Instituto Superior Técnico

Departamento de Engenharia Electrotécnica e de Computadores

Machine Learning

3rd Lab Assignment

Shift: 5ª 17h Group Number: 2

Number 81138 Name João Ramiro

Number 81567 Name José Pedro Boavida Miragaia

Multilayer perceptrons

This assignment aims at illustrating the applications of neural networks. In the first part we'll train a multilayer perceptron (MLP) for classification and in the second part we will train a MLP for regression.

This assignment requires MatLab's Neural Network Toolbox.

1 Classification

Our classification problem is a pattern recognition one, using supervised learning. Our goal is to classify binary images of the digits from 0 to 9, with 5x5 pixels each. The following figure illustrates some of the digits.

1.1 Data

Data are organized into two matrices, the input matrix X and the target matrix T.

Each column of the input matrix represents a different pattern or digit and will therefore have 25 elements corresponding to the 25 pixels in each image. There are 560 digits in our data set.

Each corresponding column of the target matrix will have 10 elements, with the component that corresponds to the pattern's class equal to 1, and all the other components equal to -1.

Load the data and view the size of inputs X and targets T.

```
load digits
size(X)
size(T)
```

Visualize some of the digits using the function show_digit which was provided.

1.2. Neural Network

We will use a feedforward network with one hidden layer with 15 units. 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.

```
net = patternnet([15]);
net.performFcn='mse';
```

Both the hidden and the output layer should use the hyperbolic tangent as activation function.

```
net.layers{1}.transferFcn='tansig';
net.layers{2}.transferFcn='tansig';
```

We will use the first 400 patterns for training the neural network and the remaining 160 for testing.

```
net.divideFcn='divideind';
net.divideParam.trainInd=1:400;
net.divideParam.testInd=401:560;
```

1.3. Gradient method with fixed step size parameter and momentum

(T) 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.

O método de *gradient descent* é usado para minimizar o risco empírico que consiste em $R = \frac{1}{n}\sum_{k=1}^{n} L\left(y^k, \hat{y}^k\right)$, de forma a adaptar os pesos w_{ij} ao conjunto de treino. Neste caso será usado o método de gradient descest com fixed step size, momentum term e o treino será feito em modo *batch*, por isso as equações que expressam a evolução dos pesos são as seguintes:

$$\begin{cases} w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \\ \Delta w_{ij}^{(t)} = -\eta \sum_k \frac{\partial L\left(y^k, \hat{y}^k\right)}{\partial w_{ij}} + \alpha \Delta w_{ij}^{(t-1)} \end{cases}$$

Onde w_{ij} corresponde ao peso do ramo com nó inical i e final j, (i=0 para DC)

Para calular o $\frac{\partial L}{\partial w_{ij}} = \varepsilon_i z_i$ nas equações acima é necessário primeiro fazer na rede uma propagação da rede para a frente de forma a obtermos os z_i e posteriormente uma propagação para trás (backpropagation) obtendo os ε_j dando nos todos os dados necessários para adaptar os pesos consoante o conjunto de treino.

Train the network using Gradient descent with momentum backpropagation. Initially, set the learning rate to 0.1 and the momentum parameter to 0.9. Choose, as stopping criterion, the cost function reaching a value below 0.05 or the number of iterations reaching 10000.

```
net.trainFcn = 'traingdm'
net.trainParam.lr=0.1; % learning rate
```

```
net.trainParam.mc=0.9;% Momentum constant
net.trainParam.show=10000; % # of epochs in display
net.trainParam.epochs=10000;% max epochs
net.trainParam.goal=0.05; % training goal
[net,tr] = train(net,X,T);
```

To see the evolution of the cost function (MSE) during training, click the "Performance" button.

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: 0.9 Momentum: 0.5

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: 235

1.4. Gradient method with adaptive step sizes and momentum

(T) Describe how the minimization of a function through the gradient method with adaptive step sizes and momentum is performed. Be precise. Use equations when appropriate.

A expressão permanece igual à representada na alínea 1.3 com uma pequena adição, o η é agora modificado a cada iteração segundo as seguintes equações:

$$\begin{cases} \eta^{(t+1)} = \eta^{(t)} * u \ , u > 1 \quad se \ \dfrac{\partial L}{\partial w_{ij}}^{(t)} * \dfrac{\partial L}{\partial w_{ij}}^{(t-1)} > 0 \\ \eta^{(t+1)} = \eta^{(t)} * d \ , 0 > \ d > 1 \qquad c.c. \end{cases}$$

Isto fará com que o valor inicial de η seja menos relevante pois este é modificado a cada iteração. O uso de adaptive step sizes permite então adaptação do η segundo características locais permitindo geralmente uma convergência mais rápida.

