Optimization Methods for Machine Learning: Homework #2

Due on Monday, November 21, 2016

Laura Palagi - Ing. Umberto Dellepiane

Alessandro Gallo

Contents

Instructions **3**

Question 1.1 **3**
- Which activation function do you choose . . . 3
- The final setting for the number of neurons N and for the parameter c . . . 3
- Which optimization routine do you use for solving the minimization problem and the setting of its parameters . . . 3
- The value of the error on the training and test set . . . 3
- The plot of the function representing the approximating function . . . 5

Question 1.2 **6**
Which RBF function do you choose . . . 6
- The values of the parameters ρ . . . 6
- The final setting for the number of neurons N and for the parameter σ . . . 6
- Which optimization routine do you use for solving the minimization problem and the setting of its parameters . . . 7
- The value of the error on the training and test set . . . 7
- The plot of the function representing the approximating function . . . 8
- A comparison of performance between MLP and RBF network . . . 9

Question 2.1 **9**

Question 2.2 **9**

Instructions

In this report there are just the answers and plots requested by the homework. In the folder HW2 there are the complete codes .m with the comments to understand them (comments are mostly in the MLP related files because RBF files are very similar). The answers for the Question 1.1 are obtained running q11.m script that calls [MLP.m, MLP_error.m, MLP_test.m, MLP_plot]. For the Question 1.2 instead answers are obtained running q12.m script that calls [RBF.m, RBF_error.m, RBF_test.m, RBF_plot]. Unfortunately i couldn't implement the gradient for the other two exercises.

Question 1.1

- Which activation function do you choose

The activation function used in the MLP algorithm is the hyperbolic tangent.

Ex: the hyperbolic tangent implemented in the MLP_error.m file

```
f2(j)=(1-exp(-c*f1(j)))/(1+exp(-c*f1(j)));
```

- The final setting for the number of neurons N and for the parameter c

The setting chosen as example is N=6 neurons and the parameter for the hyperbolic tangent c=5.

- Which optimization routine do you use for solving the minimization problem and the setting of its parameters

The MLP.m file implements a matlab routine of the optimization toolbox called 'fminunc'. The function minimizes the error in the MLP_error.m file and uses an anonymous function to catch the optimal weights

```
fun=@(w_all)MLP_error(w_all,train,c,N);
[opt_ws,fval] = fminunc(fun,w_all,options);
```

The parameters are set in 'options' and let with the default parameters. The algorithm used is the 'quasi-newton'.

```
options = optimoptions(@fminunc,'Algorithm','quasi-newton','Display','none');
```

- The value of the error on the training and test set

In the bar plots below we can check the error valuation in the training and test sets for increasing number of neurons, each one for three different c values. We consider the case from 1 neuron to 15 for each value of c=[0.5 1 5]

