RNA - Exercicios 03 e 04

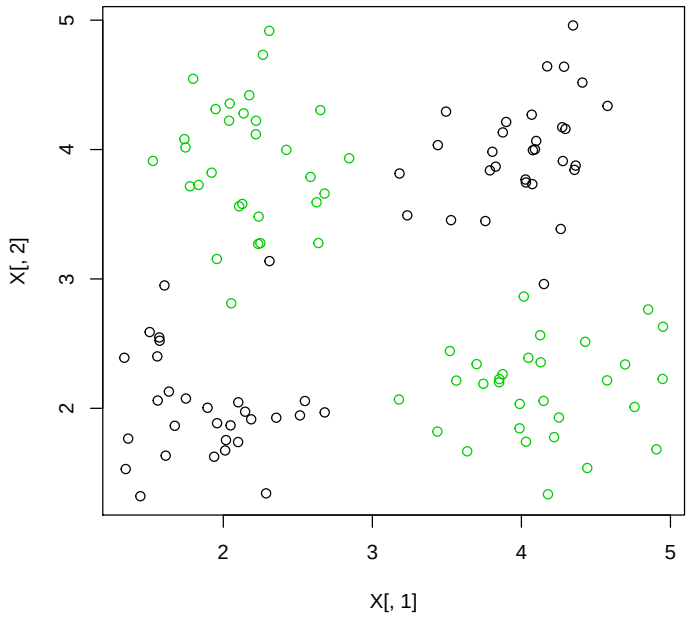
Ramon Durães, 2019720188

September 9, 2019

1 Exercício 03

1.1 a) ELM para os dados XOR

Para este exercício, o primeiro passo é carregar a base de dados "data2classXOR.txt" fornecida. Abaixo está uma visualização destes dados:



Percebe-se que os dados não são linearmente separáveis.

O teorema de Cover diz que, com alta probabilidade, é possível transformar dados não-linearmente separáveis em dados linearmente separáveis se projetados em um espaço dimensional maior. As Extreme Learning Machines (ELMs) fazem exatamente isso ao introduzir uma camada escondida com mais unidades que entradas. Os pesos Z de projeção dos dados de entrada X nessa camada

são gerados aleatoriamente. A função de ativação desta camada é a tangente hiperbólica, de forma que calcula-se a saída da camada escondida:

$$H = tanh(X * Z) \tag{1}$$

Abaixo está a função de treinamento das ELMs utilizando o método analíticio: os pesos W que mapeiam os dados da camada escondida para a saída são ajustados pela resolução de um sistema de equações:

$$W = H' * Y \tag{2}$$

Onde Y é a classe de cada observação.

```
> # ELM Training function
> train_ELM <- function(X, Y, n_hidden, augment=TRUE){
+
+   if(augment){
+     X <- cbind(replicate(dim(X)[1],1),X)
+   }
+
+   # Calculating H and Hinv
+   Z <- replicate(n_hidden, runif(dim(X)[2],-0.5, 0.5))
+   H <- tanh(X %*% Z)
+   Hinv <- solve(t(H) %*% H) %*% t(H)
+   #Hinv <- pseudoinverse(H)
+   # Calculating W
+   W <- Hinv %*% Y
+   return(list("Z"=Z,"W"=W))
+ }
```

Para calcular a saída, basta avaliar as novas entradas, calculando sua matriz H equivalente, a multiplicando pelos pesos W. Então, extrai-se o sinal do resultado para transformar a função linear em uma função degrau, como mostrado abaixo. Nota-se que as classes reais são passadas como parêmtro para o cálculo da matriz de confusão.

```
> test_ELM <- function(ELM, Xnew, Ytrue, augment=TRUE){
+   if(augment){
+     Xnew <- cbind(replicate(dim(Xnew)[1],1),Xnew)
+   }
+   Yhat <- sign(tanh(Xnew %*% ELM$Z) %*% ELM$W)
+   confMat <- confusionMatrix(as.factor(Yhat), as.factor(Ytrue))
+   return(list("Yhat" = Yhat, "confMat" = confMat))
+ }
```

