1 Introduction

A 10-class classifier was developed to classify 28x28 images (samples) basing on their descriptions (labels)[1]. The classifier is a neural network that employs a hidden layer and a non-linear activation function. The structure is trained using backpropagation and/or regularisation.

2 The training algorithm

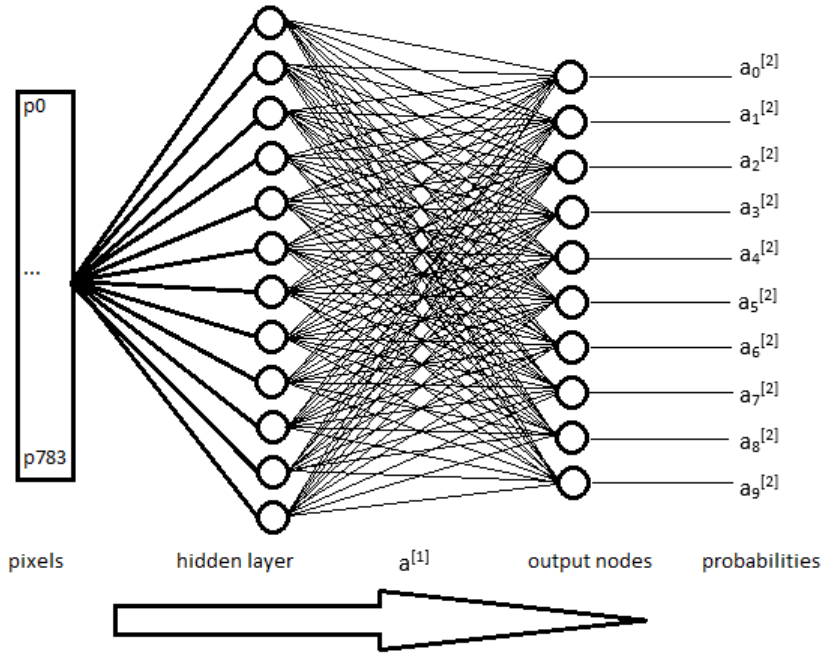

Figure 1: The structure

The neural network structure depends on the configuration parameters at the beginning of the FeedForwardNeuralNetwork Java class. Figure 1 shows the neural network structure defined by the following settings (tanh and atan are always mutually exclusive):

- NoOfNodes = 12, i.e. the number of nodes of the hidden layer
- alphaZero = *any*, i.e. the value of α_0 (defined below)
- tanh = true, i.e. the activation function of the hidden layer (if true)

[1]Train and test data were taken from the Fashion-MNIST dataset created by Zalando.

- atan = false, i.e. the activation function of the hidden layer (if true)
- perc = *any*, i.e. the percentage of hidden nodes not to be dropped during regularisation

The resulting structure is a 2-layer neural network with twelve nodes in the hidden layer and ten in the output layer. For every sample of the training set, all of its pixels are connected to all the twelve nodes of the hidden layer, and all the hidden nodes are then connected to all the ten nodes of the output layer. A vector of probabilities (one probability per label) is obtained at the end of the forward propagation. This vector is used to calculate loss, cost and gradients, which are necessary to update the weights and biases at the end of each iteration. at the beginning of the whole process, the weights are randomly initialised, while the biases are set at 0. The activation functions used in the hidden and output layers are respectively $tanh(x)$ and $softmax(x)$. The images of the dataset have been flattened (784-vectors) and matrices are used to avoid iterating on the training set:

- W_1 is the matrix containing the weights of the nodes in the hidden layer, one node per column (784 rows, 12 columns)
- b_1 is the matrix obtained by repeating the bias vector of the hidden layer once for each train sample (12 rows, 60000 columns)
- $trainData$ is the matrix with the 60000 train samples as columns (784 rows, 60000 columns)
- $Z_1 = {W_1}^{\mathrm{T}} trainData + b_1$ (12 rows, 60000 columns)
- $A_1 = tanh(Z_1)$ is the matrix with the twelve values computed by the hidden layer for each train sample (12 rows, 60000 columns)
- W_2 is the matrix containing the weights of the nodes in the output layer, one output node per column (12 rows, 10 columns)
- b_2 is the matrix obtained by repeating the bias vector of the output layer once for each train sample (10 rows, 60000 columns)
- $Z_2 = {W_2}^{\mathrm{T}} A_1 + b_2$ (10 rows, 60000 columns)
- $A_2 = softmax(Z_2)$ is the matrix with the ten final probabilities for each train sample (10 rows, 60000 columns)

At this point it is possible to calculate the cost as an average of all the losses coming from the probabilities in A_2. For every element in A_2, its loss is calculated through the binary cross-entropy function:

$$L = -(ylog(a) + (1 - y)log(1 - a))$$

where:
- a is the processed element of A_2

- y is 1 if the label associated to a is correct for the sample associated to a, 0 otherwise.
The gradients are calculated through the following formulas:

- Y is the matrix with the real probabilities (either 0 or 1) for each class, and for each sample (10 rows, 60000 columns)
- $dZ_2 = A_2 - Y$ (10 rows, 60000 columns)
- $dW_2 = \frac{1}{60000} dZ_2 {A_1}^{\mathrm{T}}$ (10 rows, 12 columns)
- $ones$ is the column vector containing 60000 1s (60000 rows, 1 column)
- $db_2 = \frac{1}{60000} dZ_2 ones$ (10 rows, 1 column)
- $g'(Z_1) = tanh'(Z_1)$ (12 rows, 60000 columns)
- $dZ_1 = W_2 dZ_2 * g'(Z_1)$ (12 rows, 60000 columns). * is the element by element multiplication
- $dW_1 = \frac{1}{60000} dZ_1 {trainData}^{\mathrm{T}}$ (12 rows, 784 columns)
- $db_1 = \frac{1}{60000} dZ_1 ones$ (12 rows, 1 column)

Before updating weights and values the matrices dW_i are transposed while the vectors db_i are transformed in matrices with 60000 columns, where each column is the previous db_i vector.
The updates are implemented by the following formulas:

- α_0 is the learning rate of the first iteration
- $\alpha = \alpha_0 * \frac{1001 - iterationIndex}{1000}$ is the learning rate
- $W_i = W_i - \alpha dW_i$
- $b_i = b_i - \alpha db_i$

The whole process described above is repeated at each iteration, until convergence is reached.

3 Output

After the last iteration, the trained structure is tested on a set of 10000 new samples. The cost of each training iteration and the accuracy on the test set are outputted in console, while the learning curve is plotted in a new window.