BEN-GURION UNIVERSITY OF THE NEGEV

FACULTY OF ENGINEERING SCIENCE

DEPARTMENT OF INFORMATION SYSTEMS ENGINEERING

Deep learning course project #1:

Neural network from scratch on the MNIST images dataset

Submitted by:

Itay Gabbay

ID: 205784648

[itaygabb@post.bgu.ac.il](mailto:itaygabb@post.bgu.ac.il)

Omri Kaduri

ID: 311291462

[kaduro@post.bgu.ac.il](mailto:kaduro@post.bgu.ac.il)

Overall explanation

In our training process, we built a neural network as specified in the assignment.

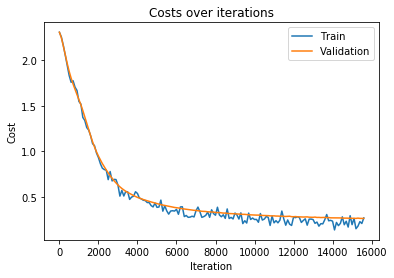
We splitted the data to train & validation sets, using randomly chosen indexes. The validation set size is 12000 samples (20% of the training data), and the training data is 48,000 samples.

In each epoch, we first divide our training data to mini-batches of fixed size, using randomly chosen indexes. We didn’t handle the case when the minibatch size isn’t divide the training set size. Then, on each iteration, we feed forward the network with the selected mini batch and then backpropagate. After that we compute the training & validation costs, and in each 100 iteration we save it for evaluation.

All data and code (CostsPerIteration.ipynb) for graphs reproduction exists in the zip file (at text files costs/costs\_normalization/costs\_dropout)

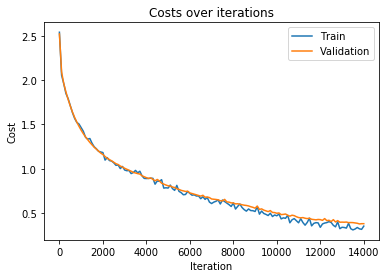
MNIST dataset without batch normalization and dropout:

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Early stopping iterations | 100 |
| Learning rate | 0.009 |
| Number of iterations | 15600 |
| Number of epochs | 195 |
| Batch size | 600 |
| **Train Accuracy** | **94.26%** |
| **Validation Accuracy** | **92.72%** |
| **Test Accuracy** | **93.07%** |



MNIST dataset with batch normalization:

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Early stopping iterations | 100 |
| Learning rate | 0.009 |
| Number of iterations | 14000 |
| Number of epochs | 175 |
| Batch size | 600 |
| **Train Accuracy** | **94.26%** |
| **Validation Accuracy** | **92.72%** |
| **Test Accuracy** | **93.07%** |



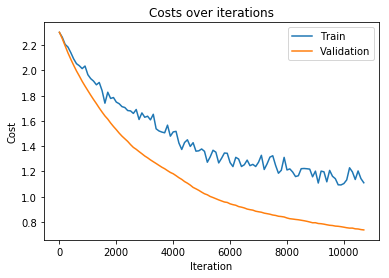
MNIST dataset with dropout

In order to add dropout, at feedforward we added apply\_dropout function which gets as an argument the activations of the layer and the rate of neurons to keep.

For the back propagation stage we saved the mask used during feed forward and applied the mask on the derivative of activations.

We tested dropout with number of rates. At 0.5 (a common first choice for dropout rate) we achieved 45% accuracy on the test. We found 0.8 and 0.9 to be a decent rate for us. The detailed results are:

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Early stopping iterations | 100 |
| Learning rate | 0.009 |
| Number of iterations | 10700 |
| Number of epochs | 135 |
| Dropout rate | 0.8 |
| Batch size | 600 |
| **Train Accuracy** | **87.55%** |
| **Validation Accuracy** | **87.42%** |
| **Test Accuracy** | **87.6%** |



|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Early stopping iterations | 100 |
| Learning rate | 0.009 |
| Number of iterations | 6200 |
| Number of epochs | 79 |
| Dropout rate | 0.9 |
| Batch size | 600 |
| **Train Accuracy** | **87.89%** |
| **Validation Accuracy** | **87.56%** |
| **Test Accuracy** | **88.68%** |

