Chalmers University of Technology

FFR135 - Artificial Neural Networks

Examples sheet 4

Jacopo Credi
(910216-T396)

October 19, 2015

Task 1a

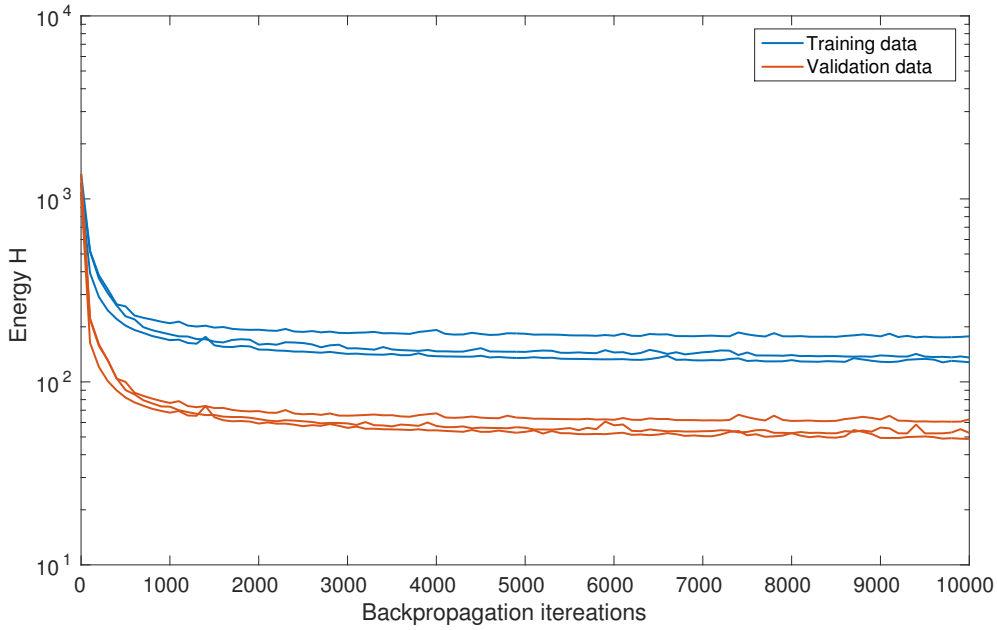

Figure 1: Energy (in log scale) vs iterations of backpropagation algorithm, for $N_G = 5$ Gaussian nodes. Training data in blue, validation data in red. For clarity, only three runs out of 20 are shown, each with only one point every 100 iterations. Parameters in the competitive learning phase: 10^5 iterations, learning rate $\eta_c = 0.02$, neighbourhood function width $\sigma = 0.1$. Parameters in the backpropagation phase: 10^4 iterations, learning rate $\eta_b = 0.1$, activation function $g(b) = \tanh(\beta b)$, with $\beta = 0.5$.

The provided data were classified using a neural network which combines unsupervised (competitive learning) and supervised learning (simple perceptron). For the competitive learning part, $N_G = 5$ Gaussian nodes were used. Given an input vector[1] $\boldsymbol{x}^\mu$, the winning neuron (denoted by i_0) is the unit whose activation function is maximum: $g_{i_0}(\boldsymbol{x}^\mu) \geq g_i(\boldsymbol{x}^\mu)$ for all $i = 1, \dots, N_G$. After determining the winning unit, all weights are updated, according to the following learning rule:

$$\delta \boldsymbol{w}_i \;=\; \eta\, \Lambda(i, i_0)\, (\boldsymbol{x}^\mu - \boldsymbol{w}_i)\ ,$$

where the neighbouring function $\Lambda(i, i_0)$ was taken to be $\Lambda(i, i_0) = \exp\left[-||i - i_0||^2/(2\sigma^2)\right]$. Then, the Gaussian nodes are used as input units of a simple perceptron, trained with backpropagation. In order to do this, the data were randomly divided in two parts: one for training (70%) and the other for validation (the remaining 30%).

Figure 1 shows the energy change as the backpropagation training is carried out. The energy quickly drops by a factor of ~ 8 in the first 2000 iterations, but then does not decrease further. The average final energy, with parameters in figure caption and $N_G = 5$ Gaussian nodes, is

$$H^{\text{final}}_{\text{training}} = (14 \pm 2) \cdot 10^1, \qquad \text{and} \qquad H^{\text{final}}_{\text{validation}} = (6.0 \pm 1.1) \cdot 10^1$$

Values are averages over 20 runs, with standard deviation. The ratio of the values reflect the size of the two datasets, as energy is extensive. We will later see (**2a**) that such values correspond to a rather poor classification performance.

[1] In this case, the data need not be normalised because we are not going to pass them (directly) through a sigmoid function, and there is no risk of zero derivative as in the perceptron case.

Task 1b

Final classification error (training set): 0.069, uncertainty: 0.010 Final classification error (validation set): 0.070, uncertainty: 0.014

UpdatePlots.m

```
function [] = UpdatePlots(tPlot, trainingErrorHandle, trainingError, ...
    validationErrorHandle, validationError, trainingEnergyHandle, ...
    trainingEnergy, validationEnergyHandle, validationEnergy)
%UPDATEPLOTS

plotTrainingError = get(trainingErrorHandle,'YData');
plotTrainingError(tPlot) = trainingError;
set(trainingErrorHandle,'YData', plotTrainingError);

plotValidationError = get(validationErrorHandle,'YData');
plotValidationError(tPlot) = validationError;
set(validationErrorHandle,'YData', plotValidationError);

plotTrainingEnergy = get(trainingEnergyHandle,'YData');
plotTrainingEnergy(tPlot) = trainingEnergy;
set(trainingEnergyHandle,'YData', plotTrainingEnergy);

plotValidationEnergy = get(validationEnergyHandle,'YData');
plotValidationEnergy(tPlot) = validationEnergy;
set(validationEnergyHandle,'YData', plotValidationEnergy);

drawnow;

end
```