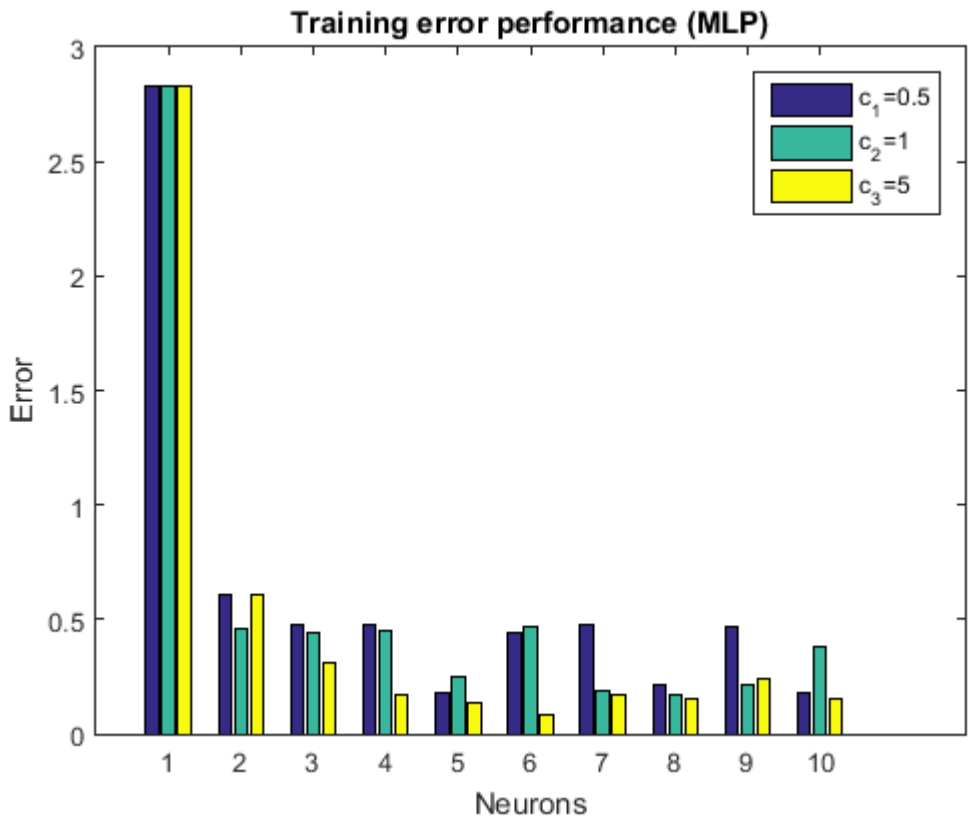

Figure 1: Training error perfomance (MLP)

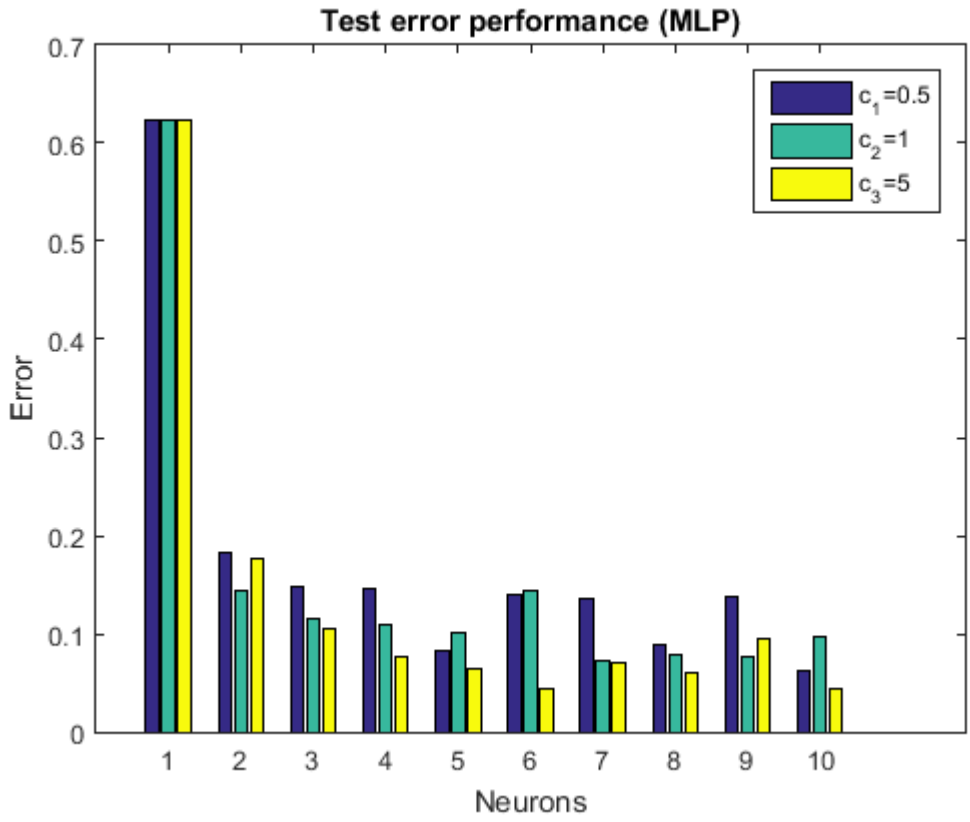

Figure 2: Test error perfomance (MLP)

- The plot of the function representing the approximating function

In the plot below we can see the approximating function with parameters N=6 and c=5 on the real franke function.

