Logistic Regression with MNIST

MTH786P - Machine Learning with Python Mid-term Project

Jui-Ting Hu 170376051
Mei Wang 180680771
Oya Kesgin 180840869

1 Introduction

Supervised learning is one of the machine learning systems to forecast a certain output from a given input. Through creating a model from these input and output data set can allow us to make a precise prediction for new data. There are two main tasks for supervised learning: classification and regression. The aim of classification is to predict a class label from predefined list of possibilities and the goal of regression is to predict a continuous number. However, some regression algorithms can be used for both classification and logistic regression is one of them. In this report, we will implement logistic regression by different models and optimization algorithms to classify MNIST digits as binary classification which distinguishes exactly two classes and multi-class classification which classify the image into one of the ten classes.

2 Data Exploration

Digits MNIST contains 70.000 handwritten images by high school students and employees of the US Census Bureau. The data set split into 60.000 images in training set and 10.000 images in test set. Each image is 28x28 pixels and has 784 features. Each feature represents one pixel's intensity and associated with a number between 0 and 255. 0 represents white and 255 represent a black pixel. The shades of grey are also defined with numbers between 0 to 255. Before processing the classification, we focused on the exploring data to understand better the data set. Firstly, we plotted some image digits for each label to look at their shapes (Figure 1). Then, we looked at the distribution plot of the labels to check whether there is a bias in training and test set towards certain label numbers. As can be seen, Digit 1 is slightly higher and 5 is lower count than others. There is a little bias because of this but overall the data set seem balanced (Figure 2).

Figure 1: MNIST Training Data Image

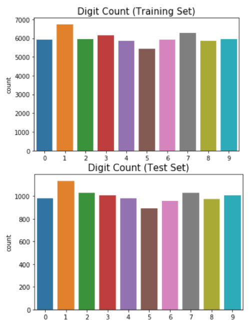


Figure 2: Digit Count for Each Label

3 Binary Classification

In this section, we will explain binary classification and optimization method to experience logistic regression before entering the multi-class model. The process and results of two models will be introduced: gradient descent and newton's method.

3.1 Gradient Descent

We used gradient descent to build our first model which is considered perhaps the simplest algorithm for unconstrained optimization. It is used to minimize cost function by iterative moving the direction of descending gradient or the word, the direction of steepest descent. Once the gradient is zero, it means that we reached a minimum. Each step tries to reduce the cost function until the algorithm converges to the minimum (Geron, 2017).

Binary classification logistic regression by gradient descent is built as following:

a) Filter the MNIST data set for binary classification

Firstly, we selected 6 and 8 to build our new training set and test set.

b) Derive the Sigmoid function

In order to transform the prediction into a probability in binary classification, we use the sigmoid function which is a continuous function. The output can take only two values; 0 and 1. In our model, if the estimated probability for digit 6 is greater than 50%, the output is 1, if the probability is lower than, the data is classified as 0 and in our case, it is 8.

$$\sigma(z) = \frac{e^z}{1+e^z}.$$

c) Derive the log-likelihood

We specified that cost function is log loss in our logistic classification. Because sigmoid function is not a linear and because of that the gradient descent algorithm will have a difficulty to find local minimum point. The other reason is, we need convex function to ease gradient descent to run and converge at the optimal minimum point.

$$-log(p(y \mid X, w)) = -\sum_{i=1}^{s}[y_i log(\sigma(\langle x_i, w\rangle) + (1 - y_i) log(1 - \sigma(\langle x_i, w\rangle))]$$

d) Derive the gradient with respect to each ω_k

We tweaked learning rate and iteration to find the best optimal minimum. In this model, we found that the best learning rate is 0.2 and iteration 3000. A good way to check if gradient descent runs properly is by plotting cost functions as the gradient descent run. In figure 3, after 1000 iteration we could say that it has started to converge.

