Nadav Yakar 036241818

Keren Babay 037974334

Evolutionary Algorithms – exercise 1:

Following is a general description of the code scheme and setup and architecture we chose, and its outcome. We are going to describe the backpropagation and evolutionary ways we used in order to train the neural networks over the mnist data set:

**Neural Network trained using backpropagation:**

**Flow**:

We have built a test-suit which went over several possible configurations, such as different layer sizes, learning rates, etc. and outputs the statistics and results. The network we chose to work with is the one we got the best results for.

Each layer is represented by a matrix of weights from the previous layers neurons (columns) to the current layer neurons (rows).

Activation – we got good results using the sigmoid activation function for the hidden layer output, and a softmax for the output layers neurons, used to predict the right class.

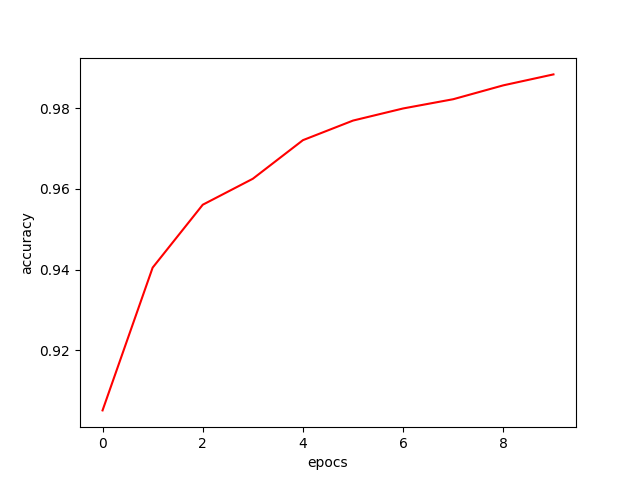
We have executed a validation check over a constant segment of a previously shuffled training set of a size of 0.2 of the total sum. During which, we have calculated the average accuracy over the validation set.  
We have used this intermediate data conducting my experiments, in order to indicate which configurations are worth continuing to run the network with, and which are wrong.

**Setup**:

* Architecture - the network was compounded of 784 (28x28) input neurons which got the data representing the pictures as an input, followed by a 100 neurons hidden layer, followed by a 50 neurons hidden layer, followed by a 10 output neurons layer, used to classify the data to the respective 1-hot-vector which represents each class.
* Initial weights – we initialized each weights matrix to a value uniformly distributed between -0.08 and 0.08.
* Learning rate – 0.01

**Outcome**:

Using this setup, we got a good average accuracy of above 98% over the validation set pretty fast (after 10 epocs):



Worth mentioning that on some occasions we got an underflow error using the sigmoid activation function, which have been solved using a thinner initial weight gap (we experienced it using a uniform init weights between -0.5 to 0.5).

**Neural Network trained using evolutionary algorithms:**

**Flow**:

We have trained the neural network with several possible configurations, such as layer sizes, elitism rates, number of examples form the train set to train a network, parent selection method, mutation rate, initial population method, different sizes of population and generations. We outputs the statistics and results. The network we chose to work with is the one we got the best results for.

Each layer is represented by a matrix of weights from the previous layers neurons (columns) to the current layer neurons (rows).

**Setup**:

* Architecture - the network was compounded of 784 (28x28) input neurons which got the data representing the pictures as an input, followed by 128 neurons hidden layer, followed by a 64 neurons hidden layer, followed by a 10 output neurons layer, used to classify the data to the respective 1-hot-vector which represents each class.
* 6000 samples per gen
* 100 population
* 20000 generations
* We initiate a network with tow layer sizes - The initial population was created using xavier glorot: sqrt(6 / (fan\_in + fan\_out))
* Network train- randomly selected 6000 examples from the train set to use by the current generation
* Activation: relu activations for faster results
* Breed method:
  + Elitism: 10% with the highest accuracy in the population was selected to continue to the next generation
  + As potential parents we removed the 10% worse accuracy networks
  + We implemented rank selection for choosing the parents, created 2 children’s – 50% from each parent.
  + Each child got mutation from -0.01 to 0.01
  + Add the children’s to the next generation

**Outcome**:

Using this setup, we got a good average accuracy of above 80% over the validation set (after 7000 generations):

**How to run:**

In file ev\_alg\_ex1\_nn.py change the following code to the name of the test files and place them in the code folder

**elif** dataset **is "testing"**:  
 fname\_img = os.path.join(path, **'t10k-images-idx3-ubyte'**)  
 fname\_lbl = os.path.join(path, **'t10k-labels-idx1-ubyte'**)

run main.py