Universidade Fedeal de Minas Gerais

Exercício

A.P. Braga

October 3, 2018

Backpropagation

Perceptrons de múltiplas camadas (MLP) são capazes de aproximar qualquer função quando possuem duas camadas escondidas, quando há apenas uma camada intermediária é possível a aproximação qualquer função contínua. O algoritmos mais difundido para o treinamento de uma rede MLP é chamado *backpropagation*, ele utilizada o conceito da retro propagação do erro da rede para corrigir os pesos da última camada até a primeira. O ajuste dos pesos na camada de saída é dado por

$$\Delta w_{ji} = \alpha e_j(n) f'(u_j(n)) h_i(n)$$

em que i é o índice dos neurônios da camada escondida, j é o índice dos neurônios da camada de saída, $e_j(n)$ é o erro apresentado na saída para a entrada n, $h_i(n)$ é a saída do i-ésimo neurônio da camada escondida, $u(n)$ é o somatório das saídas da camda escondida ponderada pelos pesos w_{jk} e f' é a derivada da função de ativação da camada de saída. A constante α é uma taxa de aprendizado arbitrária que, em geral, deve apresentar um valor baixo.
Para a camada escondida deve ser considerada a retro propagação do erro, assim o ajuste dos pesos da camada escondida é dado por:

$$\Delta w_{ik} = \eta h_i'(u_i(n)) \Sigma_j e_j(n) f'(u_j(n)) w_{ji}(n) x_k(n)$$

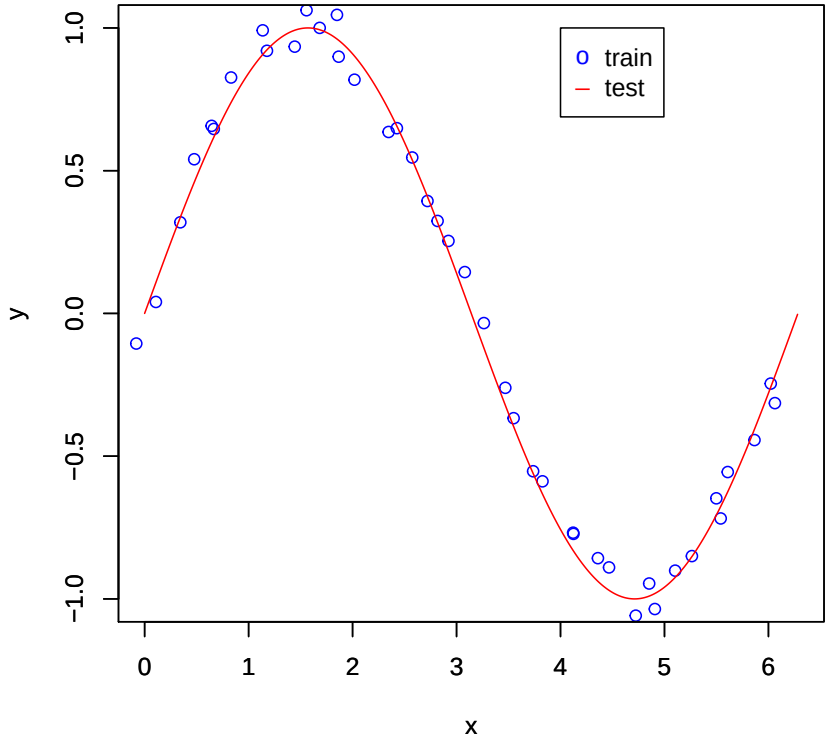
em que k é o índice da entrada x, h' é a derivada da função de ativação da camada escondida, no caso da função linear a derivada da função é uma constante. E η é a taxa de aprendizado.

Instruções

Para observar que o MLP é capaz de aproximar qualquer função contínua, deve realizar a regressão de um ciclo de uma senoide MLP com backpropagation. A função de ativação da camada de saída deve ser linear, e devem haver 3 neurônios na camada escondida. Adaptar código desenvolvido em sala de aula, utilizando saída linear.

1. Criar os dados de trainamento e de teste, de acordo com uma senoidal acrescida de um erro:

```
> x_train<-seq(from=0, to=2*pi, by =0.15)
> x_train<-x_train + (runif(length(x_train))-0.5)/5
> i <- sample(length(x_train))
> x_train <- x_train[i]
> y_train <- sin(x_train)
> y_train<-y_train + (runif(length(y_train))-0.5)/5
> plot(x_train,y_train,col='blue',xlim = c(0,2*pi),
+      ylim = c(-1,1),xlab = 'x',ylab = 'y')
> x_test <-seq(from=0, to=2*pi, by =0.01)
> y_test <-sin(x_test)
> par(new=T)
> plot(x_test,y_test,col='red',type='l',xlim = c(0,2*pi),
+      ylim = c(-1,1),xlab = 'x',ylab = 'y')
> legend(x=4, y=1, legend = c('train','test'),
+        col = c('blue','red'),pch=c('o','_'))
```



2. Definir uma RNA com uma camada escondida de 3 neurônios e um neurônio de saída com função de ativação linear. De forma semelhante a feita em sala para o problema da XOR.

3. Implementar um função de treinamento utilizando o backpropagation e ajustar um modelo com os dados de treinamento.

4. Testar o modelo obtido com os dados de teste.

5. Calcular o erro quadrático médio (MSE) percentual do classificador.

6. Repetir 5 vezes os procedimentos 2 ao 5 e estimar o MSE percentual médio e o desvio padrão do classificador.

7. Plotar as saídas da função aproximada e os valores esperados y de forma ilustrativa para uma única execução.