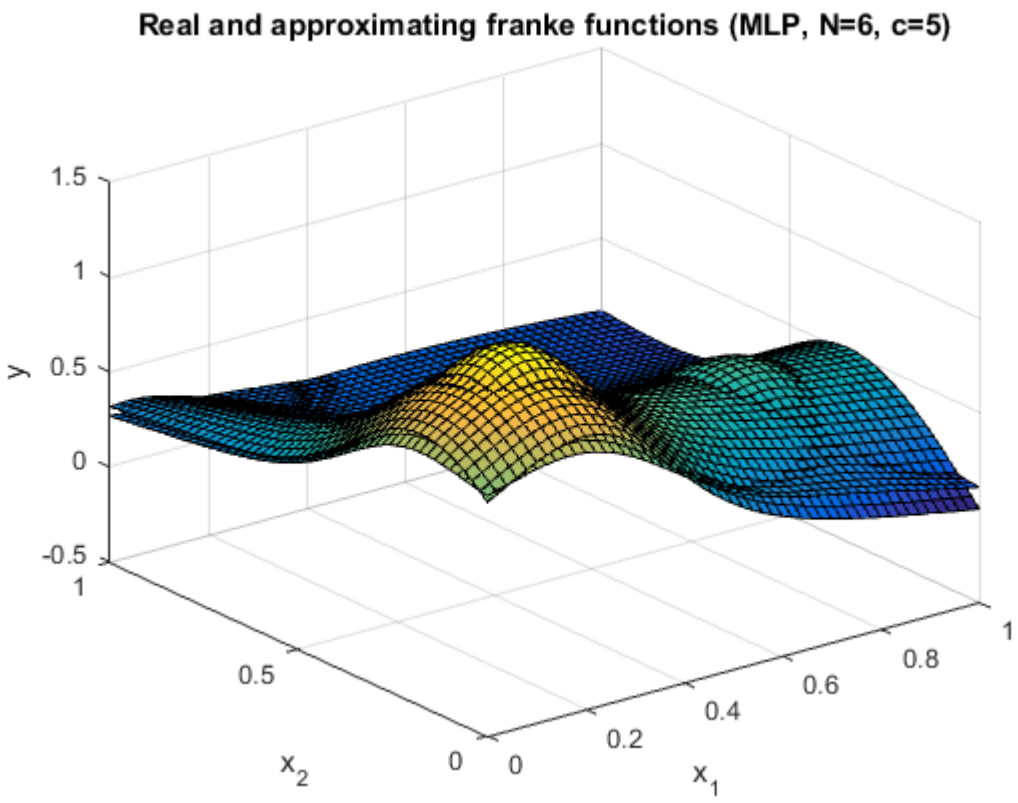

Figure 3: Real and approximanting franke functions (N=6, c=5)

Question 1.2

Which RBF function do you choose

The RBF function chosen is the Inverse Multiquadric

Ex: the Inverse Multiquadric implemented in the RBF_error.m file

```
f2(j)=(f1(j)^2+sigma^2)^(-1/2);
```

- The values of the parameters ρ

The parameters chosen for ρ_1 and ρ_2 are:

```
rho1=10^(-3.5);
rho2=10^(-4.5);
```

- The final setting for the number of neurons N and for the parameter σ

The setting chosen as example is N=6 neurons and the parameter for the Invese Multiquadric $\sigma = 0.3$.

- Which optimization routine do you use for solving the minimization problem and the setting of its parameters

The RBF.m file implements a matlab routine of the optimization toolbox called 'fminunc'. The function minimizes the error in the RBF_error.m file and uses an anonymous function to catch the optimal weights.

```
fun=@(w_all)RBF_error(w_all,train,sigma,rho1,rho2,N);
[opt_ws,fval,~,~] = fminunc(fun,x0,options);
```

The parameters are set in 'options' and let with the default parameters. The algorithm used is the 'quasi-newton'.

```
options = optimoptions(@fminunc,'Algorithm','quasi-newton','Display','none');
```

- The value of the error on the training and test set

In the bar plots below we can check the error valuation in the training and test sets for increasing number of neurons, each one for three different σ values. We consider the case from 1 neuron to 10 for each value of σ=[0.3 0.5 1]