$$w^{k+1} = w^k - \tau X^T(\sigma(Xw^k) - y)$$

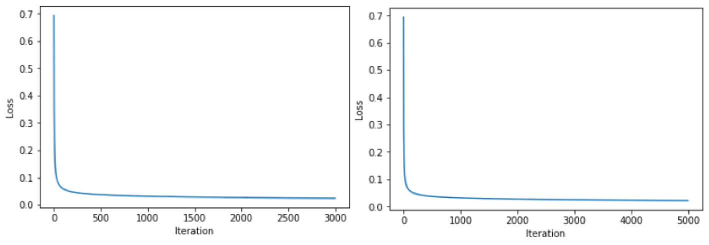
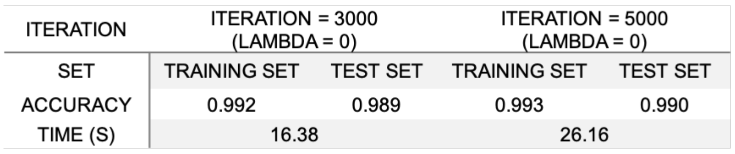


Figure 3: Loss of Binary Logistic Regression by Gradient Descent in 3000 and 5000 Iterations

The result of binary classification logistic regression by gradient descent:



ITERATION	ITERATION = 3000 (LAMBDA = 0)		ITERATION = 5000 (LAMBDA = 0)	
SET	TRAINING SET	TEST SET	TRAINING SET	TEST SET
ACCURACY	0.992	0.989	0.993	0.990
TIME (S)	16.38		26.16	

Figure 4: Accuracy of Binary Logistic Regression by Gradient Descent in 3000 and 5000 Iterations

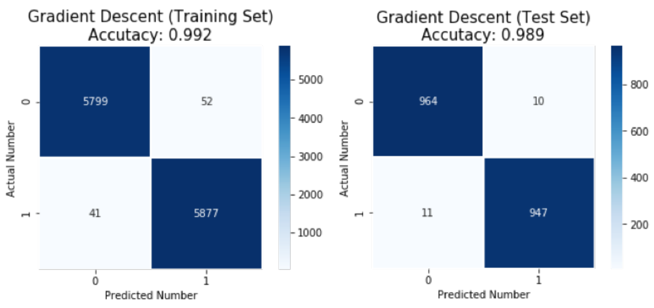


Figure 5: Confusion Matrix of Binary Logistic Regression by Gradient Descent in 3000 Iterations

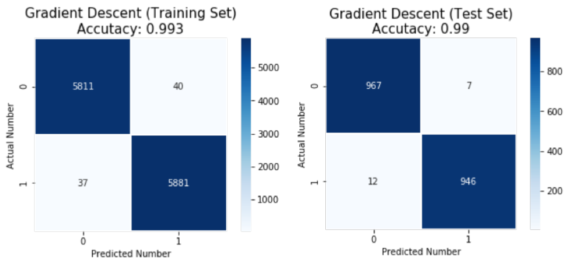
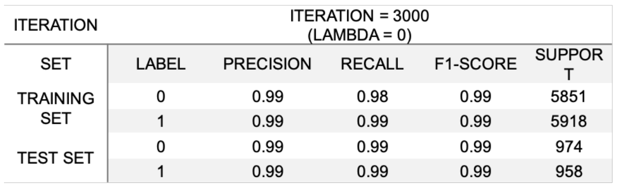


Figure 6: Confusion Matrix of Binary Logistic Regression by Gradient Descent in 5000 Iterations



ITERATION	ITERATION = 3000 (LAMBDA = 0)				
SET	LABEL	PRECISION	RECALL	F1-SCORE	SUPPORT
TRAINING SET	0	0.99	0.98	0.99	5851
	1	0.99	0.99	0.99	5918
TEST SET	0	0.99	0.99	0.99	974
	1	0.99	0.99	0.99	958

Figure 7: Precision, Recall, F1-score and Suppot of Binary Logistic Regression by Gradient Descent in 3000 Iterations

As seen above figures, the performance of the model is well. When we decrease the learning rate; iteration and time increases but its performance is better. 5000 iteration ended up in 27.5 second and its accuracy is 99.4%. However, 5000 iterations seem not necessary in this model. Because according to the result of the confusion matrix is not too different from the 3000 iterations. Also, we compute the precision and recall rates in 3000 iterations model which are high; 99% and same on training and test set.

3.2 Newton's method

Next, we try to build the second model for binary classification by newton's method. Comparing with gradient descent method uses the first derivatives in choosing suitable direction, newton uses the first and second derivatives. Ultimately, it performs better to find global minimum and converges in a few steps. Especially, this optimization algorithm needs inverting Hessian matrix and the inverse Hessian must be calculated at every training iteration.

