Conhecimento Não Simbólico: Redes Neuronais Artificiais

SRCR 2º SEMESTRE 2016/2017

Mestrado integrado em Engenharia Informática

Universidade do Minho, Campus de Gualtar

Bruno Miguel Salgado Machado	74941
Carlos Manuel Magalhães da Silva	75107
Gonçalo Leal da Mota Meireles Moreira	73591
João Rui de Sousa Miguel	74237
Paulo Jorge Machado Guedes	74411

Resumo

O presente relatório contém a documentação de todo o trabalho realizado para a disciplina de Sistemas de Representação de Conhecimento e Raciocínio.

O objetivo deste trabalho foca-se em determinar os níveis de exaustão dos trabalhadores de uma dada empresa, bem como o tipo de tarefa que estes possam estar a realizar. Para tal, divide-se as tarefas em três possíveis, "*programming*", "*office*" e "*Work*", divide-se ainda a exaustão sob três critérios possíveis, 7 níveis de exaustão, a presença ou ausência desta e uma escala identificadora para a fatiga.

Este trabalho incorre do desenvolvimento de uma RNA (rede neuronal artificial) que recebe vários dados de entrada (métricas relativas á interação entre o utilizador e o rato / teclado) para que consiga determinar com o menor erro possível os níveis de exaustão e tarefas a estes associados.

Será importante, ainda, referir que todo o desenvolvimento deste programa foi efetuado através da linguagem de programação *R*, recorrendo ao *IDE RStudio* e a todas as bibliotecas que se consideraram uteis para o efeito.

Índice

Índice de Imagens

Introdução

Neste projeto foi-nos solicitado o desenvolvimento de redes neuronais artificiais para ajudar a identificar e dar resposta à exaustão presente no local de trabalho. Para tal recolheram-se dados numa escala com sete níveis distintos. Pretende-se ainda estipular qual a tarefa que um utilizador esta a efetuar, sendo que, para tal, desenvolveu-se uma rede neuronal com base em métricas, como por exemplo, a interação humano-computador através do teclado e através do rato.

O objetivo do trabalho é, então, treinar redes neuronais de forma a que sejam capazes de responder a *queries* com um grau de correção elevado, para que se possa inferir relativamente ao nível de exaustão dos funcionários e o tipo de tarefas que estes mesmos realizam naquele momento, utilizando o mesmo tipo de informação que a utilizada para efeitos de treino.

Descrição do Trabalho

Determinar Tarefa a Executar

Para treinar uma rede neuronal que determinasse qual tarefa a executar, foram efetuados alguns preliminares que se acharam necessários para que houvesse uma melhor constituição e treino da rede:

- Normalizaram-se as tarefas existentes, para valores entre 0 e 1 em vez de se utilizar uma *String*, visto que estas dificultam a tarefa de treino de uma RNA.

  Inicialmente procedeu-se á passagem para um valor inteiro, no entanto, verificou-se que este originava um erro maior, sendo que para que tal não acontecesse, normalizou-se posteriormente, para valores entre 0 e 1. (Ver Figura 1)

```
normaliza_tarefa <- function(e) {
  e$Tarefa[e$Tarefa == "work"] <- 1
  e$Tarefa[e$Tarefa == "programming"] <- 2
  e$Tarefa[e$Tarefa == "office"] <- 3
  e$Tarefa <- as.numeric(e$Tarefa)
  e$Tarefa[e$Tarefa == 1] <- round(1/3, digits = 2)
  e$Tarefa[e$Tarefa == 2] <- round(2/3, digits = 2)
  e$Tarefa[e$Tarefa == 3] <- round(3/3, digits = 2)
  return(e)
}
```

Figura 1 - Normalização da Tarefa

- Normalizaram-se também a exaustão para valores entre 0 e 1, ao contrário do que originalmente estava de 1 a 7, para que se consiga obter um erro menor, tal como acontecia na tarefa. (Ver Figura 2)

```
normaliza_exaustao <- function(e) {
  e$Exaustao[e$Exaustao == 1] <- round(1/7, digits = 2)
  e$Exaustao[e$Exaustao == 2] <- round(2/7, digits = 2)
  e$Exaustao[e$Exaustao == 3] <- round(3/7, digits = 2)
  e$Exaustao[e$Exaustao == 4] <- round(4/7, digits = 2)
  e$Exaustao[e$Exaustao == 5] <- round(5/7, digits = 2)
  e$Exaustao[e$Exaustao == 6] <- round(6/7, digits = 2)
  e$Exaustao[e$Exaustao == 7] <- round(7/7, digits = 2)
  return(e)
}
```

Figura 2 - Normalização da Exaustão

- Decidiu-se trocar a ordem dos dados, pois notou-se que alguns casos de treino relativos às tarefas realizadas estavam aglomerados, podendo levar a inconsistências no treino e teste da rede.

  Após recorrer ao *"shuffle"* dos elementos, notou-se que a dispersão dos dados aumentou, estando, deste modo, dispersos mais equitativamente tanto nos casos de treino como de teste. (Ver Figura 3)

```
shuffle <- function(e) {
  e <- e[sample.int(nrow(e)), ]
  return(e)
}
```

Figura 3 - Baralhar os casos

- Para que se consiga determinar a significância dos atributos para, utilizou-se a função desenvolvida *test_optimal*, que aplica a função *regsubsets* fornecida pela biblioteca *leaps*. Esta diz quais os melhores parâmetros a considerar na determinação do *output* desejado, neste caso a tarefa.
- Observou-se que quando se verifica os melhores parâmetros para treino, estes variam consoante o número de casos considerados para treino, deste modo adicionou-se a capacidade de testar os melhores parâmetros relativos a todos os dados presentes no ficheiro de input, bem como apenas os dados considerados como grupo de treino. (Ver Figura 4)

```
test_optimal <- function(e = "geral") {
  if (e == "treino")
    regg1 <- regsubsets(Tarefa ~ KDT + MA + MV + TBC + DDC + DMS + AED + ADMSL,
                        package$treino, nvmax = 10)
  else
    regg1 <- regsubsets(Tarefa ~ KDT + MA + MV + TBC + DDC + DMS + AED + ADMSL,
                        package$exaustao, nvmax = 10)
  print(summary(regg1))
}
```

Figura 4 - Determinar significância

- Finalmente, nos preparativos para o treino da rede, foi desenvolvida uma função que abria o ficheiro desejado, aplicava as funções de normalização tanto da tarefa como da exaustão e também a de baralhar.

  Além disto, inicializa a variável *"pckt"*, que irá ser utilizada ao longo do programa, esta mantem o registo de todos os dados lidos, todos os casos a utilizar para treino da RNA e o grupo que compõe os casos de teste. (Ver Figura 5)

```
init_pacote <- function(tt, tst, path) {
  e <- abrir_xls(path)
  e <- normaliza_tarefa(e)
  e <- normaliza_exaustao(e)

  e <- shuffle(e)
  e <- shuffle(e)

  pckt = NULL
  pckt$exaustao <- e
  pckt$treino   <- e[tt, ]
  pckt$teste    <- e[tst, ]

  return(pckt)
}
```

Figura 5 - Iniciar os Casos para treino/teste

Com tudo isto preparado falta também definir o treino, a apresentação de como o treino foi efetuado e cálculo do erro.

Assim foi definida uma função para treino (Ver Figura 6), e outra para que sejam apresentados um gráfico com representação das respostas obtidas a partir das *queries* de teste efetuadas bem como o valor da RMSE (Ver Figura 7). Este gráfico terá a comparação entre os resultados esperados pelos testes e o que é esperado, calculado pela rede.

- A função para treino, foi definida de forma a poder mudar-se o algoritmo usado bem como os nodos interiores da rede. Estipulou-se que não seriam necessárias mais de 20000 iterações para determinar a solução, no entanto, definiram-se 3 repetições, para apurar não só o melhor caso de treino, mas também como método de garantir que o treino ocorre sempre á primeira invocação da função, com os parâmetros adequados. (Ver Figura 6)

```
treina_rede <- function(ninter = c(7,6,4), alg = "rprop+") {
  p <- package
  p$nnet <- neuralnet(formula, package$treino, hidden = ninter, rep = 3,
                      lifesign = "full", linear.output = FALSE,
                      stepmax = 20000, threshold = 0.01,algorithm = alg)
  return(p)
}
```

Figura 6 - Função que treina a rede

- A função *showCasosTeste* é utilizada para apresentar de forma informativa os resultados obtidos pelo cálculo das respostas para os casos de teste. Esta é responsável por criar um gráfico, contendo os pontos nos seus valores esperados, e corretos, é responsável ainda por apresentar a linha sobre a qual todos os pontos deviam recair, e finalmente, por arredondar os valores para valores que sejam mais facilmente entendidos pelo utilizador. (Ver Figura 7)

```
showCasosTeste <- function() {
  p = package
  p$nnet_result <- compute(package$nnet, teste.01)
  p$result <- data.frame(atual = package$teste$Tarefa,
                         previsao = p$nnet_result$net.result)
  temp <- scale(p$result$previsao)
  temp <- temp * 2 + 1

  print("RMSE:")
  print(rmse(round(p$result$atual * 3), round(temp)))
  plot(round(p$result$atual * 3), round(temp), xlab = "Tarefa Correta",
       ylab = "Tarefa Prevista",col = "red", main = "Tipo de Tarefa",
       pch = 16, cex = 1.2)
  abline(h = 1:3, v = 1:3, col = "lightgray", lty = 3)
  abline(a = 0, b = 1, lwd=2, col = "blue")
  points(round(p$result$atual * 3), round(temp), col = "red", pch = 16, cex = 1.2)
  return(p)
}
```

Figura 7 - Apresentação dos casos de treino

Determinar os 7 níveis de Exaustão

Através da especificação do problema, retira-se que os 7 níveis de exaustão que se pretendem determinar são definidos como:

1. Totalmente bem;

2. Responsivo, mas não no pico;

3. Ok, normal;

4. Em baixo de forma/do normal, a sentir-se em baixo;

5. Sentido moleza, perdendo o foco;

6. Muito difícil concentrar, meio tonto;

7. Incapaz de funcionar, pronto a "desligar".

Para determinar estes níveis procedeu-se da seguinte forma:

- Normalizou-se os dados da exaustão para que estejam numa melhor escala para o treino da rede neuronal (entre 0 e 1), tal como havia sido feito anteriormente na tarefa. (Ver Figuras 2 e 8)

```
normaliza_exaustao <- function(e) {
  e$Exaustao[e$Exaustao == 1] <- round(1/7, digits = 2)
  e$Exaustao[e$Exaustao == 2] <- round(2/7, digits = 2)
  e$Exaustao[e$Exaustao == 3] <- round(3/7, digits = 2)
  e$Exaustao[e$Exaustao == 4] <- round(4/7, digits = 2)
  e$Exaustao[e$Exaustao == 5] <- round(5/7, digits = 2)
  e$Exaustao[e$Exaustao == 6] <- round(6/7, digits = 2)
  e$Exaustao[e$Exaustao == 7] <- round(7/7, digits = 2)
  return(e)
}
```

Figura 8 - Normalização da Exaustão (0 e 1)

- Decidiu-se trocar a ordem dos dados, pois notou-se que alguns casos de treino relativos aos níveis de exaustão estavam aglomerados, sendo que ou seriam utilizados para efeito de treino, ou seriam utilizados ara efeitos de teste.
- Após recorrer ao "shuffle" dos elementos, notou-se que a dispersão destes poucos casos era maior, ocorrendo muito frequentemente tanto nos casos de teste como nos casos de treino, significando que se treina e teste a RNA para mais casos possíveis de ser cobertos. (Ver Figuras 3 e 9)

```
shuffle <- function(e) {
  e <- e[sample.int(nrow(e)), ]
  return(e)
}
```

Figura 9 - Baralhar os casos

- Tal como feito anteriormente, para que se consiga determinar a significância dos atributos, utiliza-se a função *regsubsets*. Esta diz quais os melhores parâmetros a considerar na determinação do *output* desejado, neste caso o nível de exaustão.
- Observou-se, tal como havia sido verificado na determinação da tarefa, que quando se verifica os melhores parâmetros para treino, estes variam consoante o número de casos considerados para treino, deste modo adicionou-se a capacidade de testar os melhores parâmetros relativos a todos os dados presentes no ficheiro de input, bem como apenas os dados considerados como grupo de treino. (Ver Figuras 4 e 10)

```
test_optimal <- function(e = "geral") {
  if (e == "treino")
    regg1 <- regsubsets(Exaustao ~ KDT + MA + MV + TBC + DDC + DMS + AED + ADMSL,
                        package$treino, nvmax = 10)
  else
    regg1 <- regsubsets(Exaustao ~ KDT + MA + MV + TBC + DDC + DMS + AED + ADMSL,
                        package$exaustao, nvmax = 10)
  print(summary(regg1))
}
```

Figura 10 - Teste de significância

- Definiu-se uma função para que sejam normalizados os níveis de exaustão, aplicando a função já referida (Figura 8). Baralharam-se também os casos, definindo-se quais os para treino e para teste, dependendo do que era recebido como parâmetros da função. São também carregados todos os dados a ser utilizados

através da passagem de um parâmetro como argumento que especifica o caminho do ficheiro que deverá ser considerado. (Ver Figura 11)

```
init_pacote <- function(tt, tst, path) {
  e <- abrir_xls(path)
  e <- normaliza_exaustao(e)

  e <- shuffle(e)
  e <- shuffle(e)

  pckt = NULL
  pckt$exaustao <- e
  pckt$treino   <- e[tt, ]
  pckt$teste    <- e[tst, ]

  return(pckt)
}
```

Figura 11 - Iniciar casos para treino/teste

Finalmente foram definidas funções para o treino e para a apresentação de resultados. A função para treino é igual á já apresentada (Figura 6). A função para apresentação dos resultados difere na já apresentada pois é redefinido os valores da exaustão para a sua escala normal (de 1 a 7) enquanto a anterior é definida para a tarefa que varia entre 1 e 3. Apresenta-se então a função que foi definida para a exposição da rede treinada. (Ver Figura 12)

```
showCasosTeste <- function() {
  p = package
  p$nnet_result <- compute(package$nnet, teste.01)
  p$result <- data.frame(atual = package$teste$Exaustao,
                         previsao = p$nnet_result$net.result)
  temp <- scale(p$result$previsao)
  temp <- round(temp * 6) + 1

  print("RMSE:")
  print(rmse(round(p$result$atual * 7), temp))
  plot(round(p$result$atual * 7), temp, xlab = "Valor Correto",
       ylab = "Valor Previsto",
       col = "red", main = "Niveis de Exaustao", pch = 16, cex = 1.2)
  abline(h = 1:7, v = 1:7, col = "lightgray", lty = 3)
  abline(a = 0, b = 1, lwd=2, col = "blue")
  points(round(p$result$atual * 7), temp, col = "red", pch = 16, cex = 1.2)
  return(p)
}
```

