MLP (Multilayer Perceptron)

lkathary@student.21-school.ru, rglorfin@student.21-school.ru

September 2022

1 MLP

- The program was developed in C++ language of **C++17** standard using g++ (GNU C++) compiler.
- GUI implementation, based on the **QT** library with API for **C++17**.
- The program provides the ability to form and train neural network models to classify handwritten Latin letters.
- The perceptron can:
  - classify images with handwritten letters of the Latin alphabet
  - have from 2 to 5 hidden layers
  - use a sigmoid activation function for each hidden layer
  - be able to learn on an open dataset (e.g. EMNIST-letters presented in the datasets directory). The ratio of the test sample to the training one should be no more than 2:8, i.e. the test sample makes no more than 20
  - show accuracy on a test sample over 70 percent
  - be trained using the backpropagation method
  - Work with mazes.
- The perceptron is implemented in two ways:
  - in matrix form (all layers are represented as weight matrices)
  - in graph form (each neuron is represented as some node object connected to other nodes by refs)
- The input data is normalized (by size and color) before neural network execution, in order to match the format of the EMNIST sample;
- The interface of the program provides the ability to:

- run the experiment on the test sample or on a part of it, given by a floating point number between 0 and 1 (where 0 is the empty sample - the degenerate situation, and 1 is the whole test sample). After the experiment, there should be an average accuracy, precision, recall, f-measure and total time spent on the experiment displayed on the screen
- load BMP images (image size can be up to 512x512) with Latin letters and classify them
- draw two-color square images by hand in a separate window
- start the real-time training process for a user-defined number of epochs with displaying the error control values for each training epoch. Make a report as a graph of the error change calculated on the test sample for each training epoch
- run the training process using cross-validation for a given number of groups k
- switch perceptron implementation (matrix or graph)
- switch the number of perceptron hidden layers (from 2 to 5)
- save to a file and load weights of perceptron from a file