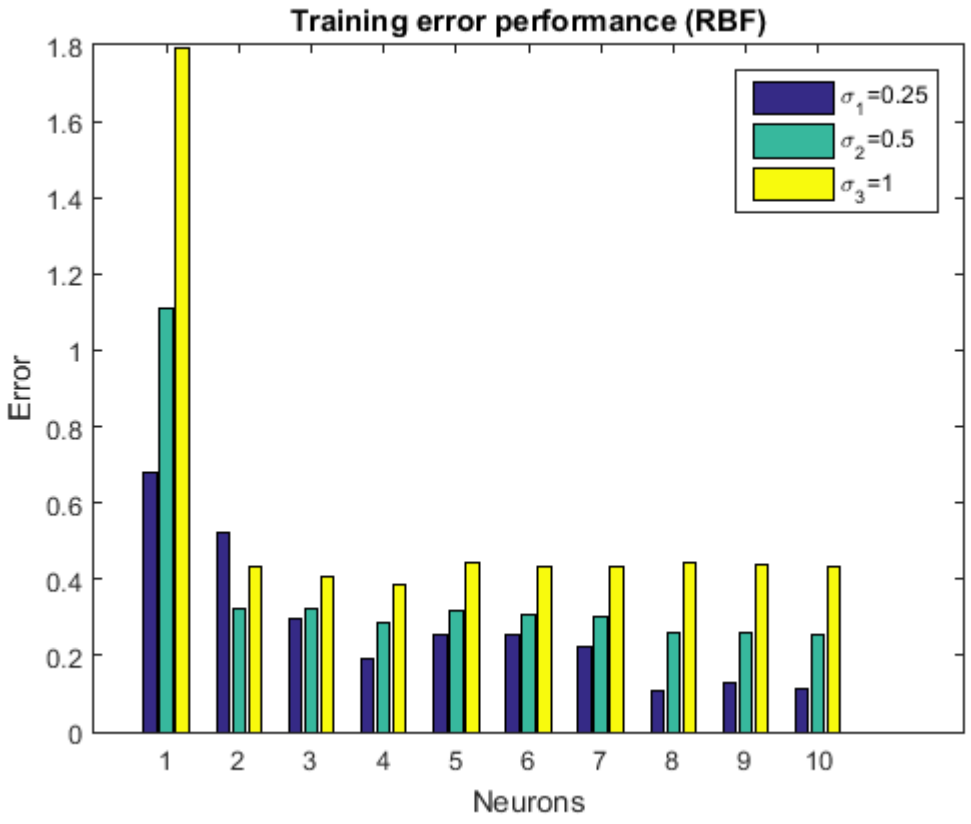

Figure 4: Training error perfomance (RBF)