Figura 12 - Apresentação da rede treinada

Determinar existência ou não de Exaustão

Um segundo critério necessário na realização deste trabalho era a capacidade de deteção de existência ou ausência de fatiga/exaustão num utilizador. O trabalho apresentado de seguida apresenta apenas as alterações efetuadas relativamente a toda a especificação já desenvolvida para os casos de identificação de exaustão por níveis.

Para que fosse possível determinar a existência ou ausência de fatiga, adaptamos a escala definida anteriormente para os seguintes valores:

0. Utilizador não apresenta sinais de exaustão;
1. Utilizador apresenta sinais de exaustão.

A conversão da escala baseada em sete níveis distintos para a escala binária correu do seguinte modo:

1. Totalmente bem – Agora **0**;
2. Responsivo, mas não no pico – Agora **0**;
3. Ok, normal – Agora **0**;
4. Em baixo de forma/do normal, a sentir-se em baixo – Agora **1**;
5. Sentido moleza, perdendo o foco – Agora **1**;
6. Muito difícil concentrar, meio tonto – Agora **1**;
7. Incapaz de funcionar, pronto a "desligar" – Agora **1**.

Apresentar-se-á as alterações efetuadas nas funções de normalização dos valores de exaustão e apresentação dos gráficos com os resultados de teste.

- Esta nova versão da função desenvolvida para representar a exaustão, normaliza os valores binários em 0.5 e 1. Optámos por estes valores pois segundo referido nas aulas, o valor 0 têm uma significância acrescida, sendo que os resultados poderiam ser de alguma forma manipulados. (Ver Figura 13)

```
normaliza_exaustao <- function(e) {
  e$Exaustao[e$Exaustao == 1] <- 0.5
  e$Exaustao[e$Exaustao == 2] <- 0.5
  e$Exaustao[e$Exaustao == 3] <- 0.5
  e$Exaustao[e$Exaustao == 4] <- 1
  e$Exaustao[e$Exaustao == 5] <- 1
  e$Exaustao[e$Exaustao == 6] <- 1
  e$Exaustao[e$Exaustao == 7] <- 1
  return(e)
}
```

Figura 13 - Normalização da exaustão para identificação binária

- Para que fosse possível apresentar os resultados nesta nova escala binária, foi necessário recorrer a adaptação dos dados. Relembre-se que a rede neuronal é treinada com os valores normalizados, apenas a sua apresentação é efetuada com valores adaptados, para que mais facilmente seja identificável pelo utilizador. (Ver Figura 14)

```
showCasosTeste <- function() {
  p = package
  p$nnet_result <- compute(package$nnet, teste.01)
  p$result <- data.frame(atual = package$teste$Exaustao, previsao = p$nnet_result$net.result)

  temp <- scale(p$result$previsao)

  print("RMSE:")
  print(rmse(round(p$result$atual * 2 - 1), round(temp)))
  plot(round(p$result$atual * 2 - 1), round(temp), xlab = "Valor Correto", ylab = "Valor Previsto",
       col = "red", main = "Existência de Exaustão", pch = 16, cex = 1.2)
  abline(h = 0:1, v = 0:1, col = "lightgray", lty = 3)
  abline(a = 0, b = 1, lwd=2, col = "blue")
  points(round(p$result$atual * 2 - 1), round(temp), col = "red", pch = 16, cex = 1.2)
  return(p)
}
```

Figura 14 - Função de apresentação para a identificação binária

Determinar escala de identificação da Exaustão

O terceiro critério necessário na realização deste trabalho era a capacidade de deteção de existência de exaustão de acordo com uma escala que se considerasse apropriada. Seguidamente apresentar-se-á o trabalho associado a esta nova identificação, sendo que apenas se apresentarão as alterações efetuadas relativamente ao trabalho realizado para a identificação de exaustão por níveis.

Para que fosse possível determinar os novos níveis de exaustão, adaptamos a escala definida inicialmente para os seguintes valores:

1. Utilizador não apresenta sinais de exaustão;
2. Utilizador apresenta poucos sinais de exaustão;
3. Utilizador apresenta muitos sinais de exaustão.

Efetuou-se a conversão da escala baseada em sete níveis distintos para a escala binária do seguinte modo:

1. Totalmente bem – Agora 1;
2. Responsivo, mas não no pico – Agora 1;
3. Ok, normal – Agora 2;
4. Em baixo de forma/do normal, a sentir-se em baixo – Agora 2;
5. Sentido moleza, perdendo o foco – Agora 3;
6. Muito difícil concentrar, meio tonto – Agora 3;
7. Incapaz de funcionar, pronto a "desligar" – Agora 3.