Binary classification logistic regression by newton's method is built as following:

a) Derive hessian matrix function

In order to find the second partial derivatives, we take the partial derivative of each first partial, with respect to each parameter(Harrington, 2017). Our hessian is a square matrix of second-order partial derivatives of order 785x785.

$$H_E(w) = X^T S(w) X$$

b) Define inverse function

The below equation clearly shows that we need the inverse of hessian to calculate our update in Newton's method. Besides, because hessian is a matrix, we use the inverse instead of taking reciprocal.

$$w^{k+1} = w^k - H_E(w^k)^{-1}(\nabla E(w^k))$$

c) Derive the sigmoid function and the log-likelihood

With the above steps, the update in every iteration can be solved. And then, using the same concept to build the sigmoid function and log-loss function. Through the outer loop using Newton's method, we found that just need 8 iterations and 1e-8 learning rate can obtain the best result in this model.

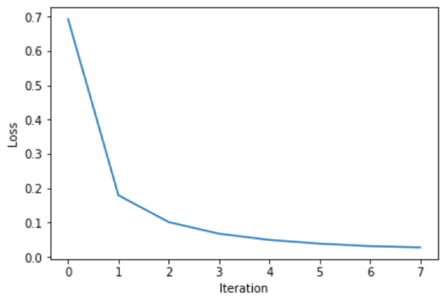
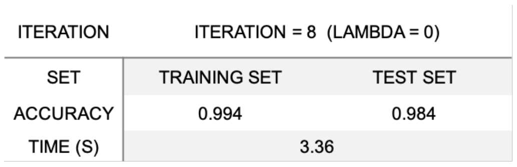


Figure 8: Loss of Binary Logistic Regression by Newton's Method

The result of binary classification logistic regression by newton's method:



ITERATION	ITERATION = 8 (LAMBDA = 0)	
SET	TRAINING SET	TEST SET
ACCURACY	0.994	0.984
TIME (S)	3.36	

Figure 9: Accuracy of Binary Logistic Regression by Newton's Method

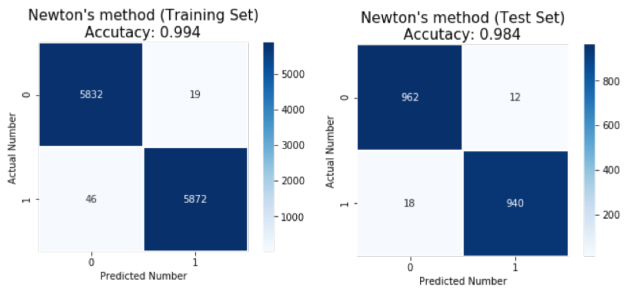
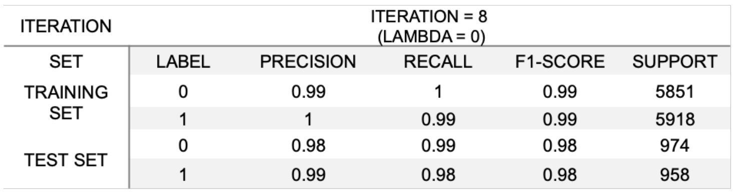


Figure 10: Confusion Matrix of Binary Logistic Regression by Newton's Method



ITERATION	ITERATION = 8 (LAMBDA = 0)				
SET	LABEL	PRECISION	RECALL	F1-SCORE	SUPPORT
TRAINING SET	0	0.99	1	0.99	5851
	1	1	0.99	0.99	5918
TEST SET	0	0.98	0.99	0.98	974
	1	0.99	0.98	0.98	958

Figure 11: Confusion Matrix of Binary Logistic Regression by Newton's Method

This model does train surprisingly faster than the gradient descent which we used previously. It only takes 8 iterations and 3 seconds. The running time more quickly than the model of gradient descent at least 10 seconds. In addition, the result is very similar to gradient descent when we run the confusion matrix. Precision and recall are maintained still 99%.

4 Multi-class Classification

After successfully classify two digits, we move on to extend the binary logistic regression model to a multi-class classification model which can distinguish between more than two classes. We apply the gradient descent to the logistic regression again. However, in this model, two new functions will be required, namely softmax and one - hot encoding which specially deal with the multiclass. Subsequently, the procedure of model will be illustrated first and the result will be presented.

