Project 1

Andrej Macko

March 2022

1 Introduction

The problem we were solving was a classification problem. We were trying to classify data in which of three categories they belong to. Types of categories are A, B, and C. We created a multi-layer neural network consisting of input (three neurons), hidden, and output layer (three neurons). We used feed-forward neural and used the back-propagation algorithm to calculate error and adjust weights. The training was done using batch gradient descent with and without momentum, mini-batch gradient descent with and without momentum, and adam. We used sigmoid, tahn and relu as activation function.

2 Hyper-parameters

Tested hyper parameters were number of neurons, activation function, optimizer, learning rate, batch size, and epoch. When momentum was used additional hyper-param β and with adam $\beta 1$, $\beta 2$, and ε.
Hyper parameters for adam $\beta 1 = 0.9$, $\beta 2 = 0.999$, and $\varepsilon = 10^{-8}$ are default values that we took from authors of adam algorithm. We tried different values but the overall error doesn't change. For entry every test these hyper parameters are same, they aren't changed.

3 Tests

3.1 Test

1. neurons hidden layer: 5
2. epochs: 500
3. learning rate: 0.001
4. batch size: 128
5. hidden layer activation function: sigmoid

6. output layer activation function: sigmoid
7. weight distribution: Uniform
8. momentum: false

Classification Error	Training error	Validation error
Mini-batch	33.19%	31.75%
Batch	17.08%	17.68%
Adam	25.91%	26.94%

We can see our model is performing poorly. We have just 5 neurons in hidden layer.

3.2 Test

1. neurons hidden layer: 20
2. epochs: 500
3. learning rate: 0.01
4. batch size: 128
5. hidden layer activation function: sigmoid
6. output layer activation function: sigmoid
7. weight distribution: Uniform
8. momentum: false

Classification Error	Training error	Validation error
Mini-batch	7.83%	7.50%
Batch	7.62%	8.07%
Adam	16.56%	17.88%

We changed amount of neurons in hidden layer from 5 to 20 and we can see classification error has decreased.

3.3 Test

1. neurons hidden layer: 20
2. epochs: 500
3. learning rate: 0.001
4. batch size: 64
5. hidden layer activation function: sigmoid
6. output layer activation function: sigmoid

7. weight distribution: Uniform
8. momentum: false

Classification Error	Training error	Validation error
Mini-batch	7.75%	7.75%
Batch	7.06%	7.56%
Adam	6.62%	6.00%

Changed batch size from 128 to 64 for model using adam error decreased.

3.4 Test

1. neurons hidden layer: 20
2. epochs: 300
3. learning rate: 0.01
4. batch size: 64
5. hidden layer activation function: sigmoid
6. output layer activation function: sigmoid
7. weight distribution: Uniform
8. momentum: false

Classification Error	Training error	Validation error
Mini-batch	2.55%	2.19%
Batch	36.28%	30.38%
Adam	1.50%	1.75%

We increased learning rate from 0.001 to 0.01. As learning rate was increased algorithms converge much faster and we don't need to wait for 500 epoch. But in batch gradient descent we are updating weights after we go trough whole data set algorithm can't converge that fast as mini-batch or adam.

3.5 Uniform vs. Normal distribution

In first test we were generating weights from uniform distribution. After switching distribution to normal train error and validation error are the same but what we observe is that algorithm converge much faster.

3.6 Momentum

1. neurons hidden layer: 20
2. epochs: 300
3. learning rate: 0.01
4. batch size: 64
5. hidden layer activation function: sigmoid
6. output layer activation function: sigmoid
7. weight distribution: Uniform
8. momentum: true

Classification Error	Training error	Validation error
Mini-batch	2.52%	2.31%
Batch	8.86%	10.75%
Adam	1.50%	1.75%

Adam optimization algorithm is using momentum in default settings. But we can see that batch with momentum decreased test error from 36.28% to 8.86% and validation error from 30.38% to 10.75% and mini-batch during training converge much faster.

3.7 Hidden layer tahn activation function

1. neurons hidden layer: 20
2. epochs: 300
3. learning rate: 0.01
4. batch size: 64
5. hidden layer activation function: tahn
6. output layer activation function: sigmoid
7. weight distribution: Uniform
8. momentum: true

Classification Error	Training error	Validation error
Mini-batch	2.48%	3.00%
Batch	16.94%	15.44%
Adam	1.75%	1.81%

Switching activation function in hidden layer from sigmoid to tahn we observe that tahn is computation expensive, training was slower and even error increased.