Apresentar-se-á as alterações efetuadas nas funções de normalização dos valores de exaustão e apresentação dos gráficos com os resultados de teste.

- Esta nova versão da função desenvolvida para representar a exaustão, normaliza os valores binários em $\frac{1}{3}$, $\frac{2}{3}$ e 1. Posteriormente procede-se ao arredondamento com 2 casas decimais, utilizando, efetivamente os valores de 0.33, 0.66 e 1. Optámos por estes valores porque normalizando os dados, diminui-se o risco de ocorrência de qualquer tipo de tendências na representação dos resultados. (Ver Figura 15)

```
normaliza_exaustao <- function(e) {
  e$Exaustao[e$Exaustao == 1] <- round(1/3, digits = 2)
  e$Exaustao[e$Exaustao == 2] <- round(1/3, digits = 2)
  e$Exaustao[e$Exaustao == 3] <- round(2/3, digits = 2)
  e$Exaustao[e$Exaustao == 4] <- round(2/3, digits = 2)
  e$Exaustao[e$Exaustao == 5] <- round(3/3, digits = 2)
  e$Exaustao[e$Exaustao == 6] <- round(3/3, digits = 2)
  e$Exaustao[e$Exaustao == 7] <- round(3/3, digits = 2)
  return(e)
}
```

Figura 15 - Normalização da exaustão em escala

- Atualizou-se, novamente, a função responsável por apresentar os resultados obtidos pelo cálculo dos casos de teste, bem como o cálculo do erro. Deste modo garante-se que na escala apresentada são facilmente identificáveis os variados níveis de exaustão. (Ver Figura 16)

```
showCasosTeste <- function() {
  p = package
  p$nnet_result <- compute(package$nnet, teste.01)
  p$result <- data.frame(atual = package$teste$Exaustao, previsao = p$nnet_result$net.result)

  temp <- scale(p$result$previsao)
  temp <- round(temp * 2) + 1

  print("RMSE:")
  print(rmse(round(p$result$atual * 3), temp))
  plot(round(p$result$atual * 3), temp, xlab = "Valor Correto", ylab = "Valor Previsto",
       col = "red", main = "Escala de Exaustao", pch = 16, cex = 1.2)
  abline(h = 1:7, v = 1:7, col = "lightgray", lty = 3)
  abline(a = 0, b = 1, lwd=2, col = "blue")
  points(round(p$result$atual * 3), temp, col = "red", pch = 16, cex = 1.2)
  return(p)
}
```

Figura 16 - Função de apresentação para identificação em escala

Análise de Resultados

Apresenta-se agora os resultados de cada uma das redes neuronais desenvolvidas, comparando os seus resultados com os diferentes algoritmos e nodos que podiam ter sido utilizados. Para efeitos de teste, efetuar-se-á uma pequena alteração á função de apresentação inicial (Figura 17), para que sejam visíveis todos os valores obtidos, num novo gráfico (Figura 18).

Neste novo modelo, é possível identificar as respostas obtidas pela RNA antes de estas serem convertidas para os valores mais próximos, e, subsequentemente mais facilmente identificáveis.

