Instituto Superior Técnico

Departamento de Engenharia Electrotécnica e de Computadores

Machine Learning

2nd Lab Assignment

Group Number: _____

Number_____ Name_____

Number_____ Name_____

Multilayer perceptrons

This assignment aims at illustrating the application of neural networks to pattern recognition, using supervised learning. In the first part, we'll train a multilayer perceptron to classify binary images of the digits from 0 to 9, with 5x5 pixels each. A training set of 400 patterns is used, corresponding to 40 images per digit. The network has 25 inputs, corresponding to the pixels of the input pattern, and has 10 outputs, corresponding to the ten digits. For each input pattern, the desired values for the network form a binary vector of 10 components, with the component that corresponds to the pattern's class equal to 1, and all other components equal to –1. When classifying an input pattern, the output with the highest value will represent the class that the network assigns to that pattern.

Throughout this assignment, network training will be performed using the gradient method in batch mode. The cost function will be the mean squared error,

$$C = \frac{1}{KP}\sum_{k=1}^{K}\sum_{i=1}^{P}\left(e_i^k\right)^2,$$

where K is the number of training patterns, P is the number of output components, and e_i^k is the ith component of the output error corresponding to the kth pattern.

1. Gradient method with fixed step size parameter and with momentum

1. Describe how the minimization of a function through the gradient method, with fixed step size parameter and with momentum, is performed. Be precise. Use equations when appropriate.

2. Launch the *neural_gui* environment, and create a network with 25 inputs (one for each pixel of the patterns that describe the digits) and 10 outputs (one for each digit class). The network should have one hidden layer with 15 units. Both the hidden and

the output layer should use the hyperbolic tangent as activation function (if necessary, right-click on each layer box and choose the appropriate activation function). In the input, -1 corresponds to a white pixel, and 1 corresponds to a black one. The desired values are as described above. The training data for the network are contained in the file *400dig.txt*.

- Select this file as the source of input and desired patterns for the network. To do that, right-click on the 'input' box, and enter the filename in the appropriate field.
- Switch off 'adaptive steps' and 'cost control' in the 'Training parameters' window (to open that window, right-click on the green 'training' box). Initially, set the momentum parameter to 0.5. Choose, as stopping criterion, the cost function reaching a value below 0.01 (this parameter appears in the 'desired cost' field, in the 'Training parameters' window).
- Train the network. Try to find a parameter set (step size and momentum) that approximately minimizes the training time of the network. Indicate the values that you obtained.

  Step size: Momentum:

- With the training performed in these conditions, click "Classify", and advance one pattern at a time, by clicking 'next' or 'random', checking whether the network has learned to classify most of the digits. If you want, you can also generate your own digits: press 'clear' to erase the current digit, and then click with the left/right mouse buttons on the digit visualization box.
- After closing the window that shows the classification, determine how many epochs it takes for the desired minimum error to be reached (execute at least five tests and compute the median of the numbers of epochs).

  Median of the numbers of epochs:

2. Gradient method with adaptive step sizes, momentum, and cost function control

1. Describe how the minimization of a function through the gradient method with adaptive step sizes, momentum, and cost function control, is performed. Be precise. Use equations when appropriate.

2. Activate ‘adaptive steps’ and ‘cost control’ in the ‘Training parameters’ window. Using as initial values of the step size and momentum parameters the values obtained in step 1.2 above, how many epochs are required to reach the desired minimum error? Make at least five tests and compute the median of the numbers of epochs.

   Median of the numbers of epochs:

3. Try to approximately find the set of parameters (initial step size and momentum) which minimizes the number of training epochs. Indicate the results that you have obtained (parameter values and median of the numbers of epochs for training). Comment on the sensitivity of the number of training epochs with respect to variations in the parameters, in comparison with the use of a fixed step size parameter. Indicate the main results that led you to your conclusions.

Next, we’ll use better ways of analysing the classification quality. We shall be using a test set with 160 digits, independent from the ones of the training set. To use this test set, right-click on the ‘input’ box and enter the filename *160dig.txt* in the ‘Test data’ field.

4. Train the neural network until the desired mean squared error is reached. Then, left-click on the ‘training’ box. This will print, in the main Matlab window, the values of the cost function in the training and test sets. Indicate the values that you obtained.

   Training error: Test error:

5. Now click on the ‘Confusion’ button. In the main Matlab window, two matrices called *confusion matrices* will appear, corresponding to the training and test sets,

respectively. Each matrix has 10 rows and 11 columns. The rows correspond to the desired values. The first 10 columns correspond to the classifications assigned by the neural network. Each row of this 10×10 submatrix indicates how the patterns from one class were classified by the network. In the best case (if the network reaches 100% correct classification) this submatrix will be diagonal. The last column of the matrix indicates the percentage of patterns of each class that were correctly classified by the network. The global percentage of correctly classified patterns is printed below each matrix.

Indicate the percentages of correctly classified patterns obtained by your network.

Training set: Test set:

6. In the 'Training parameters' window, set 'number of epochs' to 100 and 'desired cost' to 0. Keep on training the network and clicking on 'Confusion', until you reach 100% correctly classified patterns in the training set. Indicate the number of epochs that were necessary, and the percentage of correctly classified test set patterns at the end of this training process.

