Bio - Intelligent Algorithms - Exercise 1

Table of Contents

[Names 2](#_Toc73611114)

[Validation Score 2](#_Toc73611115)

[Note 2](#_Toc73611116)

[Scores 2](#_Toc73611117)

[CPU Accelerated (NumPy) 2](#_Toc73611118)

[GPU Accelerated (CuPy) 2](#_Toc73611119)

[Experiments 3](#_Toc73611120)

[Model architecture 3](#_Toc73611121)

[Hidden Layers 3](#_Toc73611122)

[Number of neurons 3](#_Toc73611123)

[Output layer 3](#_Toc73611124)

[Activation function 3](#_Toc73611125)

[Results 3](#_Toc73611126)

[Optimizer 4](#_Toc73611127)

[Results 4](#_Toc73611128)

[Regularization 4](#_Toc73611129)

[L2 regularization 4](#_Toc73611130)

[Dropout 4](#_Toc73611131)

[L2 regularization and Dropout Combination 4](#_Toc73611132)

[Results 5](#_Toc73611133)

[Learning rate decay 6](#_Toc73611134)

[Early stop 6](#_Toc73611135)

[Gaussian distribution initialization 6](#_Toc73611136)

[Hyper parameters tuning 6](#_Toc73611137)

# Names

* Simon Raviv, 312847478.
* Eliyahu Zilberbrand, 317937100.

# Validation Score

We have implemented KFold cross validation, therefore we will supply both validation score on the given validation dataset and the 10 KFold cross validation accuracy.

## Note

The predictions file supplied in this exercise, was predicted with the pre-trained **NumPy model**. Predicting with the CuPy model, will give slightly different prediction due to the difference in random state between the libraries.

## Scores

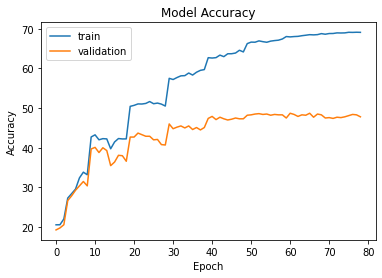
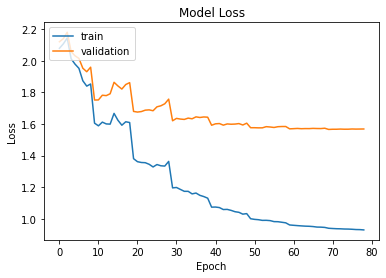
Due to different random implementation on the different modules, we have used, we will supply the results from both for the validation set and only the CuPy module KFold cross validation.

See notes regarding both accelerated implementations in *instructions\_highlits\_and\_more.docx* document.

### CPU Accelerated (NumPy)

* Given validation dataset accuracy:
  + **48.1 [%]**

### GPU Accelerated (CuPy)

* Given validation dataset accuracy:
  + **48.7 [%]**
* 10 KFold cross validation accuracy:
  + **47.82 [%]**

# Experiments

## Model architecture

### Hidden Layers

We got a small dataset; thus, we have chosen to use **2 hidden layers** architecture, and not more. We saw that when adding more hidden layers, the accuracy didn’t improve. This behavior seems expected to us, since when adding more layers, we add more learning parameters to optimize. When having small amount of data, we assume it makes harder to *learn and not to memorize*, which leads to overfitting on the training data.

### Number of neurons

We have experimented with various permutations: more neurons on the first layer, more neurons on the second layer and same number of neurons. In each permutation, we have changed the number of neurons in each layer. Finally, we have decided to use the same number of neurons on both layers. We have chosen to use **1500 neurons in each layer**.

### Output layer

We have chosen to use *Softmax* activation function for the output layer, since we are using categorial multi class labeled data. Using this method, we are getting the probability for each class.

## Activation function

We have experimented with various activation functions: *Sigmoid*, *ReLU* and *Leaky ReLU.*

We have chosen to use *ReLU*, since it had the highest accuracy. We assume that using *Leaky ReLU* will be more beneficial for architectures with multiple hidden layers, since it can handle the *dying gradients* problem.

The results below are without other features, e.g., regularization, learning rate decay.

### Results

## Optimizer

We have chosen to use Mini-Batch Gradient Descent and experimented with different amount of batch sizes. We have chosen to use **mini batch size of 65**, based on the experiments.

The results below are without other features, e.g., regularization, learning rate decay.

### Results

## Regularization

We have experimented with two types of regularization methods: L2 regularization and dropout.

In addition to that, we experimented with input noise, but didn’t see it helps beyond dropout.

We did it to make the model not to overfit on the training data, but to generalize.

### L2 regularization

We see that when increasing the L2 regularization factor, the overfitting decreases as the factor increases. Up to some boundaries.

### Dropout

We see that when decreasing the number of active neurons, the overfitting decreases. Up to some lower limit.

### L2 regularization and Dropout Combination

When combing both, we see that it is important not to exaggerate with the factor’s sizes.

We see that combining both regularizes when both not too intense, we see the best results.

### Results

|  |  |  |
| --- | --- | --- |
| **L2 Regularization** | **Dropout** | **Accuracy** |
| 0 | 0 | 38.80% |
| 1.5 | 0.7 | 37.40% |
| 1.3 | 0.2 | 41.70% |
| 0.001 | 0.8 | 38.70% |
| 0.001 | 0.2 | 40.80% |
| 0.001 | 0.4 | 47.50% |

## Learning rate decay

We have reached to a point we done optimizing the model using the technics mentioned in the previous sections and noticed the graphs of the accuracy and loss move with too high intense during the training. That also happened for last epochs; therefore, we have decided to implement the learning rate decay. Indeed, we saw it “smooths” the graphs (see graphs in *Validation* section), but there is no big impact after applying the regularization. We assume that it is not the bottleneck with the regularization. When not using regularization, we indeed saw that learning rate decay improves the results quite a few.

## Early stop

We have implemented early stop in order not to find the “correct” epoch to stop by hand. We have experimented a bit with how many epochs without improvement to stop parameter and decided to use 18, since we saw it gives the best results.

## Gaussian distribution initialization

We have experimented with some numbers and found that since our data is between 0 to 1, good approach is to use expectation value of 0 and standard deviation value of 0.01.

## Hyper parameters tuning

We wanted to implement hyper parameters tuning using Genetic/Evolutionary algorithm but didn’t have the time. We have hand tuned the parameters and those are the results in the validation section.