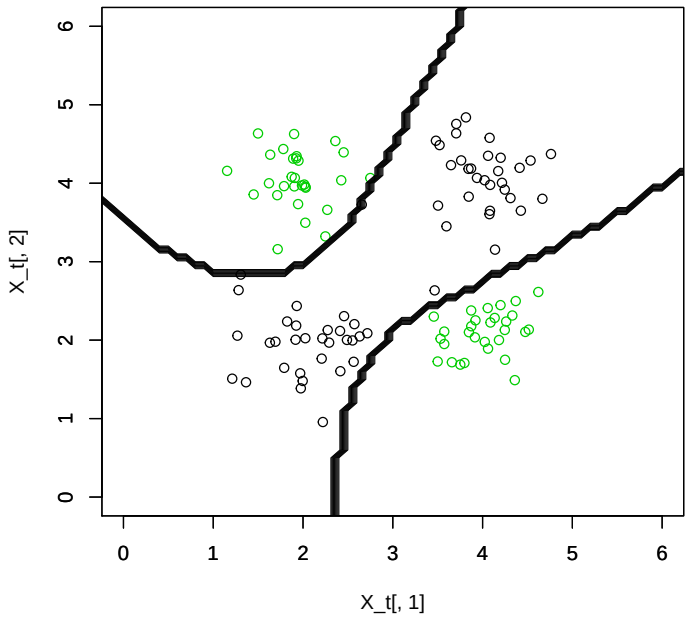
Aplicando as funções nos dados XOR e exibindo as métricas de classificação:

```
> n_hidden <- 6
> ELM <- train_ELM(X, Y, n_hidden)
> results <- test_ELM(ELM, X_t, Y_t)
> # Analyzing classification results
> print(results$confMat)

Confusion Matrix and Statistics

          Reference
Prediction -1  1
        -1 60  2
        1   0 58

               Accuracy : 0.9833
                 95% CI : (0.9411, 0.998)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9667

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 1.0000
            Specificity : 0.9667
         Pos Pred Value : 0.9677
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5167
      Balanced Accuracy : 0.9833

       'Positive' Class : -1
```

Uma ELM com 6 unidades na camada escondida costuma ser suficiente para separar perfeitamente ou quase perfeitamente os dados, obtendo acurácia superior a 95%. Abaixo é mostrado o contorno de classificação obtido:

```
> # Plotting results and decision function (contour)
> xseq <- seq(-2, 10, 0.1)
> lseq <- length(xseq)
> MZ <- matrix(nrow=lseq, ncol=lseq)
> for (i in 1:lseq){
+   for(j in 1:lseq){
+     x1x2 <- as.matrix((cbind(1,xseq[i], xseq[j])))
+     MZ[i, j] <- sign(tanh(x1x2 %*% ELM$Z)%*%ELM$W)
+   }
+ }
```

```
> plot(X_t[,1], X_t[,2], col=results$Yhat+2, xlim=c(0,6), ylim=c(0,6))
> par(new=T)
> contour(x= xseq, y=xseq, z=MZ, xlim=c(0,6), ylim=c(0,6), lwd=0.5, drawlabels = F)
```



Mais unidades na camada escondida fazem com que a classificação seja cada vez mais sobreajustada nos dados de treinamento, o que é um grande problema para estes modelos pois eles falham ao generalizar para novos dados.

1.2 b) ELM para a base BreastCancer

A base de dados BreastCancer, do pacote mlbench, foi carregada. Valores faltantes (NA) foram removidos, a classe "malignant" foi codificada com 1 e "benign" com -1. Os dados foram embaralhados aleatoriamente e divididos entre 70% treinamento e 30% teste:

```
> # Loading and preprocessing data
> library(mlbench)
> data(BreastCancer)
> bc <- BreastCancer
> # Removing NA
> bc <- bc[complete.cases(bc), ]
> # Removing Id and Class
> bc <- bc[,!names(bc) %in% c("Id", "Class")]
```

```
> # Defining benign as 0 as malign as 1
> class <- 2 * (BreastCancer$Class == "malignant") -1
> # Converting to numeric matrix
> X <- data.matrix(bc)
> Y <- data.matrix(class)
> # Train test splits with random sort
> seq <- sample(1:dim(X)[1])
> train_perc <- 0.7
> ntrain <- round(train_perc * dim(X)[1])
> Xtrain <- X[seq[1:ntrain],]
> Ytrain <- Y[seq[1:ntrain],]
> Xtest <- X[tail(seq, -ntrain),]
> Ytest <- Y[tail(seq, -ntrain),]
```

