Artificial neural networks - Exercise session 1

Supervised learning and generalization

Cristina Luna

April 29, 2018

1 Introduction

MLP has been used as function approximation since 1993, when Barron demonstrated that using these networks, the approximation error becomes independently of the input space. This afirmation implies some advantages in processing time and complexity in comparison with polynomial expansions.

In this project, we will analyze the performance of the MLP for the mentioned problem using different learning algorithms and number of neurons per layer. In addition to this, a comparison between noisy data and noiseless data will be performed. Finally, another analysis was performed by changing the number of samples used for training in the case of noisy data. As a result of these experiments, some conclusions will be commented in terms of overfitting, speed and generalization capacity of each system.

In order to give a fair comparison between parameter settings, all the data presented in the document have been calculated by averaging the results of the systems after running them for 10 times. Another important remark is that all the networks started their training s with the default weights that the first one had in each test but this initialization changed in each of the 10 tests.

As final note, the initial number of neurons was 20, the standard deviation of the noise added to the data was 0.3 and the maximum number of epochs was fixed at 2.000 although in some cases the algorithms stops before because the early stopping implementation that Matlab has configured by default, altough we set the parameter: $trainParam.min\_grad = 1e-4$, in order to avoid overfitting.

The reason of the neurons choice is based on previous experiments performed with trainlm as training algorithm (because it gives good results in general). In these experiments, if the net had lower number of neurons, it wasn't able to follow the waveform of the sine for some ranges of time, even with 20 neurons some of the parts of the sine are not learned during the training, however I maintained this value because it could be interesting for the later comparisons.

The second parameter, noise deviation, was selected by comparing with others values and because this one adds some noise to the samples but it is still under the half of the maximum amplitude of the signal, so it's not going to distorted it completely.

2 Results and Conclusions

Some of the most important conclusions that we extract from the results is that SGD is the worst in terms of correlation values reached, one of the possible reasons is that this algorithms needs more epochs to converge in comparison with the others, although it invests less time than the others in each epochs because it just needs to calculate gradien information to update the weights, instead of the hessian or an approximation of the hessian as the Newton and quasi-Newton methods do.

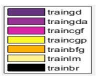
The algorithm that shows the best ratio correlation-time is the trainlm, because trainbfg and trainbr obtains really good results in correlation but the time that they take in training is higher than other algorithms. The reason of this is that the LM method uses the jacobian as approximation for the hessian, the BFGS also uses an approximation of the hessian but it is not as faster as the previous one so probably the operations takes longer than LM. Trainbr gets always good results but when we increase the number of neurons, the time that it takes is much higher than the others in most cases, even when we are training with not noisy data.The good performance in terms of R for higher number of neurons for the trainbr algorithm can be explained because it uses Bayesian regularization.

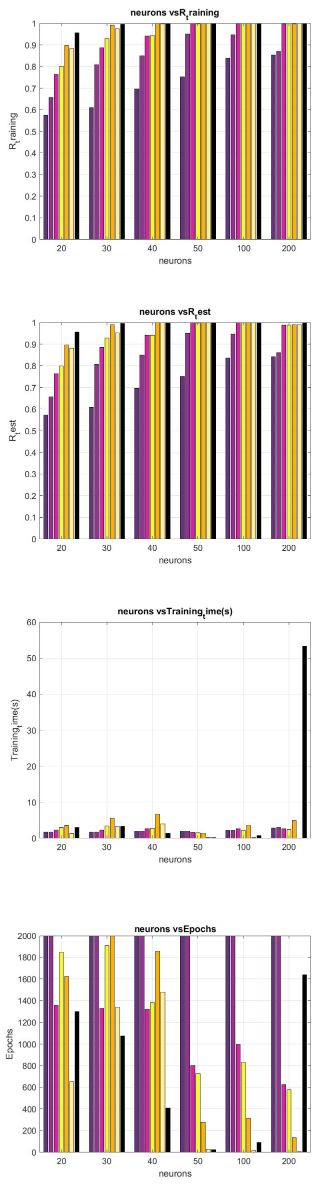
Regarding the overfitting and generalization capacity, we can see that for the case of small number of samples and noise, when we use 50 neurons or more, the R value in train and test starts to differ, these signals indicate that these nets are using too many neurons and thus there are too many free parameters and the systems follow each sample in the training but they are not able to generalize and for this reason in the test, they fail.

For the case of not having noise, there are no clear evidences of overfitting, although for 200 neurons case it seems that the test predictions are not as good as the training, so it could mean that for this case, it's starting to appear.

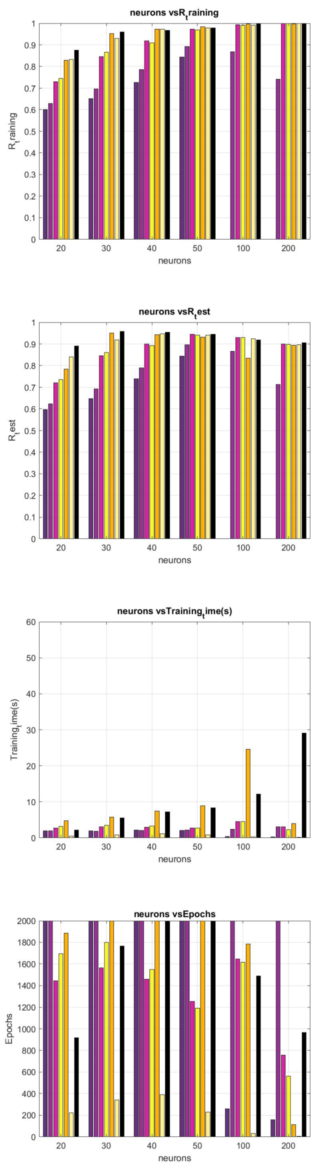
Finally, if we compare with the case of more samples, the R value is worst than for the case of using less number of samples although the test correlations are higher, which means that system are overfitting free and, therefore, increasing the number of samples help to perform a better training, although probably there is a limit in the amount of new samples, once we pass through this threshold, new data don't add new information and the training time gets higher.

In the following figure,we can see some of the graphs obtained from the results and used for extracting the previous mentioned conclusions The legend used in each of the plots also appears below:

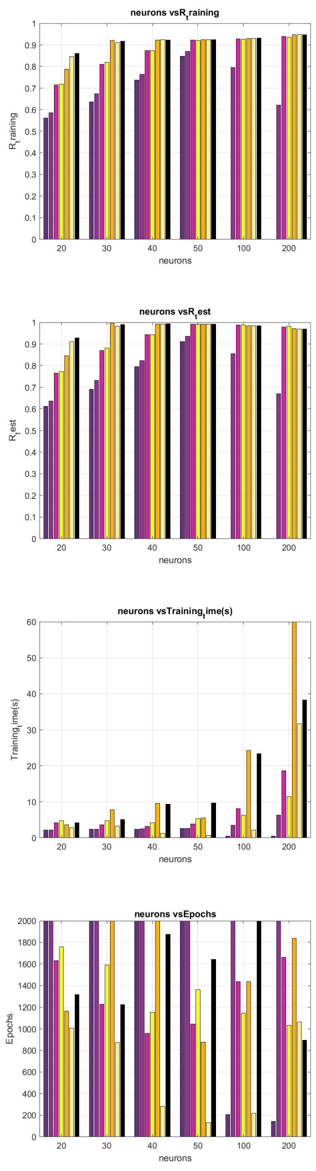




(a) Results without noise



(b) Results with noise and 189 samples in training



(c) Results with noise and 1.000 samples in training

Figure 1: R in training, R in testing, trianing time and number of epochs obtained for each experiment