Assignment 3: Neural Networks

H. Blum, D. Cavezza, A. Paudice and M. Rohbeck
Machine Learning CO395
Imperial College London

November 28, 2015

1 Implementation

In our second assignment, we apply neural networks to the emotion recognition problem. We use the Neural Network Toolbox provided by MATLAB to train and compare the performance of different neural networks on the dataset at our disposal, in order to find the best training algorithm along with the best parameter configuration.
We compare four different training algorithm:

- Standard gradient descent backpropagation (`traingd` in MATLAB);
- Gradient descent with adaptive learning rate (`traingda`);
- Gradient descent with momentum (`traingdm`)
- Resilient backpropagation (`trainrp`).

In this section we describe our implementation of:

- selection of the best set of parameters for each algorithm;
- evaluation of NN's performance on unseen data.

1.1 Parameter selection

In the first part, we use cross- validation to select the best performing algorithm on the dataset and the best parameter configuration for it. Cross-validation is performed by splitting the dataset into 10 folds and using 9 folds for training and 1 for validation; iteratively, each fold is in turn used for validation, and ultimately the algorithm and parameter set that yield the best average performance over the folds is chosen.
For splitting, we use the same function as in the previous exercise, which performs *stratified* cross-validation: each fold contains approximately the same proportion of examples in every class as the whole dataset. It is implemented in the file `getFoldsPartitioning.m`.

1.2 Performance evaluation

2 Performance results

3 Questions

3.1 Part A: Questions

3.1.1 Question 1

We chose the optimal topology and parameters through cross-validation. In detail, in each iteration we tested the performance of the chosen topology on the fold used as validation set. In the end, we averaged the performances of each topology and parameters configuration and chose the setting that showed the best average performance.
We tested topologies with one and two layers. Topologies with one layer can fit any boolean function, while two-layer topologies can approximate arbitrarily well any real-valued function. For each layer, a common practice is to use a number of neurons between the sizes of the input and the output layer. Choosing less neurons than the output leads to a data compression that may cause information loss before reaching the output.
For the algorithms' parameters, we had to trade off the number of tests executed and the total time for testing. We chose

learning rates that covered different orders of magnitude where possible.
The optimal parameters for the different learning techniques are the following.

- **standard gradient descent:**
  Neurons per layer = 18, Number of layers = 2, Learning rate (LR) = 0.5, Avg. Error = 0.0458
- **adaptive gradient descent:**
  Neurons per layer = 15, Number of layers = 2, Learning rate = 0.1, LR decrease rate = 0.7, LR increase rate = 1.4, Avg. Error = 0.0449
- **gradient descent with momentum:**
  Neurons per layer = 42, Number of layers = 1, Learning rate = 1, Momentum coefficient = 0.9, Avg. Error = 0.0679
- **resilient backpropagation:**
  Neurons per layer = 14, Number of layers = 2, Delta increase = 1.3, Delta decrease = 0.5, Avg. Error = 0.0444

3.1.2 Question 2

3.1.3 Question 3

Matlab already implemented *early stopping* by default. Therefore the available data is divided into three subsets: the training, validation and test set. The first one is used to find the weights for the network. If the error on the validation set starts to increase for several iterations, which is an indication for overfitting, the process stops and the the weights which produce the minimum error on the validation set are returned.
Early stopping is not a 100% safe method as it is possible that the error on the validation set will decrease again after an unknown number of steps, but as it also reduces the training time and therefore comes somehow with negative cost, it is especially a good first approach for our (limited) computing power.

Furthermore, we considered to use *regularisation*. Regularisation is a mechanism to reduce the occurence of large weights. As large weights imply that some particular paths through the network are much more important than others, it could be more likely for overfitting to occur. Therefore, a function taking the input weights to a perceptron as an input is added to the original Error-Function that is minimized in the training. For example, in L2-Regularisation the squares of the weights are added together:

$$E_{new} = E_{original} + \lambda \sum_{all\ weights} w^2$$

λ is a parameter that sets the balance between the original minimization of the error and the minimization of the weights-function. Therefore, optimisation would include another parameter to optimise and therefore a multiplication of the computing time. Because the optimisation of our other parameters already took approximately two days, we could not spend triple/quadrupel/... of time.

Theoretically also other techniques, for example *dropout*, are possible. Also *Data Augmentation* by using the noisy dataset additionally to the clean dataset would be an option.

In the dropout-method, random network nodes are dropped out in every training iteration with a constant probability for each node. Therefore, the network architecture is randomized and every architecture is just trained for a small random subset of the training data. This intuitively reduces the chance that the network is overfitted to the training data set as actually the network with n nodes represents an average of all the before trained dropout-architectures (out of the set of 2^n possible networks), where each network node is only fitted to parts of the training data. On the other hand, with small amounts of training data, the dropout-technique can actually reduce the classification. In particular, the paper from Srivastava et al. referenced in the lecture slides shows that dropouts only reduce the classification error for MNIST classification training data sets of size 5000 or bigger, for 100 training examples the classification error was bigger than without dropout. Therefore, we are not confident that the dropout-technique would improve our performance whereas the implementation effort would be big.