Aplicando as funções de treinamento e teste de ELMs para os dados de BreasCancer, e visualizando as métricas de classificação:

```
> # Training and testing ELM
> n_hidden = 40
> ELMbc <- train_ELM(Xtrain, Ytrain, n_hidden)
> resultsbc <- test_ELM(ELMbc, Xtest, Ytest)
> resultsbc$confMat

Confusion Matrix and Statistics

          Reference
Prediction  -1   1
        -1 112  67
        1   11  15

               Accuracy : 0.6195
                 95% CI : (0.5493, 0.6862)
    No Information Rate : 0.6
    P-Value [Acc > NIR] : 0.3102

                  Kappa : 0.1055

 Mcnemar's Test P-Value : 4.739e-10

            Sensitivity : 0.9106
            Specificity : 0.1829
         Pos Pred Value : 0.6257
         Neg Pred Value : 0.5769
             Prevalence : 0.6000
         Detection Rate : 0.5463
   Detection Prevalence : 0.8732
      Balanced Accuracy : 0.5467
```

```
'Positive' Class : -1
```

Percebe-se que a acurácia não é excelente para este problema, ficando por volta de 65%. O ideal então seria fazer uma seleção de modelos, calculando a acurácia para cada um de acordo com o número de neurônios da camada escondida. Assim evita-se o overfit mantendo a capacidade de generalização do classificador.

2 Exercício 04

2.1 a) RBF para os dados XOR

Mais uma vez utilizaremos os dados XOR exibidos no primeiro exercício.

As redes RBF (Radial Basis Function) têm como motivação a proximidade espacial entre observações de uma mesma classe. Dessa forma, cada unidade da camada esconida representa uma função de base radial. No nosso caso simplificado, as RBFs utilizadas são Gaussianas bivariadas simples (sem correlação).

Para definir os centros μ dessas Gaussianas, utilizou-se o método K-médias. O desvio padrão σ foi mantido em 1, de forma que a matriz H é obtida por:

$$H_{i,j} = e^{-(1/\sigma_j^2)*t(x_i - mu_j)*(x_i - mu_j)} \tag{3}$$

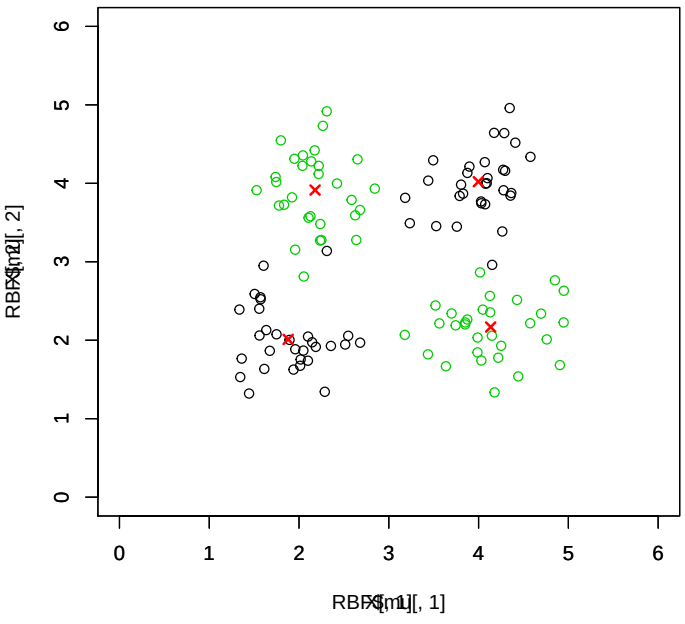
Mais uma vez, para treinar a rede basta calcular os pesos de forma analítica como mostrado na equação (2). Obtém-se então a função de treinamento das RBFs:

```
> # RBF Training function
> train_RBF <- function(X, Y, n_hidden, augment=FALSE){
+
+   n <- dim(X)[1]
+   if(augment){
+     X <- cbind(replicate(dim(X)[1],1),X)
+   }
+
+   # Finding clusters using Kmeans
+   clusters <- kmeans(X, centers = n_hidden)
+   mu <- clusters$centers
+   s <- matrix(1, nrow=n_hidden, ncol=1) # Using unit radius
+
+   # Calculating H
+   H <- matrix(nrow=n ,ncol=n_hidden)
+   for(i in 1:n){
+     xi <- X[i,]
+     for(j in 1:n_hidden){
+       H[i,j] <- exp(- (1/(s[j]^2)) * dist(rbind(xi,mu[j,])))
```

```
+     }
+   }
+
+   # Calculating Hinv
+   Hinv <- solve(t(H) %*% H) %*% t(H)
+
+   # Calculating W
+   W <- Hinv %*% Y
+   return(list("mu"= mu, "s" = s, "W" = W))
+ }
```

