COGS260 Assignment 3

Yan Sun
University of California, San Diego
La Jolla, California, USA
yas108@ucsd.edu

Abstract

Image classification is one of the most important research interests nowadays, for which MNIST and CIFAR-10 are well-known datasets for related study. In addition, IRIS is another dataset for the flower classification. Feed Forward Network and Convolutional Neural Network have become well-known methods to finish these tasks. In this assignment, several models and methods have been tried to classify the images in MNIST and CIFAR-10 dataset, including Feed Forward Network and Convolutional Neural Network. Other numerical methods are utilized to classify flowers data in IRIS dataset. During the process of implementing these methods or models, different parameters are tried in order to optimize the classification performance. Finally, optimized parameters for these models are able to make the classification accuracy as high as 99%.

1. Method

1.1. Least Square Estimation

Least Square Estimation Method is always used as a standard way in regression data fitting problem to approximate the solution, which usually minimizes the sum of square of residuals between the model prediction and target data[19]. For example, given the dataset $S = \{(x_i, y_i), i = 1, 2, 3 \cdots n\}$ when $X = [x_1, x_2, \cdots, x_n]^T$ and $Y = [y_1, y_2, \cdots, y_n]^T$. The task is to find the optimum W that could minimize the sum-of-squares error function g(W) in Equation 1:

$$g(W) = \|X \cdot W - Y\|_2^2 = (X \cdot W - Y)^T (X \cdot W - Y) \quad (1)$$

In order to get the solution, one of the most important steps is to calculate the gradient of g(W) for W. The expression of g(W) is transformed for the convenience of derivative process in Equation 2.

$$\begin{aligned} g(W) &= (X \cdot W - Y)^T (X \cdot W - Y) \\ &= W^T X^T X W - Y^T X W - W^T X^T Y + Y^T Y \end{aligned} \quad (2)$$

Based on the Matrix Calculus, the gradient of W in g(W) can be obtained in Equation 3.

$$\begin{aligned} \frac{\partial g(W)}{\partial W} &= X^T X W + X^T X W - X^T Y - X^T Y \\ &= 2X^T X W - 2X^T Y \end{aligned} \quad (3)$$

Then let the gradient of g(W) be zero, the closed form solution could be derived in Equation 4.

$$W^* = \underset{w}{argmin}\ g(W) = (X^T X)^{-1} X^T Y \quad (4)$$

Another pivotal part of least square estimation is to determine the expression of input X in the model. Before the estimation process, the most of actual situation for input X is always unknown, including determine the polynomial order relationship in the input data.

1.2. Parabola Estimation

Given the input data with specific formation such as $X = [x_1, x_2, \cdots, x_n]^T$ where $x_i = [1, x_i, x_i^2]$, f(x;W)=$x^T W$ and $W = [w_0, w_1, w_2]^T$. In this situation, the parabola estimation is needed to find the model to fit the given data[2]. Specifically, the loss function of this method could be L1 norm (Equation 5) or L2 norm (Equation 6) and solution could be obtained through closed form solution or calculated by gradient descent method.

$$g(W) = \|XW - Y\|_1 = \sum_{i=1}^{n} \mid y_i - f(x_i; W) \mid \quad (5)$$

$$g(W) = \|XW - Y\|_2^2 = \sum_{i=1}^{n} (y_i - f(x_i; W))^2 \quad (6)$$

In this assignment, closed form solution is utilized with L2 norm error function and gradient descent method is used for L1 norm error function. The gradient for L1 norm error function is in Equation 7 where the value of sign term is equal to 1 if $X^TW - Y > 0$, 0 if $X^TW - Y = 0$ or -1 if $X^TW - Y < 0$.

$$\begin{aligned}\frac{\partial g(W)}{\partial W} &= \frac{\partial \mid X^TW - Y \mid}{\partial (X^TW - Y)} \cdot \frac{\partial (X^TW - Y)}{\partial W} \\ &= sign(X^TW - Y) \cdot X\end{aligned} \tag{7}$$

1.3. Perceptron Learning

Generally, the perceptron learning algorithm is utilized for supervised learning of binary classifiers[4]. Specifically, the activation rule can be defined in Equation 8 and the whole training process is defined in Figure 1.

$$f(x) = \begin{cases} 1, & if \quad w^Tx + b \geq 0 \\ -1, & otherwise \end{cases} \tag{8}$$

Data: training data points $\mathbf{X}$, and training labels $\mathbf{Y}$;
Randomly initialize parameters $\mathbf{w}$ and b; pick a constant $\lambda \in (0, 1]$, which is similar to the step size in the standard gradient descent algorithm (by default, you can set $\lambda = 1$).
while *not every data point is correctly classified* **do**
  randomly select a data point $\mathbf{x}_i$ and its label y_i;
  compute the model prediction $f(\mathbf{x}_i)$ for $\mathbf{x}_i$;
  if $y_i == f(\mathbf{x}_i)$ **then**
    continue;
  else
    update the parameters $\mathbf{w}$ and $\mathbf{b}$:
    $\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - f(\mathbf{x}_i))\mathbf{x}_i$
    $\mathbf{b}_{t+1} = \mathbf{b}_t + \lambda(y_i - f(\mathbf{x}_i))$
  end
end

Figure 1. Perceptron Algorithm Process

1.4. Feed Forward Neural Network

The Feed Forward Neural Network is the one type of artificial neural network which could connect the neurons between different layers in the network and obtain the result in the forward propagation process without circle and update the parameters primarily via back propagation. This method could obtain wonderful performance on the regression or classification tasks on many datasets[16]. The specific update method during the back propagation process could be found in Equation 9 and Equation 10, where i,j and k indicates input layer, hidden layer and output layer, respectively.

However, there exist many hyperparameters that could be optimized[18]. As for the network architecture, the number of layers and number of neurons. For the data feeding method, the mini-batch technique could be optimized for how much data will be fed into the model. For training process, the optimizer and learning rate can be optimized. To sum up, these hyperparameters remained to be investigated for better performance on specific tasks.

$$w_{jk} := w_{jk} + \eta z_j^T \delta_k \qquad \delta_k = t_k - y_k \tag{9}$$

$$w_{ij} := w_{ij} + \eta x_i^T \delta_j \qquad \delta_j = g^{'}(a_j) \odot \delta_k w_{ij}^T \tag{10}$$

1.5. Convolutional Neural Network

Convolutional Neural Network (CNN)[9] is one type of feed-forward neural network that consists of input layers, hidden payers and output layers. In the hidden layers, there exist convolution layers, pooling layers and fully connected layers. The various combinations of these layers are able to create many types of CNN models, among which LeNet[10], VGG[17], AlexNet[9] and ResNet[3] are well-known models for image classification. Similar to Feed Forward Neural Network, CNN also has many aspects that worth investigating, which could assist in getting better results. These hyperparameters remained optimized could be the value of kernel size, stride, paddling or the definition of pooling method. Furthermore, the learning technique such as the selection of optimizer or the additional helpful tricks such as batch normalization and dropout are also research interest nowadays[20][6].

2. Experiment

2.1. Least Square Estimation

In the experiment of least square estimation, the closed form solution is used for the model fitting for given dataset. If the input feature is $[1, x_i]^T$, the ultimate solution will be a linear model for the given data, which is $Y = -15.47 + 11.61 * X$ (shown in Figure 2). If the input feature is $[1, x_i, x_i^2]^T$, the ultimate model equation is $Y = -1.71 + 3.02X + 0.87X^2$ (shown in Figure 3).

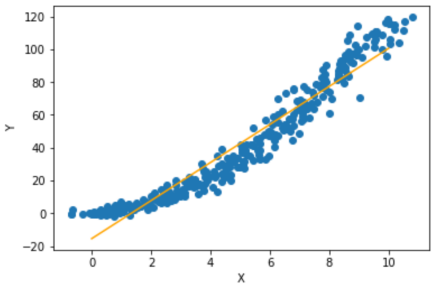

Figure 2. Least Square Estimation - Linear

2.2. Parabola Estimation

In the experiment for parabola estimation, there is some noise data added to the given dataset. Two types of solution have been tried. First, as for the L2 norm error function (Equation 6), the closed form solution is tried, the model

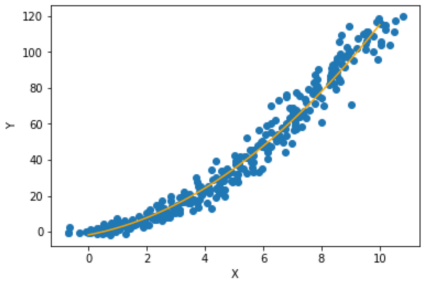

Figure 3. Least Square Estimation - Square

obtained is $Y = 51.07 + -16.06 * X + 2.36 * X^2$ (shown in Figure 4). Second, the L1 norm error function is also tried by gradient descent optimization method and the result is $Y = 1.69 + 1.12 * X + 1.05 * X^2$ (shown in Figure 5).