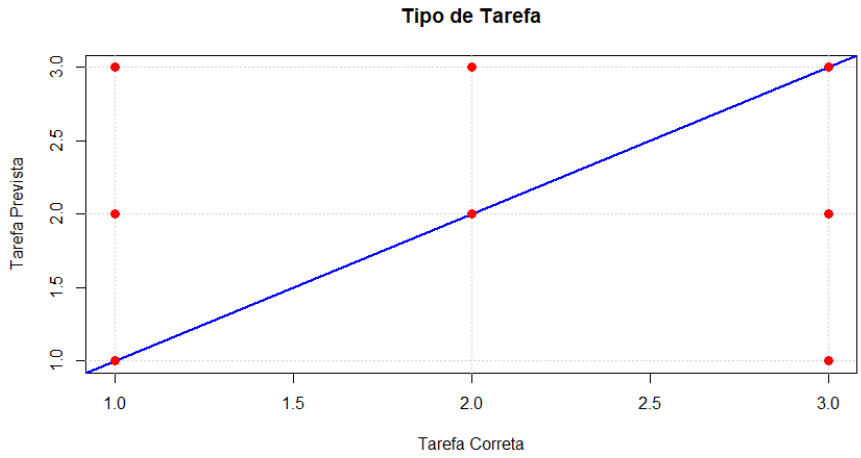

Figura 17 - Modelo inicial de apresentação dos dados

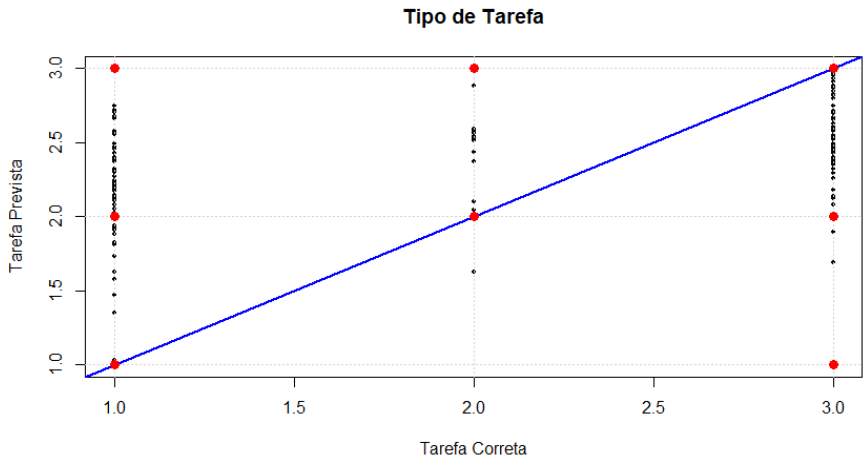

Figura 18 - Visualização dos dados e respetivos valores de ativação

Podemos variar os mais derivados aspetos para o treino das RNA's. De modo a encontrar o melhor resultado, procurou-se:

1. Variar o número de casos de treino;
2. Utilizar os melhores parâmetros, de acordo com todos os dados, ou apenas com os dados nos casos de teste;
3. Alterar o algoritmo para treino;
4. Alterar o número de neurónios e camadas intermédias.

Refira-se que todos os testes efetuados apresentam um *threshold* de 0.01 e utilizam os 3 melhores elementos para identificar o output.

Identificação de Tarefa

Casos treino	Treino com parâmetros ótimos para:	Algoritmo	Nós intermédios	Iterações	Erro de Aprendizagem	RMSE
700	Geral	rprop+	10	3950	23.761	0.917
700	Geral	rprop+	10, 5	12303	19.865	0.979
700	Treino	rprop+	10	6457	23.459	1.184
700	Treino	rprop+	10, 5	14132	19.790	1.239
700	Geral	backprop	10	2	68.345	1.000
700	Geral	backprop	10, 5	2	68.341	1.003
700	Geral	rprop-	10	5364	24.139	0.979
700	Geral	rprop-	10, 5	14280	19.901	0.975
600	Geral	rprop+	10	2775	20.240	0.947
600	Geral	rprop+	10, 5	14713	15.842	1.000

Tabela 1 - Testes efetuados para identificação de tarefa

Através das verificações efetuadas, retira-se:

1. Caso geral o erro *rmse* aumenta com um maior número de nós intermédios, apesar do erro de aprendizagem diminuir;
2. Dever-se-á treinar a rede com os parâmetros globais ótimos;
3. Um maior número de casos de treino leva a error menores;
4. O melhor algoritmo para aprendizagem é o rprop+.

Apresentam-se agora o gráfico dos resultados calculados a partir da rede e a topologia da rede gerada. (Figuras 19 e 20)