Como conhecemos a estrutura dos dados XOR, utilizaremos 4 clusters (ou 4 unidades na camada escondida). Aplicando nos dados e plotandos os centros da função K-médias para validar as escolhas de centros:

```
> n_hidden <- 4
> set.seed(5)
> RBF <- train_RBF(X, Y, n_hidden)
> # Plotting clusters
> plot(X[,1], X[,2], col=Y+2, xlim=c(0,6), ylim=c(0,6))
> par(new=T)
> plot(RBF$mu[,1], RBF$mu[,2], xlim=c(0,6), ylim=c(0,6), pch=4, col="red", lwd=2)
```



A saída $\hat{Y}$ da rede RBF é calculada da mesma forma que a saída da rede ELM acima: obtém-se o sinal da multiplicação $H * W$. Abaixo está a função que avalia uma RBF.

```
> # Evaluating RBF
> test_RBF <- function(RBF, Xnew, Ytrue, augment=FALSE){
+   if(augment){
+     Xnew <- cbind(replicate(dim(Xnew)[1],1),Xnew)
+   }
+
+
+   # Calculating H
+   n <- dim(Xnew)[1]
+   n_hidden <- dim(RBF$mu)[1]
+   H <- matrix(nrow=n ,ncol=dim(RBF$mu)[1])
+   for(i in 1:n){
+     xi <- Xnew[i,]
+     for(j in 1:n_hidden){
+       H[i,j] <- exp(- (1/(RBF$s[j]^2)) * dist(rbind(xi,RBF$mu[j,])))
+     }
+   }
+   Yhat <- sign(H %*% RBF$W)
+   confMat <- confusionMatrix(as.factor(Yhat), as.factor(Ytrue))
+   return(list("Yhat" = Yhat, "confMat" = confMat))
+ }
```

