Assignment 3: Neural Networks

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

November 25, 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 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; at 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 1 and 2 layers. Topologies with 1 layer can fit any Boolean function, while 2-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 standard gradient descent are:
Neurons per layer = 18; Number of layers = 2; Learning rate = 0.5; Avg Error = 0.0458
The optimal parameters for the adaptive gradient descent are:
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
The optimal parameters for the gradient descent with momentum are:
Neurons per layer = 42; Number of layers = 1; Learning rate = 1; Momentum coefficient = 0.9; Avg Error = 0.0679
The optimal parameters for resilient backpropagation are:
Neurons per layer = 14; Number of layers = 2; Delta increase = 1.3; Delta decrease = 0.5; Avg Error = 0.0444