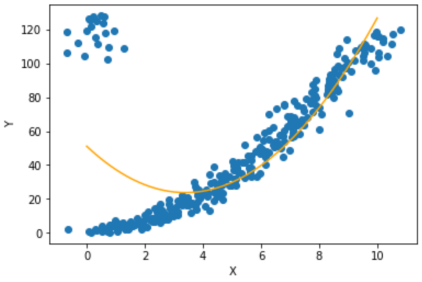

Figure 4. Parabola Estimation - L2 norm by closed form solution

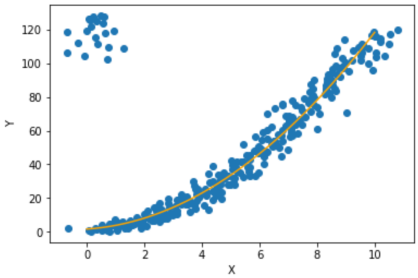

Figure 5. Parabola Estimation - L1 norm by gradient descent

2.3. Perceptron Learning

2.3.1 DataSet

The iris dataset[1] is selected as study object in this part of experiment. In order to study the relationship among given features, the pairplot process based on seaborn library [22] is utilized for the feature analysis (Figure 6).

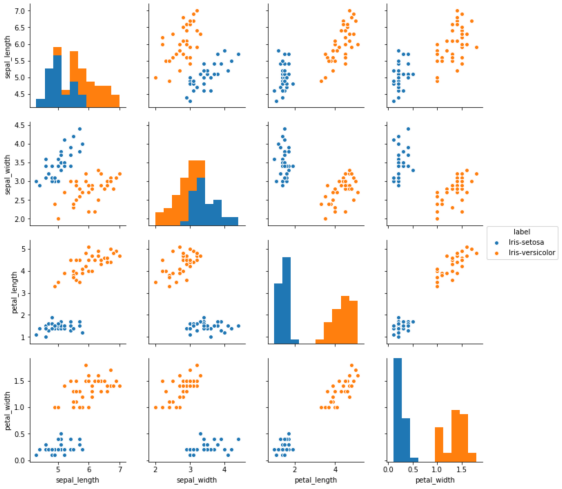

Figure 6. Pair Analysis for Iris Dataset

2.3.2 Model Fitting and Evaluation

Based on the algorithm shown in Figure 1, the optimized model for this task is trained by learning rate 0.001. Finally, the training process finished at 184 epoch (Figure 7 and 8).

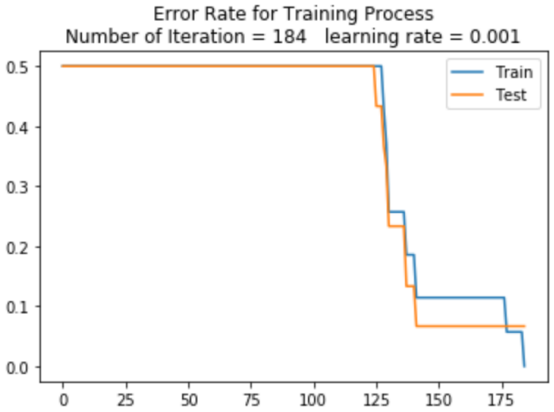

Figure 7. Loss Curve for Perceptron Learning

Assume x_1, x_2, x_3 and x_4 represents the input feature of sepal length, sepal width, petal length and petal width, respectively. Then the output decision boundary for this model is $0.21290549 - 0.54073087 * x_1 + 0.4174156 * x_2 + 0.38216564 * x_3 + 0.86654018 * x_4 = 0$. The final evaluation result for the optimized model is shown in Equation 11.

$$\begin{aligned} Accuracy &= 0.9333333333333333 \\ Precision &= 0.8823529411764706 \\ Recall &= 1.0 \\ F\_value &= 0.9375 \end{aligned} \tag{11}$$

