Contents

[1 Network configuration 2](#_Toc85636518)

[2 Forward functioning 2](#_Toc85636519)

[3 Backward functioning 2](#_Toc85636520)

[4 Network signals calculation 5](#_Toc85636521)

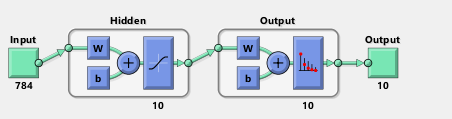
[5 Zero order pruning 5](#_Toc85636522)

[6 Pruning protocol 6](#_Toc85636523)

[7 Complete description 6](#_Toc85636524)

# Network configuration

We consider feedforward network with following structure



# Forward functioning

Input signals transformation

|  |  |
| --- | --- |
|  | () |

where vectors and value are defined once before network training.

The first layer transform data as

|  |  |
| --- | --- |
|  | () |

where is vector of biases and is matrix of weights: is weight of signal of neuron and activation function is

|  |  |
| --- | --- |
|  | (3) |

The output layer transform data as

|  |  |
| --- | --- |
|  | () |

where is vector of biases and is matrix of weights: is weight of signal of neuron and is softmax function:

|  |  |
| --- | --- |
|  | () |

where is maximal input signal of softmax function.

Cross entropy was used as loss function for NN training.

# Backward functioning

Let us consider one case with input vector and desired output class . In this case desired output distribution is

This means that loss function for case under consideration is

|  |  |
| --- | --- |
|  | () |

Derivative of loss function with respect to output signal

|  |  |
| --- | --- |
|  | () |

Let us denote

|  |  |
| --- | --- |
|  | () |

Now let us calculate

Now we need to calculate

For we have

For we have

In general we can write

Finally we have matrix

|  |  |
| --- | --- |
|  | () |

where

|  |  |
| --- | --- |
|  | () |

and is diagonal matrix with elements

|  |  |
| --- | --- |
|  | () |

Vector of derivatives can be calculated as

|  |  |
| --- | --- |
|  | () |

Let us take into account that vector contains only one nonzero element for .

This means that

Finally we can write vector of derivatives as

|  |  |
| --- | --- |
|  | () |

Derivatives of output of the first layer can be easily calculated from

Since we have

we can calculate derivative as

Now we are ready to write required derivatives

This means that vector of derivatives can be calculated as

|  |  |
| --- | --- |
|  | () |

Now we should find derivatives with respect to

We can note that

Finally we can rewrite derivative as

Required derivative can be rewritten through Hadamard product:

|  |  |
| --- | --- |
|  | () |

The last needed derivative is

This means that we need to calculate derivative

Finally we can write

The final formula is

|  |  |
| --- | --- |
|  | () |

The first level of sensitivity indicator is

where is the input signal number and is example number (record, instance, etc.).

To calculate average first order sensitivity indicator for one neuron we will use

|  |  |
| --- | --- |
|  | () |

# Network signals calculation

Input signal is correct class is . Calculations described for one example. Formulae appropriate for set of examples have white background and formulae that appropriate for one example only have yellow background.

| # | Formula | Eq | Formula | Eq |
| --- | --- | --- | --- | --- |
| 1 |  | (1) |  | (17) |
| 2 |  |  |  | (16) |
| 3 |  | (2) |  | (15) |
| 4 |  | (8) |  | (14) |
| 5 |  | (4) |  | (13) |
| 6 |  | (6) |  | (7) |

# Zero order pruning

We consider two methods of zero order sensitive indicator based pruning:

1. Weight removal.
2. Input signals removal.

Zero order sensitivity indicator of weight is absolute value of weight. The weight removal is removing weights in order of their absolute values.

We consider two zero order sensitivity indicator of input signals:

1. Average:
2. Maximal:

# Pruning protocol

1. Repeat
   1. Define candidate to remove on base of training set
   2. Remove
   3. Estimate validation set accuracy
2. Until current validation set accuracy is less than original accuracy minus tolerance.
3. Restore the last removed weight.
4. Estimate test set accuracy.

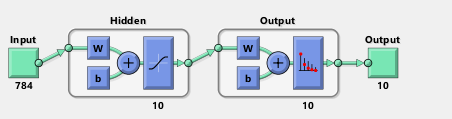
# Complete description

Used data set contains 10,000 grayscale images with resolution 28×28 pixels. This data set is subset of MNIST dataset <http://yann.lecun.com/exdb/mnist/>.

Used neural network contains 784 input signals with min-max normalisation into [-1,1] interval, 10 neurons in hidden layers with sigmoid activation function

and 10 softmax neurons in output layer.

100 NN were trained with different initial weights and the best one (with minimal validation set error) was selected for pruning.



We consider one of the classical pruning problem: reduction of input vector. Three different algorithms were used.

The first algorithm used classical zero order sensitivity indicators – absolute value of the hidden layer weights – and removed (set to zero) weight with minimal sensitivity indicators.

The second algorithm calculated mean of absolute values of all weights from one input signal and use this value as sensitivity indicator.

The third algorithm used the first order sensitivity indicators, presented in this paper: