generally classify the data more accurate than KNN although KNN has reached a great accuracy for digits dataset because one of its most useful domains to be applied.

Classifier	***Dataset***	***Hyperparameters***	***Test Accuracy***	***Time***
Softmax	digits	learning rate = 0.001 momentum = 0.7 batch size = 16.0	0.97	8.26
	vehicles	learning rate = 0.1 momentum = 0.9 batch size = 64	0.79	17.43
KNN	digits	k=3	0.98	3.8
	vehicles	k=3	0.68	0.69

Table 4- Comparison of accuracy of the models for unseen data

In terms of time complexity, although KNN is not a fast classifier, it needs less time for classifying these datasets in comparison with Softmax. Obviously, the process of gradient decent is time consuming and resulted in a slower convergence.

Also, by increasing number of features, both classifiers are slowed down because number of features can impact the performance of machine learning algorithms noticeably.

For future works, one might use other values for learning rate, momentum and batch size to get more powerful model. Also, assigning a specific learning rate for each weight is stated to improve the convergence [3].

Also, these two classifiers could be tested with other datasets so that we could get a better insight of their abilities and differences.

Finally, instead of training and validation error, we could report accuracy which is more appropriate for plotting diagrams.

References

[1] https://scikitlearn.org/stable/modules/generated/sklearn.datasets.l oad_digits.html

[2] https://www.openml.org/d/54

[3] Grégoire Montavon, Geneviève B. Orr, Klaus-Robert Müller, “Neural Networks: Tricks of the Trade”, Second Edition, Springer