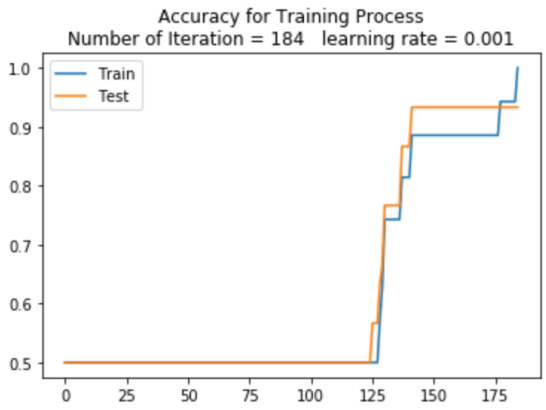

Figure 8. Accuracy Curve for Perceptron Learning

2.3.3 Z-Score

In this part, Z-Score technique is added to the data preprocess procedure. The train data and test data is transformed using the mean and standard deviation value calculated form train data. Then the following process is similar to Section 2.3.1 and 2.3.2. First, the pair analysis for the features in dataset after Z-score is shown in Figure 9.

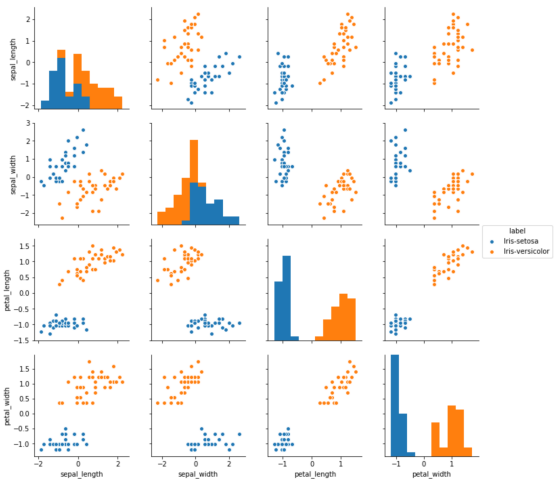

Figure 9. Pair Analysis for Iris Dataset with Z-Score

Then the training process for Z-scored input features is finished at 246 epoch with learning rate 0.001. The output curve for the whole training process is in Figure 10 and 11.

Assume x_1, x_2, x_3 and x_4 represents the Z-Scored input feature of sepal length, sepal width, petal length and petal width, respectively. Then the output decision boundary for this model is $0.31247015 + 0.30657108 * x_1 + 0.25808087 * x_2 + 0.6521715 * x_3 + 0.70696151 * x_4 = 0$. Ultimately, the evaluation result for this model is shown in Equation 12.

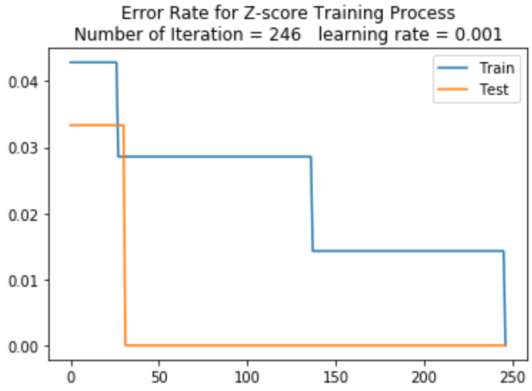

Figure 10. Loss Curve for Perceptron Learning with Z-Score

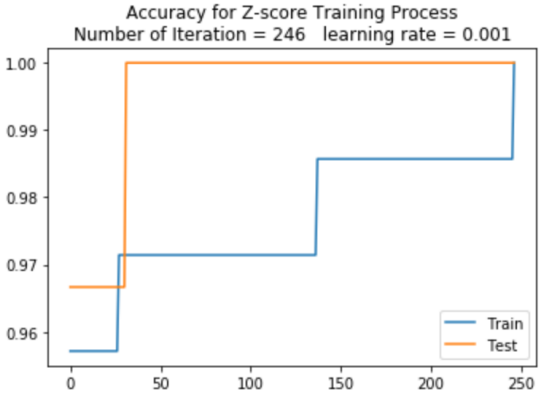

Figure 11. Accuracy Curve for Perceptron Learning with Z-Score

$$\begin{aligned} Accuracy &= 1.0 \\ Precision &= 1.0 \\ Recall &= 1.0 \\ F\_value &= 1.0 \end{aligned} \tag{12}$$

2.4. Feed Forward Neural Network

In this part of experiment, the Feed Forward Neural Network is built based on the principle explained in Section 1.4 guided by matrix calculation without using any deep learning framework.