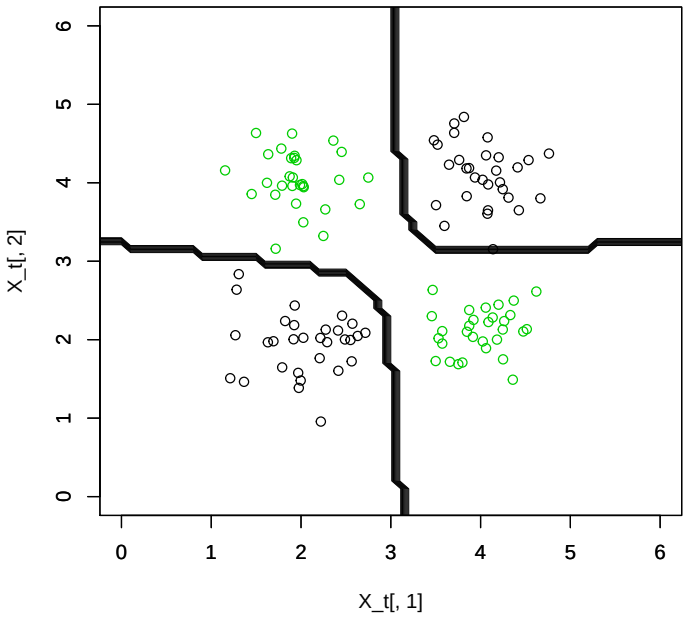
Aplicando nos dados e obtendo as métricas de classificação.

```
> results <- test_RBF(RBF, X_t, Y_t)
> # Analyzing classification results
> print(results$confMat)

Confusion Matrix and Statistics

          Reference
Prediction -1  1
        -1 60  0
        1   0 60

               Accuracy : 1
                 95% CI : (0.9697, 1)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA
```

```
           Sensitivity : 1.0
           Specificity : 1.0
        Pos Pred Value : 1.0
        Neg Pred Value : 1.0
            Prevalence : 0.5
        Detection Rate : 0.5
  Detection Prevalence : 0.5
     Balanced Accuracy : 1.0

      'Positive' Class : -1
```

Mais uma vez, considerando que os 4 clusters tenham sido bem definidos pelo K-médias, 4 unidades na camada escondida costuma ser suficiente para obter acurácia 1 ou próxima disso. Plotando o contorno de classificação:

```
> # Plotting results and decision function (contour)
> xseq <- seq(-2, 10, 0.1)
> lseq <- length(xseq)
> MZ <- matrix(nrow=lseq, ncol=lseq)
> for (i in 1:lseq){
+   for(j in 1:lseq){
+     x1x2 <- as.matrix((cbind(xseq[i], xseq[j])))
+     hi <- matrix(nrow=1, ncol=n_hidden)
+     for(c in 1:n_hidden){
+       hi[1,c] <-  exp(- (1/(RBF$s[c]^2)) * dist(rbind(x1x2,RBF$mu[c,])))
+     }
+     MZ[i, j] <- sign(hi%*%RBF$W)
+   }
+ }
> plot(X_t[,1], X_t[,2], col=results$Yhat+2, xlim=c(0,6), ylim=c(0,6))
> par(new=T)
> contour(x= xseq, y=xseq, z=MZ, xlim=c(0,6), ylim=c(0,6), lwd=0.5, drawlabels = F)
```



2.2 b) RBF para a base BreastCancer

Mais uma vez os dados da base BreastCancer foram carregados, tratados e divididos entre treinaento e teste (como mostrado no código do exercicio 3-b).

Aplicando as funções de treinamento e teste de RBFs aos dados, considerando 40 clusters, obtemos as métricas abaixo:

```
> # Training and testing RBF
> n_hidden =30
> RBFbc <- train_RBF(Xtrain, Ytrain, n_hidden)
> resultsbc <- test_RBF(RBFbc, Xtest, Ytest)
> resultsbc$confMat

Confusion Matrix and Statistics

          Reference
Prediction  -1   1
        -1 123  65
        1   10   7

               Accuracy : 0.6341
                 95% CI : (0.5642, 0.7001)
```

```
    No Information Rate : 0.6488
    P-Value [Acc > NIR] : 0.6975

                  Kappa : 0.0267

 Mcnemar's Test P-Value : 4.507e-10

            Sensitivity : 0.92481
            Specificity : 0.09722
         Pos Pred Value : 0.65426
         Neg Pred Value : 0.41176
             Prevalence : 0.64878
         Detection Rate : 0.60000
   Detection Prevalence : 0.91707
      Balanced Accuracy : 0.51102

       'Positive' Class : -1
```

Mais uma vez a acurácia obtida fica em torno de 60%, o que mostra que o classificador falha ao generalizar e é necessária uma seleção de modelos para aprimorar os resultados. Outra possibilidade é aumentar um pouco a complexidade das funções de base, alterando os desvios padrão e permitindo correlação, por exemplo.