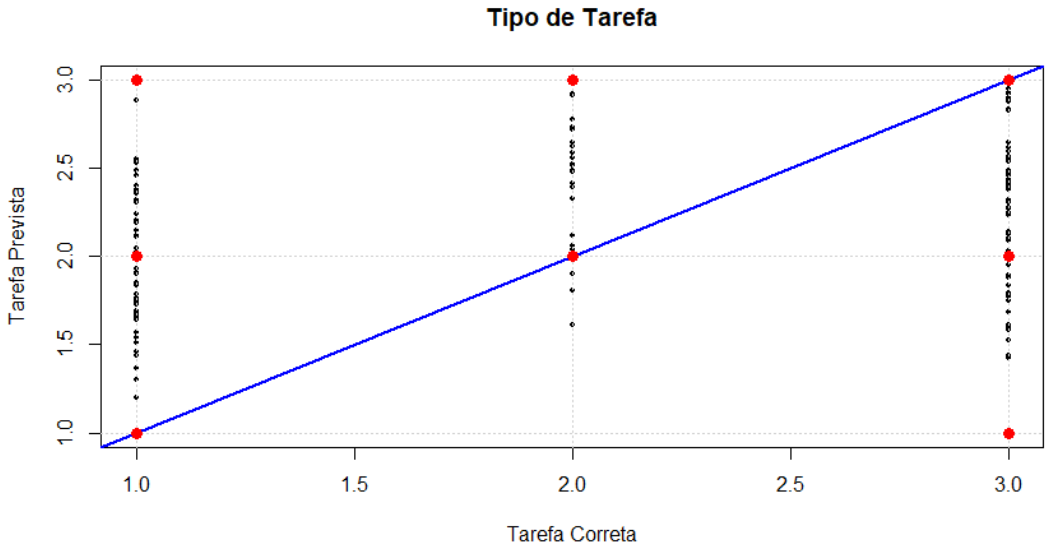

Figura 19 - Caso ótimo para identificação da tarefa

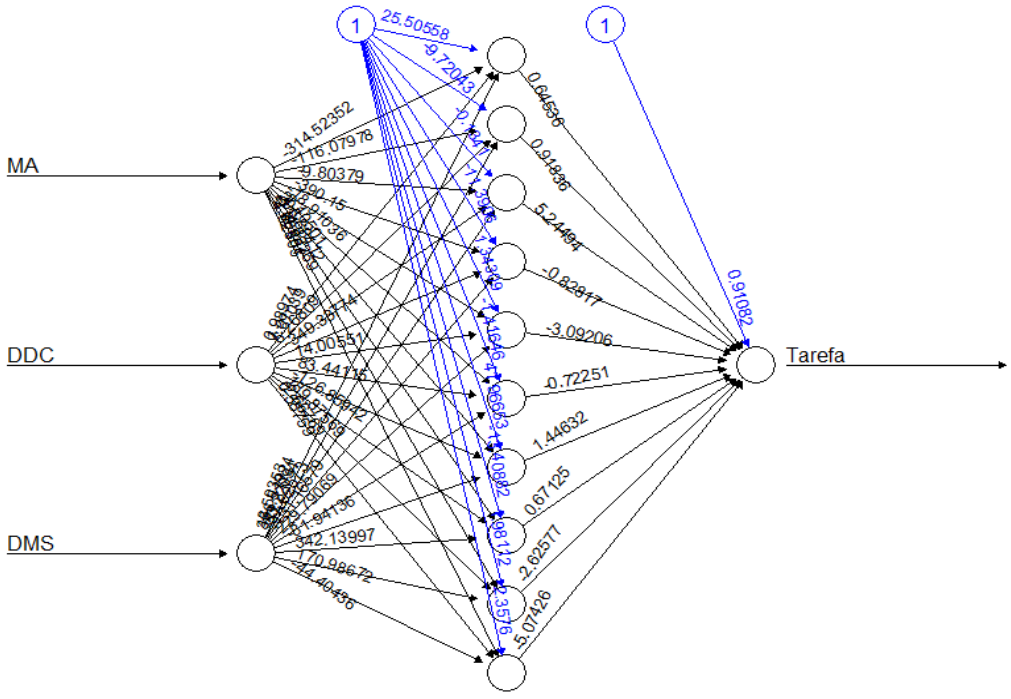

Figura 20 - Rede neuronal artificial gerada para identificação da tarefa

Identificação dos níveis de Exaustão

Casos treino	Treino com parâmetros ótimos para:	Algoritmo	Nós intermédios	Iterações	Erro de Aprendizagem	RMSE
700	Geral	rprop+	10	1427	7.523	2.599
700	Geral	rprop+	10, 5	12473	6.088	2.043
700	Geral	rprop+	7, 5, 4	12260	6.036	1.567
700	Treino	rprop+	7, 5, 4	3346	7.001	2.233
700	Geral	backprop	10, 5	6	52.970	1.822
700	Geral	rprop-	10	2558	7.422	3.295
700	Geral	rprop-	10, 5	5818	6.541	1.258
600	Geral	rprop-	10, 5	4505	5.660	2.482
800	Geral	rprop-	10, 5	7203	7.421	1.977

Tabela 2 - Testes efetuados para a identificação dos níveis de exaustão

Através das verificações efetuadas, conclui-se que:

1. Os melhores parâmetros de treino são os dos dados completos. Utilizando parâmetros ótimos para treino a rede não converge, ou tem um erro maior;
2. Uma variação na quantidade de casos significa aumento no erro;
3. O melhor algoritmo foi o rprop- com um erro associado de 1.25;
4. Introduzir mais que 3 níveis intermédios causa um aumento no erro.

De acordo com os resultados obtidos na tabela 2, apresentam-se agora os diagramas da rede neuronal obtida com o algoritmo que obteve um *rmse* menor, e o respetivo gráfico de resultado. (Figuras 21 e 22)