Choose as training method, gradient descent with momentum and adaptive learning rate backpropagation.

```
net.trainFcn = 'traingdx'
```

Train the network using the same initial values of the step size and momentum parameters as before. 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: 87

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.

Os valores encontrados que minimizam o número de épocas são 0.9 para o learning rate e 0.7 para o momentum term que correspondem a XXX epochs.
Para o mesmo momentum term (α) devido à robustez deste algoritmo, ao variar o learning rate o número de epoch é praticamente inalterado.
No caso em que não é usado passo adaptativo ocorrem grandes alteração o número de epochs ao variar minimamente os parâmetros, isto deve-se simplemeste ao facto de o η não se alterar consoante a região da função de custo que está a operar.
Assim concluí-se que ao usar passo adaptativo, o valor inicial de η tem pouca importância e que o algoritmo vai diminuir o erro quadrático mínimo mais rapidamente.

Next, we'll analyze the classification quality in the test set with a confusion matrix. This matrix has 11 rows and 11 columns. The first 10 columns correspond to the target values. The first 10 rows correspond to the classifications assigned by the neural network. Each column 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 row 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 at the bottom right cell.

```
x_test=X(:,tr.testInd);
t_test=T(:,tr.testInd);
y_test = net(x_test);
plotconfusion(t_test,y_test);
```

Comment on the values in the confusion matrix you obtained. Is the accuracy the same for every digit? Are the errors what you'd expect?

A percentagem geral com que a rede identifica corretamente os dígitos de entrada é cerca de 73% o que é satisfatório.

É importante referir que para dígitos diferentes obtêm-se percentagens diferentes, sendo que a causa destes erros advém das similaridades entre dígitos de números diferentes, por exemplo o dígito 8 é facilmente confundido com o 9 devido às suas partes superiores serem semelhantes ficando com taxas de erro da à volta de 60% e o 1 com o 7 devido a ambos terem regiões retilíneas.

É também relevante apontar que as percentagens de accuracy variam ao correr outro treino com os mesmos conjunto de treino e de teste devido a os pesos iniciais serem escolhidos aleatoriamente.

Write down the performance values that you obtained:

Training error: 0.05492 Test set Accuracy: 73.3%

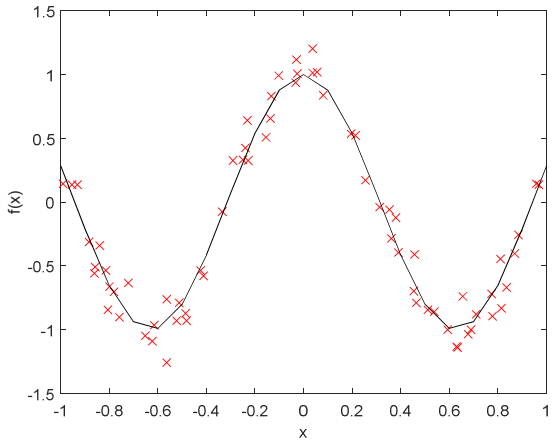
Which quantity (mean squared error, or global percentage of correct classifications) is best to evaluate the quality of networks, after training, in this problem? Why?

Ambos os parâmetros são relevantes no entanto é mais fácil e intuitivo de averiguar a qualidade da rede através do test set accuracy, pois nos é dito logo qual a percentagem de sucesso que é mais simples de compreender do que o Mean squared error.

2. Regression

The goal of this second part is to estimate a function and to illustrate the use of a validation set.

The function is $f(x) = \cos(5x)$, with $x \in [-1,1]$ for which we have a small number of 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 targets. The following figure shows the function $f(x)$ $(black\ line)$ and the data we will use for training (*red crosses*).



2.1 Data

Load the data in file 'regression_data.mat' and check the size of inputs X and targets T.

2.2 Neural Network

Create a network with a single hidden layer with 40 units. Set the activation of the output 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).

```
net = fitnet(40);
net.layers{2}.transferFcn='purelin';
```

Choose 'mse' as cost function and, as stopping criterion, the cost function reaching a value below 0.005 or the number of iterations reaching 10000.