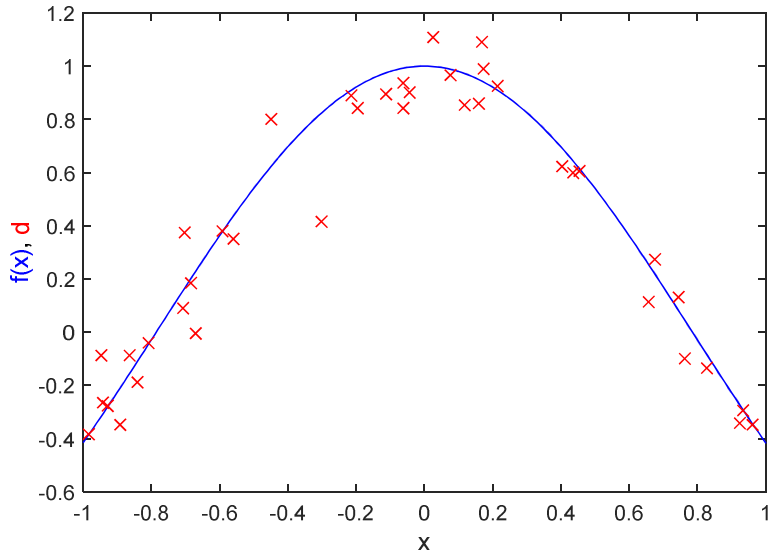
Number of epochs: Test set:

7. Which quantity (mean squared error, or global percentage of correct classifications) is best to evaluate the quality of networks, after training, in this problem? Why?

3 Use of cross-validation

To illustrate the use of cross-validation, we'll use a regression problem. We'll assume that we wish to estimate a function, of which we only have a small number of noisy observations. The function is $f(x) = \cos(2x)$, with $x \in [-1,1]$. We'll assume that we have noisy observations $d = f(x) + \varepsilon$, in which ε is Gaussian noise with a standard deviation of 0.1. The values of x will be used as inputs to the network, and the values of d

will be used as desired outputs. The file *f_train.txt* contains the training set, consisting of 40 randomly chosen values of x, together with corresponding values of d, generated with random ε. The files *f_valid.txt* (validation set) and *f_test.txt* (test set) are similar, but contain only 20 observations each. The following figure shows the function $f(x)$ $(blue\ line)$ and the data from the training set (red crosses).[1]



To prepare the network for handling this regression problem, right-click on the 'input' box and enter the names of the training, validation and test data files in the appropriate boxes; also set the numbers of inputs and outputs of the network to 1. Then, right-click on the 'training' box, and set 'number of epochs' to 1000, 'desired cost' to 0, and 'momentum' to 0.99 (for this problem, a momentum coefficient very close to 1 yields faster training). Finally, right-click on the rightmost 'layer' box and set the activation function of that layer to 'linear' (in the output layer, the 'tanh' function is more appropriate for classification problems, and the linear function is more appropriate for regression ones).

We'll use a network with just two hidden units, which is the minimum necessary to approximately represent $f(x)$. Right click on the leftmost 'layer' box, and set the number of units of that layer to 2.

[1] Note that in a real-life situation, we would have only a single set of data (80 observations, in this case), and we would have to decide how many to use for training, for validation and for testing.

1. Reset the network, and train it until the cost function is decreasing slowly (use at least 1000 epochs). Left-click on the 'Plot' button and then on the 'training' box. A figure (named 'Figure 3') will open, showing the evolution, during the training, of the cost function on the training set (blue line) and on the validation set (red line).[2] Sketch those functions below, and comment on their shapes. Are there signs of overfitting? Explain your answer.

2. Left-click on the 'training' box. This will report, in the main MATLAB window, the values of the cost function in the training, validation, and test sets. Indicate those values, together with the minimum that was attained in the validation set:

   Training cost:

   Validation cost: Minimum:

   Test cost:

3. Left-click on the 'Plot' button, and then on the 'error' box. Two figures will open, named 'Figure 1' and 'Figure 2'. You may disregard Figure 1. Figure 2 shows the training data (red crosses) and the function estimated by the network (blue line). Don't close that figure. Type *figure(2), hold on* in the main MATLAB window, to be able to plot more data in that figure later on.

[2] In most cases, the value of the cost function in the training set will have gone through one or more well visible minima, and will then have increased somewhat. There is a small probability that it will have kept always decreasing. If that is the case, re-initialize the weights with new random values, by clicking on 'Reset', and re-train the network. You may try several re-initializations, if you want, to convince yourself that, in most cases, the cost function actually goes through a minimum.

4. We'll now load into the network the weights from the iteration that yielded the lowest validation cost. For that purpose, left-click on the 'Best in valid.' button. Then, left-click on the 'training' box. Indicate, below, the values of the cost function corresponding to those weights. Compare them to the values obtained in step 3.2 above, and comment (include comments on the values obtained on the test set).

   Training cost:

   Validation cost:

   Test cost:

5. Left-click on the 'Plot' button, and then on the 'error' box. This will add to Figure 2 the plot of the function estimated with the current weights (the ones that yielded the lowest validation cost). Sketch the two estimated functions, identifying which is which. You don't need to sketch the training data.

6. Compare the two estimated functions with the true function, shown in Page 6 of this assignment, and comment.

7. When performing a training with cross-validation, how should one choose the weights that correspond to the result of the training process? Why?