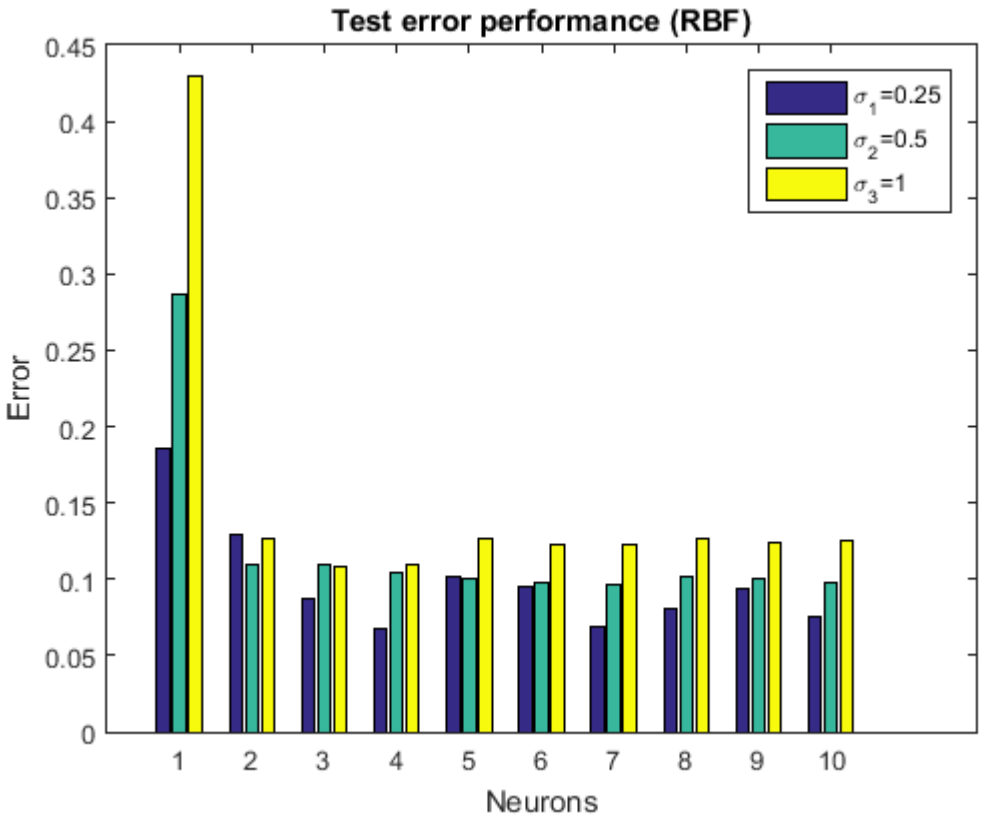

Figure 5: Test error perfomance (RBF)

- The plot of the function representing the approximating function

In the plot below we can see the approximating function with parameters N=8 and c=0.3 on the real franke function.

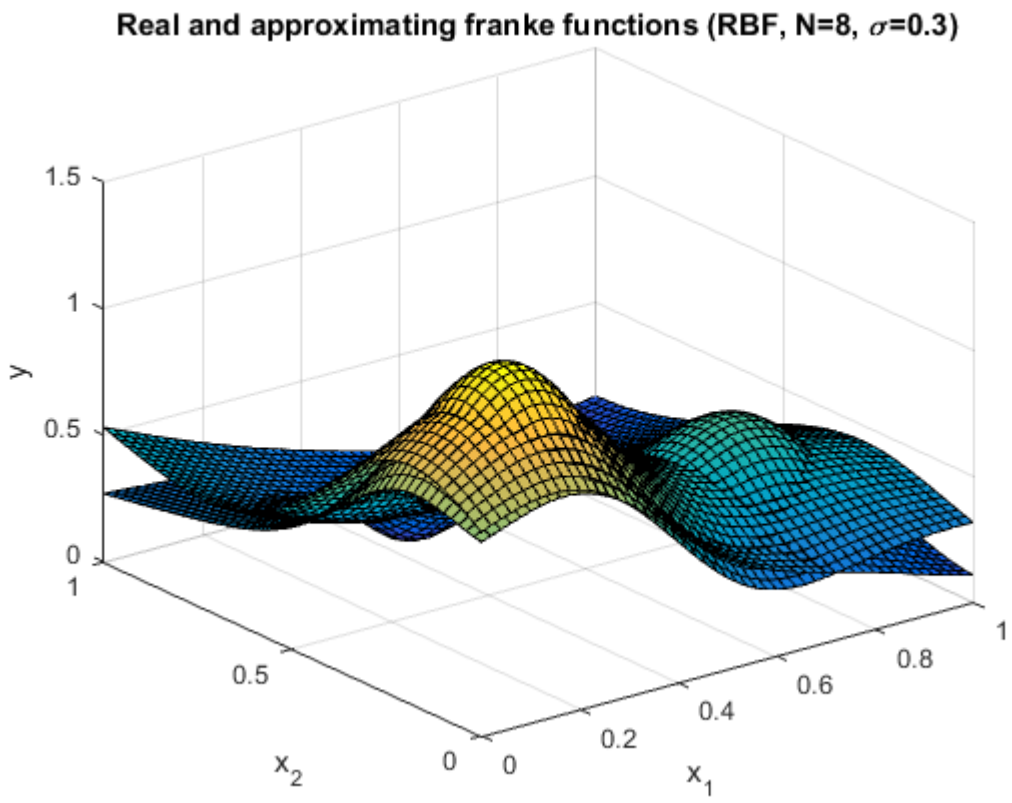

Figure 6: Real and approximating franke functions (RBF, N=8, c=0.3)

- A comparison of performance between MLP and RBF network

In the performance_table.mat file there is a comparison between the MLP and RBF algorithms in terms of the mean of the errors (training and test sets) and the time to get the solution. The RBF algorithm allows to get a better result but needs additional time compared to the MLP algorithm.

Question 2.1

Question 2.2