Multilayer neural network

Juraj Mašlej

Content:

1. Model of neural network

We have decided to use multilayer network with input, hidden and output layer. Neurons were fully connected. Activation functions used were sigmoid for hidden layer and softmax for output.

2. Data

Data provided were 2-dimensional, 3. dimension was label. We splitted training set to estimation and validation sets with 80:20 ratio. Data were divided into 3 classes, A, B, C.
Final model was trained on test set of data. We used early-stopping to prevent over-training.

3. Preprocessing of data

Data were normalized by setting mean to zero and standard deviation to 1. Classes for data, A, B and C were encoded as vectors [1,0,0], [0,1,0], [0,0,1].

4. Layers and activation functions

We used input layer with 2 neurons and bias. Weights were initialized with values from normal distribution. Sigmoid function was used in hidden layer neurons. We used softmax for output layer. Output layer had 3 neurons, representing 3 classes of data.

5. Search for hyperparameters

Parameters we needed to find were, number of neurons in hidden layer, number of epochs to run our network and alpha – learning rate.

For number of neurons, we tried different values between 10 and 30, best result were usually obtained for network with number of neurons between 20 and 25.

Regarding epoch, training neural network longer than 250 epochs never provided better results. Since we started using early stopping, training was in most cases stopped after aproximately 100 epochs in the latest stage of network development.

Learning rates we were working with were between 0.02 and 0.2. Best results were obtained with values between 0.1 and 0.16.

6. Early stopping

During early testing we spotted that after certain number of epochs training error was reducing only slightly, but later on we got bigger error on validation set. That motivated us to implement early stopping.

Let E_train(t) be training error of epoch „t“,

E_val(t) = validation error,

E_opt = min(E_val(t, t + k)) = minimal E_val() encountered from „t“ to „t + k“,

GL = generalization loss

P = progress

k = hyperparameter, number of epochs we evaluate

PQ = ratio between progress and generalization loss

max_PQ = hyperparameter, if PQ is higher than max_PQ, we stop training

GL = 100*(E_val(t) / E_opt(t, t + k) - 1)

P = 1000* (sum(E_train(t, t + k) / k * min(E_train(t, t + k))) – 1)

PQ = GL(t) / P(t)

We needed to find hypeparameters, thus „k“, number of epochs after which we want to run test on validation set and get new E_val, and max_PQ. Higher PQ means less training progres and more validation error.
During search for best hyperparameters for neural network (not early stopping) we used k = 5 and max_PQ = 1.

We did not use early stopping while training model on all data in train_data for final evaluation in test dataset

Source: http://page.mi.fu-berlin.de/prechelt/Biblio/stop_tricks1997.pdf

7. Results

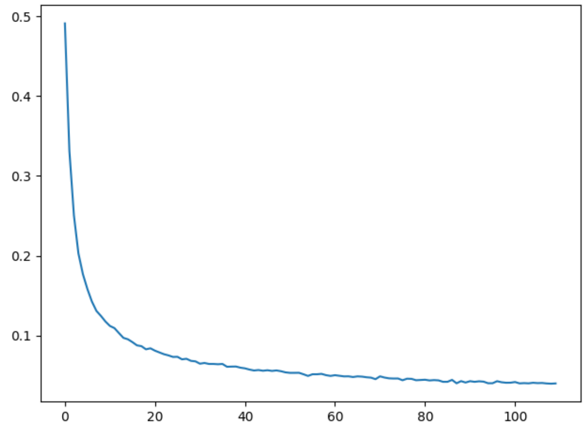
Word about notation: we used syntax eopchs:learning_rate:neurons as key to store best parameters. So 110:0.08:10 anywhere in files or code base means: model that ran 110 epochs at learning rate 0.08 with 10 neurons on hidden layer.

Best hyperparameters found : 110 epochs, 0.12 learning rate, 20 neurons
Error achieved on test set with these parameters : Ep 110/110: CE = 4.02%, RE = 2.35%
Where CE is mean squared error, RE is percentage of mis-classified examples

Graph of error vs epochs passed on final model:

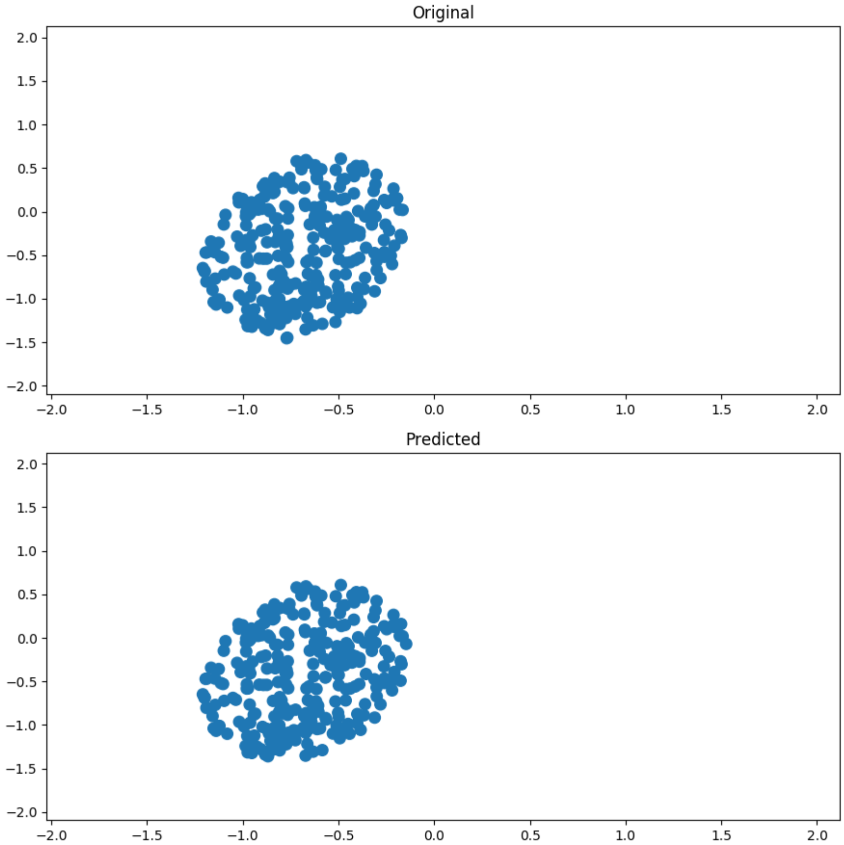
x – axis : epochs

y – axis : absolute error

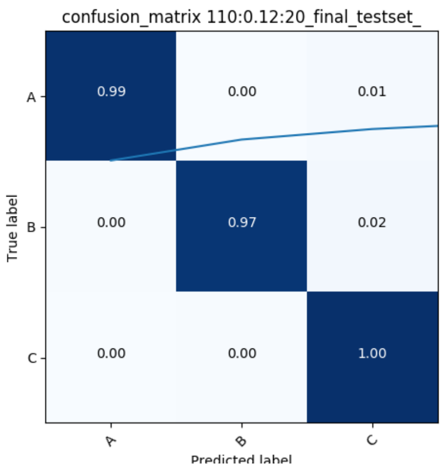


Predicted and original data:

Data were normalized, x position on x-axis, y position on y-axis.



7.1. Confusion matrix on test dataset:



Parameters used in search: epochs : 10, 110 , 210

learning rates : 0.12 , 0.16

neurons on hidden layer : 10, 20, 25

7.2. Table of estimation error

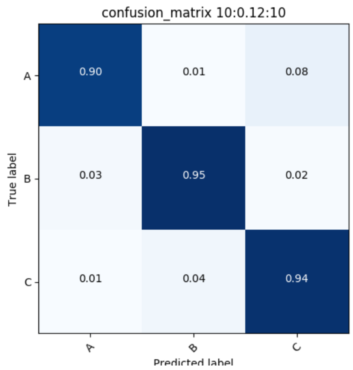
Notation: epoch:learning_rate:neurons, best model with highlighted.

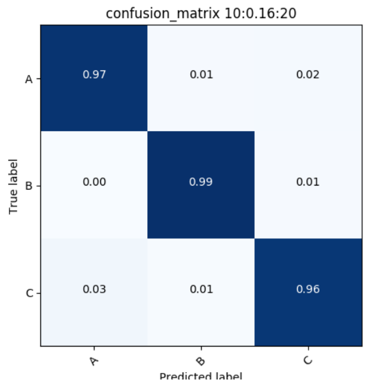
Model:	Error measured:
10:0.12:10	8.0401%
10:0.12:20	7.1577%
10:0.12:25	6.9464%
10:0.16:10	8.0747%
10:0.16:20	7.0067%
10:0.16:25	6.0404%
110:0.12:10	7.3142%
110:0.12:20	**2.7481%**
110:0.12:25	2.6278%
110:0.16:10	5.4568%
110:0.16:20	2.8722%
110:0.16:25	2.7136%
210:0.12:10	4.4090%
210:0.12:20	2.7110%
210:0.12:25	2.4889%
210:0.16:10	3.2759%
210:0.16:20	2.4593%
210:0.16:25	4.2544%

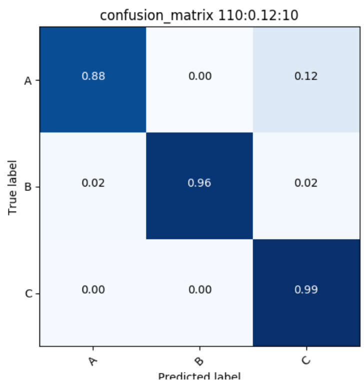
7.3. Table of validation error

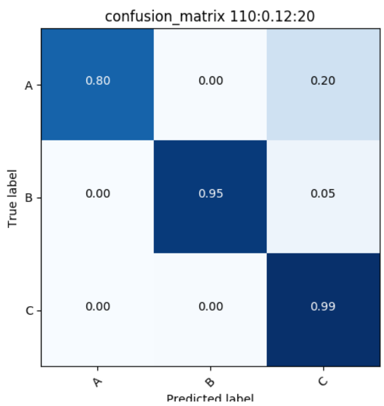
Model	Error measuered
10:0.12:10	4.5000%
10:0.12:20	4.2500%
10:0.12:25	6.2500%
10:0.16:10	4.8750%
10:0.16:20	6.6875%
10:0.16:25	5.1250%
110:0.12:10	5.8125%
110:0.12:20	**3.3125%**
110:0.12:25	3.2500%
110:0.16:10	5.2500%
110:0.16:20	4.0625%
110:0.16:25	4.0625%
210:0.12:10	4.5625%
210:0.12:20	4.1250%
210:0.12:25	3.8125%
210:0.16:10	3.8125%
210:0.16:20	4.0625%
210:0.16:25	4.6250%

7.4. Confusion matrix of models used in search for hyperparameters









8. Code base

main.py : main file, has function to test model parameters, create model, evaluate error, compute confusion matrix

data_handler.py : data loading, split data into different subsets

mlp.py : multilayer perceptron class, implement forward and backward propagation in neural network

classifier.py : main training of neural network, activation functions, predict labels for new dataset, early stopping implemented here,   mean squared error function, absolute error function, evaluate model.

9. Conclusion

Implementation of neural network with hidden layer was successful. We achieved error on data set new for network below 5%, thus, we achieved main goal. Most importantly, we have learned how to implement basic multilayer neural network and how to use backpropagation.