4.1 Gradient Descent

Multi-class classification logistic regression by gradienr descent is built as following:

a) Derive the softmax function

The softmax function which is used in various multiclass classification methods can compute the probability distribution of the output. These probabilities for each class that sum to 100% (Juliani, 2016)

$$P(y_i = k \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{j=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$

b) Convert the integer classes coding

We use a function called one - hot encoding to change the classes format. Through this function, each categorical variable becomes a group of bits among which values are combined with a single 1 bit and all the others 0(David and Sarah, n.d.). For example, if $y_i = 5$, it will be changed to [0.0.0.0.0.1.0.0.0.0.].

c) Derive the log-likelihood and the gradient with respect to each ω_k

In this part, they are the same concept with the binary classification. Therefore, we follow the binary classification model to build these two function and compute the loss and gradient. After trial and adjustment, the 300 iterations and 1e-5 learning rate can achieve the highest accuracy in this model.

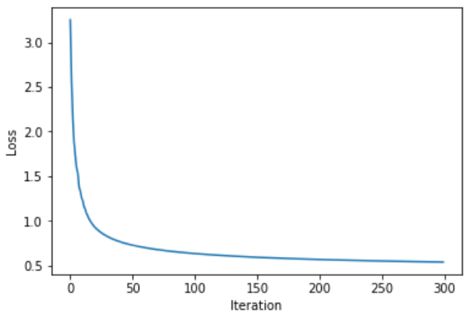
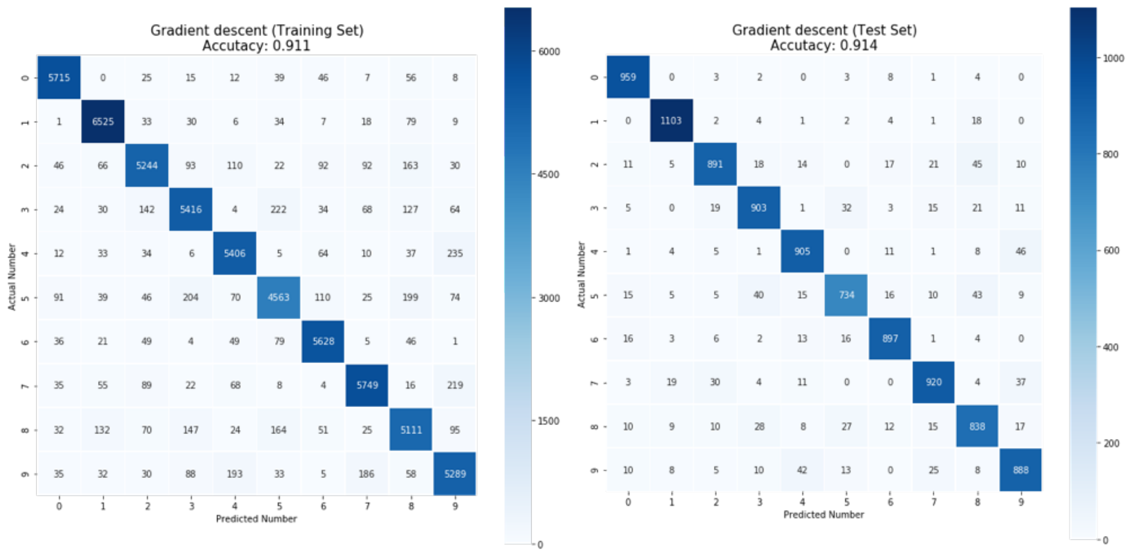


Figure 12: Loss of Multi-class Logistic Regression by Gradient Descent

The result of Multi-class classification logistic regression by gradienr descent:

ITERATION	ITERATION = 300 (LAMBDA = 0)	
SET	TRAINING SET	TEST SET
ACCURACY	0.911	0.914
TIME (S)	170.22	

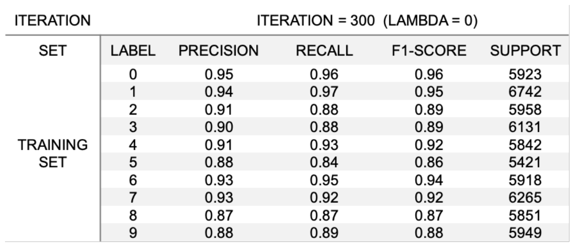
Figure 13: Accuracy of Multi-class Logistic Regression by Gradient Descent



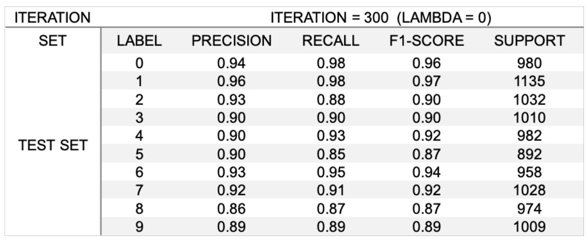
	0	1	2	3	4	5	6	7	8	9
0	5715	0	25	15	12	39	46	7	56	8
1	1	6525	33	30	6	34	7	18	79	9
2	46	66	5244	93	110	22	92	92	163	30
3	24	30	142	5416	4	222	34	68	127	64
4	12	33	34	6	5406	5	64	10	37	235
5	91	39	46	204	70	4563	110	25	199	74
6	36	21	49	4	49	79	5628	5	46	1
7	35	55	89	22	68	8	4	5749	16	219
8	32	132	70	147	24	164	51	25	5111	95
9	35	32	30	88	193	33	5	186	58	5289

	0	1	2	3	4	5	6	7	8	9
0	959	0	3	2	0	3	8	1	4	0
1	0	1103	2	4	1	2	4	1	18	0
2	11	5	891	18	14	0	17	21	45	10
3	5	0	19	903	1	32	3	15	21	11
4	1	4	5	1	905	0	11	1	8	46
5	15	5	5	40	15	734	16	10	43	9
6	16	3	6	2	13	16	897	1	4	0
7	3	19	30	4	11	0	0	920	4	37
8	10	9	10	28	8	27	12	15	838	17
9	10	8	5	10	42	13	0	25	8	888

Figure 14: Confusion Matrix of Multi-class Logistic Regression by Gradient Descent



ITERATION	ITERATION = 300 (LAMBDA = 0)				
SET	LABEL	PRECISION	RECALL	F1-SCORE	SUPPORT
TRAINING SET	0	0.95	0.96	0.96	5923
	1	0.94	0.97	0.95	6742
	2	0.91	0.88	0.89	5958
	3	0.90	0.88	0.89	6131
	4	0.91	0.93	0.92	5842
	5	0.88	0.84	0.86	5421
	6	0.93	0.95	0.94	5918
	7	0.93	0.92	0.92	6265
	8	0.87	0.87	0.87	5851
	9	0.88	0.89	0.88	5949



ITERATION	ITERATION = 300 (LAMBDA = 0)				
SET	LABEL	PRECISION	RECALL	F1-SCORE	SUPPORT
TEST SET	0	0.94	0.98	0.96	980
	1	0.96	0.98	0.97	1135
	2	0.93	0.88	0.90	1032
	3	0.90	0.90	0.90	1010
	4	0.90	0.93	0.92	982
	5	0.90	0.85	0.87	892
	6	0.93	0.95	0.94	958
	7	0.92	0.91	0.92	1028
	8	0.86	0.87	0.87	974
	9	0.89	0.89	0.89	1009

Figure 15: Precision, Recall, F1-score and Support of Multi-class Logistic Regression by Gradient Descent

In figure 3 to 5, the accuracy and the time are given. As our expected, multi-class classifier took longer time and got lower accuracy than binary classifier. And then, digit 3 and 5 and digit 4 and 9 are often confused. However, overall, we got not a bad result for this model because the precision, recall and f1-score of each lable are above 0.85 which is a high number.

5 Conclusion

In conclusion, after completing the Digits MNIST classifier, we also test our classifier model on Fashion MNIST dataset. There is no doubt that the Fashion MNIST classification would be more complex and difficult. However, we still obtain high accuracy. In binary classifier, it reaches around 98% and 96% in training set and test set. And then, in multi-class classifier, it still has about 80% accuracy in both sets.

References

Géron, A. 2017, Hands-On Machine Learning with Scikit-Learn and TensorFlow, 1st edn, O'Reilly Media, Inc.

Harris, D. Harris, S. 2007, Digital Design and Computer Architecture : From Gates to Processors, Elsevier Science Technology, Burlington.terdam;Boston;.

Müller, A.C. Guido, S. 2016, Introduction to Machine Learning with Python, 1st edn, O'Reilly Media, Inc.

Murphy, K.P. 2012, Machine learning: a probabilistic perspective, MIT Press, Cambridge, MA.

Raschka, S. Mirjalili, V. 2017, Python Machine Learning - Second Edition, 2;2; edn, Packt Publishing, GB.