The input feature is a 784 dimensional feature for the input image of MNIST dataset. Initially, the data is raw pixel value ranged from 0 to 255. In the preprocess step, all the pixel value has been transformed into the range [-1,1] via divided by 127.5 first and then minus 1. Then the dataset is randomly split into 50000 training data, 10000 validation data and 10000 test data. Every 256 data is randomly selected as one mini-batch for the training process. During the training process, the learning rate for hidden layer is 0.01 and for output layer is 0.00001.

2.4.1 One Hidden Layer

Based on the condition above, the number of neurons in the hidden layer is changed for several tries combined with minor change in the training parameter changed. The Table 1 shows the result of feed forward neural network with one hidden layer combined with different number of neurons. The final performance data is obtained within 1000 epochs.

Number of Neurons	Entropy			Accuracy		
	Train	Valid	Test	Train	Valid	Test
64	0.0200	0.0234	0.0233	0.9523	0.9351	0.9386
128	0.0097	0.0140	0.0133	0.9690	0.9513	0.9534
256	0.0037	0.0024	0.0024	0.9730	0.9722	0.9721

Table 1. Result of one hidden layer with different neurons

Finally, the one hidden layer model with 256 neurons gives the best result with 256 data per mini-batch, ReLU activation function, learning rate 0.01 for hidden layer and 0.00001 for output layer. The corresponded output curve for model with 256 neurons in hidden layer is shown in Figure 12 and 13.

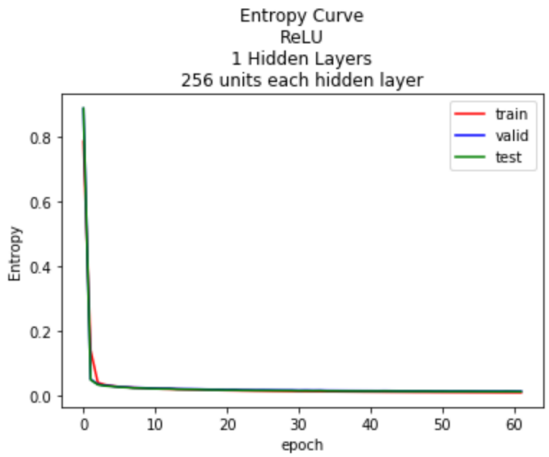

Figure 12. Loss Curve for Feed Forward Neural Network

2.4.2 Momentum and Regularization

This part of experiment is based on the configuration in Section 2.4.1.Initially, Momentum technique is added to the training process for one hidden layer model with 256 neurons in hidden layer. The momentum parameter alpha is set to be different value for study purpose (Table 2).

After finding the optimum configuration for momentum during the training process, the parameter for regularization (L2) is also studied, whose result is in Table 3. Different magnitude of regularization parameters have different influence on the training process.

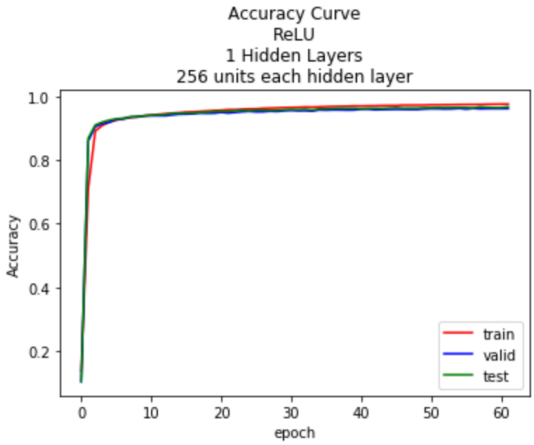

Figure 13. Accuracy Curve for Feed Forward Neural Network

Momentum Alpha	Entropy at 93% Test Accuracy			Accuracy at 93% Test Accuracy			Epoch for 93% Test Acc
	Train	Valid	Test	Train	Valid	Test	
0	0.0253	0.0253	0.0265	0.9490	0.9302	0.9307	54
0.5	0.0247	0.0249	0.0252	0.9493	0.9331	0.9301	36
0.9	0.0245	0.0250	0.0249	0.9501	0.9337	0.9301	28

Table 2. One hidden layer model with momentum

Regularization Lambda	Valid Entropy - Train Entropy Average	Epoch for 93% Accuracy
0	0.0013789	33
0.1	0.0009271	25
1	0.0566238	34
10	1.3234790	89

Table 3. Experiment Result for Regularization

2.4.3 Two Hidden Layers

Based on the condition above, the number of neurons in the hidden layer is changed for several tries. The momentum parameter alpha is set to be 0.9 and the L2 regularization parameter lambda is set to be 0.1 based on the experiment result in Table 2 and 3. Different combinations for the neuron numbers in first and second hidden layer have been tried. In order to evaluate their converge velocity, the performance for each model at 50th epoch will be extracted to be compared in Table 4.

id	1st Layer Neurons	2nd Layer Neurons	Entropy			Accuracy		
			Train	Valid	Test	Train	Valid	Test
1	16	8	0.08598	0.08824	0.08547	0.854	0.842	0.857
2	32	16	0.10359	0.10541	0.10300	0.898	0.888	0.899
3	64	32	0.14830	0.15063	0.14782	0.919	0.909	0.918
4	128	64	0.26927	0.27128	0.26953	0.938	0.930	0.937

Table 4. Two Hidden Layer Model at 50th Epoch

2.5. Convolutional Neural Network

In this part of experiment, Convolutional Neural Network (CNN) is used for the CIFAR-10 dataset[8] classification task. Primarily, AlexNet[9] is used for the main CNN

template for further modification or optimization on neural network architecture. In addition, other optimization methods have also been implemented for the training process improvement.

The main attention in this part is on the speed-up for the training process so one milestone is needed for the comparison among different models. Due to the limitation of computing resources, only 32 data is randomly selected as one mini-batch for the training process. After training for 5000 mini-batches with learning rate 0.001, the performance data will be extracted for comparison. Ultimately, the results are summarized in Table 5. Finally, the optimum modified AlexNet model is shown in Figure 14. The output curve during the training process is in Figure 15 and 16, after which the accuracy for test data could reach 82% .

id	Optimization Trick	Entropy			Accuracy		
		Train	Valid	Test	Train	Valid	Test
1	SGD	0.484	0.714	0.729	88.707%	75.910%	75.920%
2	Batch Normalization	0.485	0.680	0.726	86.925%	77.820%	77.130%
3	Average Pooling	0.472	0.717	0.720	86.082%	75.760%	76.000%
4	Adaptive Gradient	0.362	0.774	0.783	89.037%	74.190%	73.800%
5	Nesterov's Accelerate	0.459	0.738	0.728	88.218%	75.380%	76.130%
6	RMSprop	0.446	0.702	0.727	87.127%	76.520%	75.880%