3.8 Hidden layer relu activation function

1. neurons hidden layer: 20
2. epochs: 300
3. learning rate: 0.01
4. batch size: 64
5. hidden layer activation function: relu
6. output layer activation function: sigmoid
7. weight distribution: Uniform
8. momentum: true

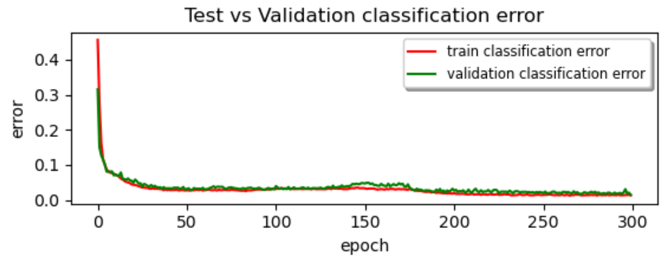
Classification Error	Training error	Validation error
Mini-batch	1.62%	2.06%
Batch	19.19%	19.56%
Adam	1.86%	1.98%

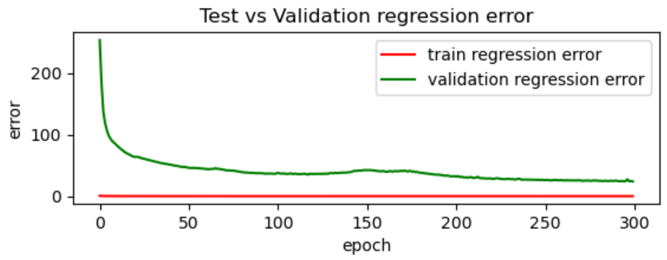
Switching activation function in hidden layer from tahn to relu we observe that relu is not computation expensive, training is much faster and error decreased.

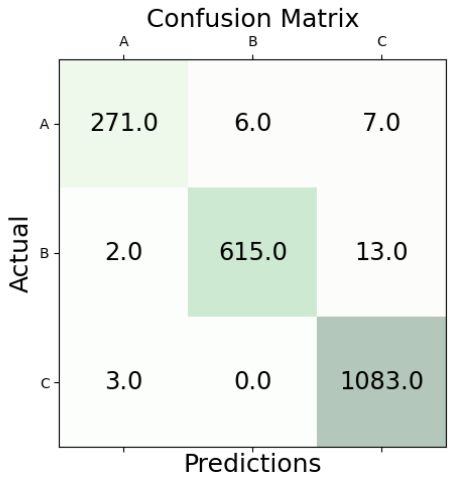
4 Best model

1. neurons hidden layer: 20
2. epochs: 300
3. learning rate: 0.01
4. batch size: 64
5. hidden layer activation function: relu
6. output layer activation function: sigmoid
7. weight distribution: Uniform
8. momentum: true

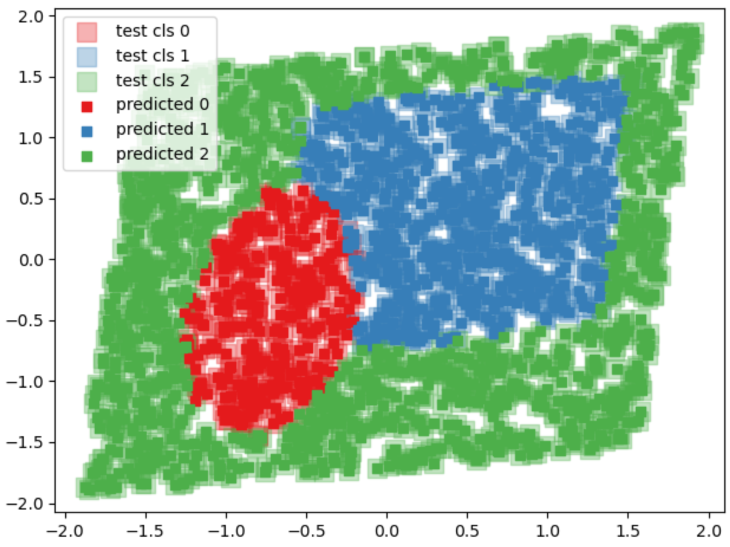
Classification Error	Training error	Validation error
Adam	1.39%	1.45%






Confusion Matrix

Actual \ Predictions	A	B	C
A	271.0	6.0	7.0
B	2.0	615.0	13.0
C	3.0	0.0	1083.0


test cls 0
test cls 1
test cls 2
predicted 0
predicted 1
predicted 2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0