2.3 Training with a validation set

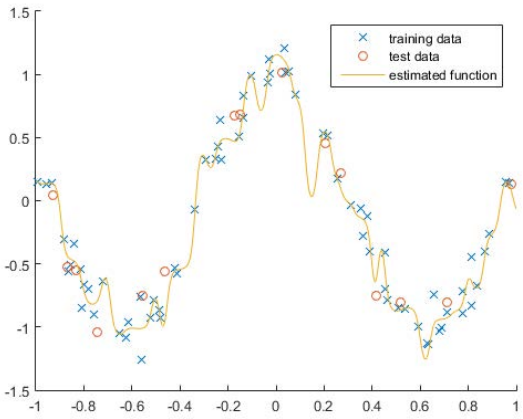
(T) When performing training with a validation set, how are the weights that correspond to the result of the training process chosen? What is the goal?

Os pesos são escolhidos de forma a obter um MSE mínimo para o grupo de validação, para fazer tal coisa é avaliado em cada epoch, qual o MSE do conjunto de validação escolhendo os pesos para o qual este é menor. O objetivo deste processo é verificar se os pesos escolhidos para a rede são válidos para um conjunto de validação que corresponde em padrões que a rede não tinha visto, de forma a entender-se qual o comportamente da rede com dados novos.
Através deste passo de validação é também possível prevenir que haja overfitting pois os pesos não serão escolhidos segundo MSE do conjunto de dados com que a rede treina mas sim com os dados com que esta é validada.

Train the network using the first 70 patterns for training, the next 15 for validation and the last 15 for testing. Click the "Performance" button. Comment on what is shown in the plot.

No plot é obervado como evolui a cada iteração o erro quadrado médio do conjunto de treino, de validação e de teste.
O MSE do conjunto de treino converge para o goal MSE definido, o validation error de cada época indica qual o pesos que devemos utilizar para a posterior utilização da rede que não corresponderão aos pesos da iteração onde se chega ao goal.

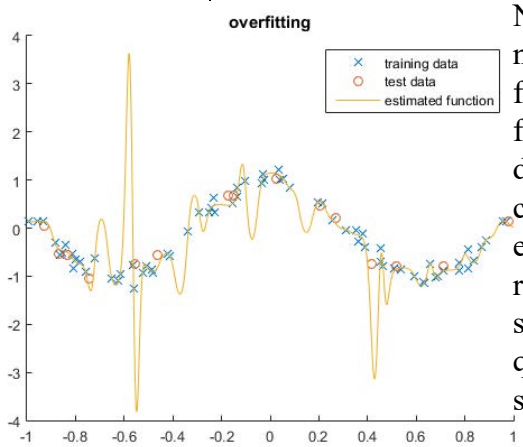
Plot the training data, the test data and the estimated function, all in the same figure and comment.



Como se pode observar no gráfico à esquerda a rede faz um bom fitting do conjunto de treino, e também ao conjunto de teste o que demonstra que os pesos escolhidps foram adequados. É possível observar que o efeito de overfitting não é notável pois a função não tenta passar por todos os pontos de treino e por isso não existem picos desnecessários ao longo da função.
Por fim compreende-se que apesar de o fitting ser bom a função continua a afastar-se da função original cos(5x) em algumas zonas.

2.4 Training without a validation set

Repeat the previous item but do not use a validation set this time. Observe the evolution of the cost function for the different sets and comment. Is there any sign of overfitting?



Não usando o validation set os pesos que a rede vais escolher no final serão os que minimizam o erro de treino o que vai forçar a função a passar pelos pontos do conjunto de treino ficando muito adaptadas a estes e gerando overfitting desviando-se da função que queremos efetivamente aproximar cos(5x). Assim a pôr outros inputs na rede como por exemplos as coordenada x do conjunto de teste, o output da rede afaste se bastante da coordenada y para esses pontos ou seja há um MSE maior para este conjunto. Assim concluí-se que é importante ter um validation set para impedir que situações destas ocorram.

Plot the estimated function on the same picture as 2.3. Compare the two estimated function and comment.

Como se pode observar na figura abaixo existem claras diferança entre as funções estimadas com e sem o uso de um conjunto de validação.
Usando conjunto de validação a função estimada fica próxima da original havendo apenas pequenos picos.
Não usando um conjunto de validação nota-se claramente no overfitting da função estimada que gera picos enorme ao longo desta que são consequência de esta estar a aproximar-se o máximo possível dos pontos do conjunto de treino o que leva a conclusão que aproximar demais aos pontos do conjunto de treino não significa de todos aproximar à função que se quer efetivamente fazer o fit.

without validation(**overfitting**) vs with validation