Table 5. Results of Different speed-up tricks for CNN

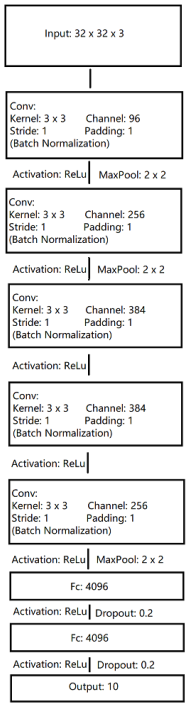

Figure 14. Modified AlexNet model

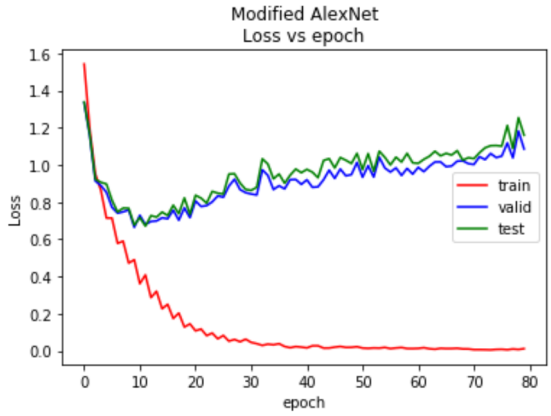

Figure 15. Modified AlexNet model Loss Curve

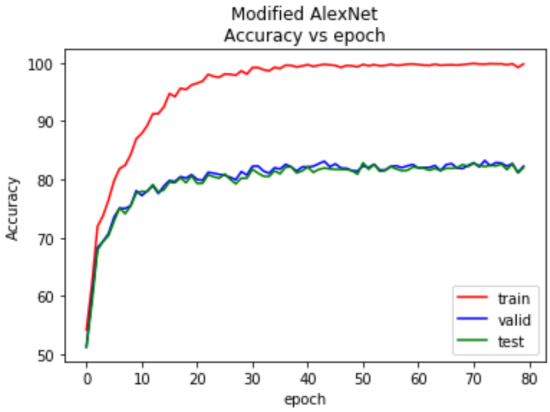

Figure 16. Modified AlexNet model Accuracy Curve

3. Discussion

3.1. Least Square Estimation

For the given dataset in this part of assignment and the estimation result in Figure 2 and 3, the least square estimation method has been proved having ability to obtain reasonable model to represent the given data. However, through the comparison of linear model and square model, it is obvious that the latter model is the better one for the given data. In this way, it can be concluded that the the selection of input feature is important in the least square estimation. When the dimension of the data goes higher and even difficult to visualize, the model selection technique[12] will be significant for least square estimation for further improvement.

3.2. Parabola Estimation

In this part of experiment, Figure 4 and 5 show the result for closed form solution on L2 error function and gradient

descent method on L1 error function. It can be observed that the former solution has been affected by the noise data while the latter model not, so the gradient descent method on L1 error function is better for the given dataset. For general situation, when the dataset is not large, it could be feasible to use closed-form solution. However, when the dataset is too large, the calculation for the matrix in the closed-form solution will require lots of time since it use matrix X directly, which means all the data will be utilized simultaneously. In order to solve the problem for large dataset, it would be better use gradient descent method to fit models.

3.3. Perceptron Learning

The Iris dataset is used in this part of experiment. Through the pair analysis in Figure 6. For each feature pair and their distribution plot, it is obvious there exist at least one line that could separate two classes in that 2-dimensional space, which means they are linearly separable. Using this algorithm, the final result for the Iris dataset classification is excellent (Equation 11).

After adding Z-Score process, the pair analysis plot has changed a lot. The distribution for the feature value has been changed to the standard distribution, which can be observed through the value of axis in Figure 9. Although the Z-Score technique make the training process a little bit slower, it make the ultimate performance better (Compare Figure 7 and 10). So the Z-Score process assist in optimizing the model. One of the most important reasons for this is that Z-Score alleviate the influence of the magnitude difference among different features. During the training process. if the input feature is not Z-Scored, their magnitude may lead to the difference of magnitude of weights in the model, making the model vulnerable to some noise data (e.g. data point with abnormal big value).

3.4. Feed Forward Neural Network

Different combinations of hyperparameters have been tried in this part of experiment. With only one hidden layer, 64, 128 and 256 neurons have been implemented into the model independently and the situation with 256 neurons turns out to be the best solution.

After adding momentum term during the training process, the training speed has been elevated (Table 2). The primary reason could be that momentum technique for the weight updating process would increase its ability to go over the local optimal area and make the change of the value of weight per epoch even larger[14]. Moreover, adding regularization term with different parameters have various effect. With the increase of the value of lambda, the value of entropy raises due to the additional term in cross entropy formula. When lambda is 0.1, the training process is accelerated and the difference between training set entropy and valid set entropy decreases, which indicates that the extent of overfitting has been alleviated. However, when lambda is 1 or 10, the training process goes slower and the difference between training set entropy and valid set entropy even increases. This result indicates that the choice for the value of lambda is significant for optimizing the model via regularization[13]. The situation when lambda is 0.1 is the better selection for this assignment.

As for the two-layers model, several combinations for the number of neurons in two hidden layers have been tried in Table 4. In the process of determining the number of neurons of hidden layers, one recommendation from other neural network design resources is selected as the guideline in this assignment, which indicated that the number of neurons should be the value between the number of nodes in input layers and output layers[5]. There also exist much assumptions but most of them has parameters remained to be optimized for determining number of neurons[15][7]. Among the results obtained for this assignment, the situation when 256 neurons in first hidden layer and 128 neurons in second hidden layer is the best choice. Nevertheless, more potential configurations for hyperparameters remained to be explored.