3.1.4 Question 4

Disadvantages
First of all this leads to a necessary implementation of a decision function, which classifies on the basis of the outputs of the six networks. As in the assignment before, there is no obvious best solution for this problem and therefore the choice of decision function is another parameter to optimize.
The task for each of the six networks is to find a boolean classifying function. If we assume that we test the networks with the same topologies as the first big network, each network can represent *exactly* one boolean function as long as it

has at least two layers. This leads to a huge risk of overfitting compared to the one network which only *approximates* an arbitrary function and is therefore more robust to noise. Furthermore, as the parameters are optimised differently for each output class, each network is more specifically fit to the training data and this might be another risk of overfitting.
Moreover, training six networks requires more resources than training one slightly (only the output layer size differs) bigger network. In general, this corresponds to a significantly longer training time. Of coures, the time consumption can be reduced by parallelizing the process of cross-validation because the networks are independent (see advantages).

Advantages
As the networks are trained more specificly to recognize one class, this approach could lead to a better performance.
Because the output layer is smaller than for one big network, there are less weights to optimize in each network for the same topology. As a result, parallel cross-validation of six networks will be faster than the cross-validation of one big network, as long as one has the required computing power.

Combination of Outputs
The combination of the output of the six networks is exactly the same problem which was solved in our last assignment. These could be reused with small modifications:

- **random choice:** This is still possible by using a threshold value. Every output with a bigger value than the threshold is considered for the random pick.
- **score-based decision:** Naturally, the output of the networks is also a score, i.e. this is algorithm can be used without any modifications.
- **depth-based decision:** The depth of a tree is a specific property of a decision tree and therefore it is not easy to find a similar and meaningful property for neural networks. Of course you could use the number of layers, but as this varies just between two values, the algorithm would probably not perform well as there would be a lot of ties.
- **error-based decision:** As the error of the networks can be calculated just as the errors of the trees, this strategy is also possible to use without modification.

3.2 Question 1

3.2.1 Overfitting

Matlab already implemented *early stopping* by default. [some words about early stopping]. We considered using *regularization*, but this would include another paramenter to optimise. Because all the parameters already took us 2 days, we didn't want to double/triple this time. However, theoretically also other techniques [...] are possible.

3.2.2 6 Networks with single output

Disadvantages
First of all this includes to implement a decision function which classifies on basis of the outputs of the 6 networks (like for the decision trees).
The task for each of the 6 networks is to find a boolean decision function. If we assume that we test the networks with the same topologies as the 1 big network, these networks can represent *exactly* one boolean function as long as they have at least 2 layers. This leads to a huge risk of overfitting compared to the 1 network which approximates an arbitrary function and is therefore more robust to noise.

Furthermore, training 6 networks could lead to a higher risk of overfitting, as the parameters are optimized differently for each output class. Also, this is a time factor as long as one does not have access to 6 powerful computers. (It is possible to parallelize the process of cross-validation as the networks are independent).
Advantages
As the networks are trained more specific to recognize one class, this approach could lead to a better performance.
Because the output layer is smaller than for 1 big network, there are less weights to optimize in each network for the same topology. As a result, parallel cross-validation of 6 networks will be faster than the cross-validation of 1 big network.

Combination
The combination of the output of the 6 networks is exactly the problem solved in out last Assignment. We will just look at each discussed strategy:

random choice still possible with a threshold value (output a random choice between each network that outputs a value bigger than the treshold.

score-based decision naturally, the output of the networks is also a score, so this is possible without modifications

depth-based decision the depth of a tree is a specific property of a decision tree and therefore not easy to find a similar property for neural networks. Of course, one could use the number of layers, but as this varies just between two values, the algorithm would probably not perform well.

error-based decision as the error of the networks can be calcualted just as the errors of the trees, this strategy is also possible to use without modification

3.3 Question 2

3.3.1 a

Neural Networks have a significant higher accuracy than our decision tree algorithms. This is of course just an approximation based on our set of samples. It is impossible to judge the general performance of an algorithm just based on a set of samples. Also, this is an arbitrary choice of performance estimator and a simple split in 'better' and 'worse' is not possible as it depends on the problem one want's so solve. Furthermore, there are problems which can be solved better with other algorithms than neural networks (e.g. [...]).

3.3.2 type of t-test

paired t-test, because the samples of the accuracy are dependend. They are based on the same set of examples.

3.3.3 why not F1

As we have a classification into 6 classes, there are 6 F_1 measures for every algorithm. This leads to the problem described by the paragraph *Multiple Comparisions*. It basically means that we loose significance.

3.3.4 Tradeoff

less folds:
-¿ less samples -¿ smaller degree of freedom -¿ higher threshold -¿ more false negatives

more folds:
-¿ higher degree of freedom -¿ smaller threshold, higher value for t -¿ reect more often the null-hypothesis -¿ more false positives
We want to note that we have no reason to believe that 10 folds is the optimal number.

3.3.5 more emotions

decision tree: just have to add another tree and retrain all trees

neural networks: add as many neurons to the output layer as new emotions introduced, retrain the network including parameter optimization with cross-Validation (and fit the possible topologies)