3.5. Convolutional Neural Netowrk

In this part, several tricks have been implemented in order to speed up the training process for the CIFAR-10 dataset classification task. The baseline model is the model in Table 5 with index 1, which just uses SGD optimizer.

For the model with index 2, Batch Normalization is added into the network and the model performance improved. The Batch Normalization will assist in gradient vanish or explode and even increase the converge velocity[6], which could be one of the reasons for the improvement.

For the model with index 3, the average pooling layer (2x2) is used to replace the last 2 fully connected layers in Figure 14. For one thing, it can reduce the computing requirement compared to the fully connected layers. For another, it is also easier to interpret and less prone to overfitting than traditional fully connected layers[11]. As for the result in this assignment, this method make the model improved a little compared to the baseline model, which basically shows the advantage of this method.

For the model with index 4, the SGD optimizer is replaced with Adaptive Gradient optimizer, for which the learning rate will shrink during the training process. The model with this optimizer does not show the improvement. It is possible that the optimum parameter for the shrinking rate has not been found, which could be possible reason for the comparably bad performance.

For the model with index 5, Nesterov's Accelerated Gradient is used combined with SGD optimizer. Compared to Momentum accelerate technique, this method has extra

term for the weight update. This term is the change for the gradient between current and previous step, which could be the similarity for the second order of loss function[21]. In this way, the training process could be faster, which can be observed that model 5 brings a little bit improvement compared to the baseline model for the test set.

For the model with index 6, RMSprop optimizer is used for the training process. the result of this model shows bad performance. Similar to the reason analysis for Adaptive Gradient Optimizer, the possible reason could be that the better parameters have not been found. In this assignment, when the learning rate is set to be 0.01, the accuracy barely improved due to the large learning rate. When the learning rate is set to be 0.001, the result is shown in Table 5 index 6. When the learning rate is set to be 0.0001, the training speed is even slower. This indicates the selection of learning rate is very sensitive for this task, which makes it hard to find a better parameter configuration.

To sum up, compared to the baseline model with SGD optimizer only, Batch Normalization, Average Pooling layer replacement and Nesterov's Accelerated Gradient technique could accelerate the training process for the task in this assignment.

4. Conclusion

In this assignment, Least Square Estimation, Parabola Estimation, Perceptron Learning, Feed Forward Neural Network and Convolutional Neural Network have been implemented for the classification for MNIST, Iris or CIFAR-10 dataset. Different types of parameter configuration or optimization tricks have been tried for study to improve the model performance. More possible configuration for the classification model or the training process remained to be found for further improvement.

References

[1] A. Asuncion and D. Newman. Uci machine learning repository, 2007. 3

[2] A. E. Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975. 1

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 2

[4] S. I. Gallant. Perceptron-based learning algorithms. *IEEE Transactions on neural networks*, 1(2):179–191, 1990. 2

[5] M. T. Hagan, H. B. Demuth, M. H. Beale, et al. *Neural network design*, volume 20. Pws Pub. Boston, 1996. 7

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2, 7

[7] B. Kosko. Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence/book and disk. *Vol. 1Prentice hall*, 1992. 7

[8] A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014. 5

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2, 5

[10] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, P. Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276, 1995. 2

[11] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 7

[12] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. 6

[13] A. Y. Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004. 7

[14] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999. 7

[15] M. Rafiq, G. Bugmann, and D. Easterbrook. Neural network design for engineering applications. *Computers & Structures*, 79(17):1541–1552, 2001. 7

[16] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. 2

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[18] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012. 2

[19] H. W. Sorenson. Least-squares estimation: from gauss to kalman. *IEEE spectrum*, 7(7):63–68, 1970. 1

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 2

[21] W. Su, S. Boyd, and E. Candes. A differential equation for modeling nesterovs accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, 2014. 8

[22] M. Waskom, O. Botvinnik, P. Hobson, J. Warmenhoven, J. Cole, Y. Halchenko, J. Vanderplas, S. Hoyer, S. Villalba, E. Quintero, et al. Seaborn: statistical data visualization. *URL: https://seaborn. pydata. org/(visited on 2017-05-15)*, 2014. 3