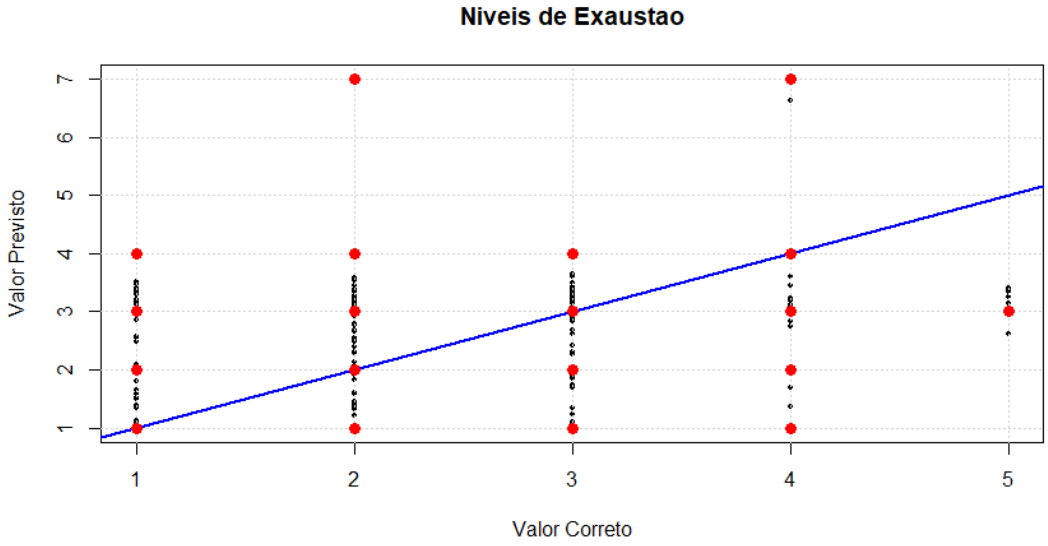

Figura 21 - Caso ótimo para identificar níveis de exaustão

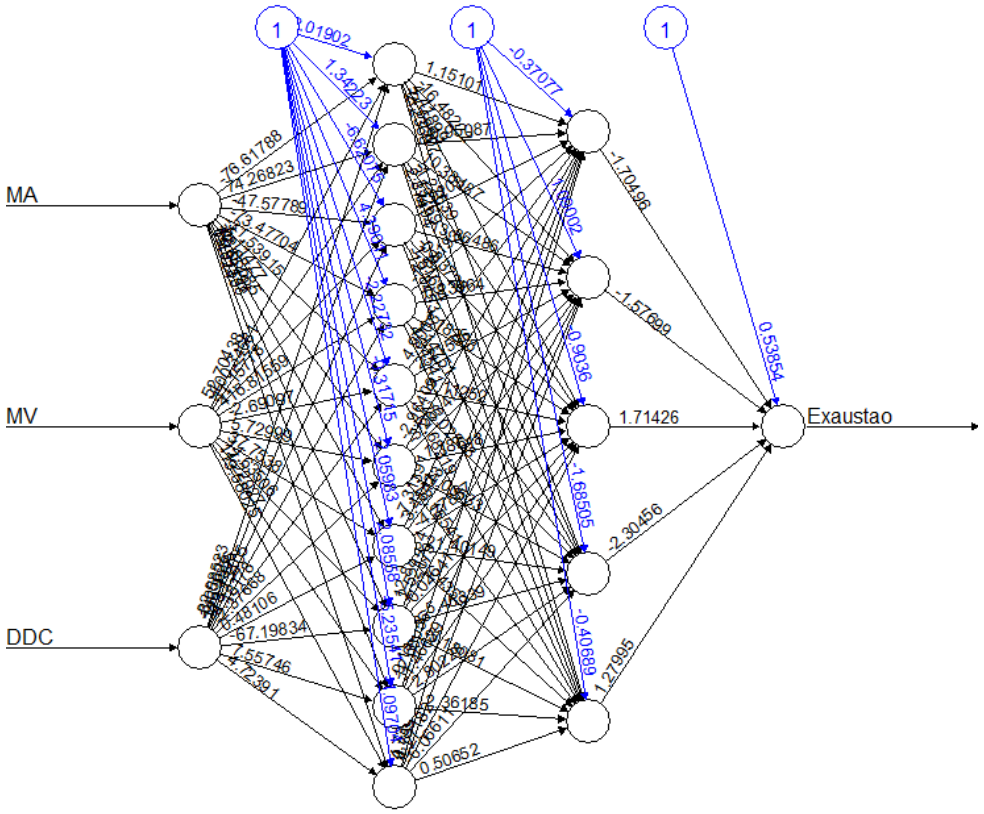

Figura 22 - Rede neuronal associada ao caso ótimo de identificação dos níveis de exaustão

Identificação de existência de Exaustão

Casos treino	Treino com parâmetros ótimos para:	Algoritmo	Nós intermédios	Iterações	Erro de Aprendizagem	RMSE
700	Geral	rprop+	10	2409	11.241	0.672
700	Geral	rprop+	10, 5	16876	8.527	0.527
700	Geral	rprop+	7, 5, 4	17664	8.829	0.449
700	Geral	backprop	7, 5, 4	2	70.624	0.479
700	Treino	rprop+	10, 5	4977	9.983	0.624
700	Treino	rprop-	10, 5	9386	8.726	0.712
600	Geral	rprop+	7, 5, 4	7151	7.287	0.457
800	Geral	rprop+	7, 5, 4	15867	11.327	0.477

Tabela 3 - Testes efetuados para verificação de existência de exaustão

Através dos testes efetuados, poder-se-á concluir o seguinte:

1. Os melhores parâmetros de treino são os dos dados completos, tal como observado anteriormente. Utilizando parâmetros ótimos para treino a rede não converge, ou tem um erro maior;
2. Uma variação na quantidade de casos significa uma alteração pequena no erro;
3. O melhor algoritmo foi o rprop+ com um erro associado de 0.449, os algoritmos rprop- e backprop raramente convergiram;
4. No caso do rprop+, quanto mais níveis intermédios, menor o erro no cálculo dos valores.

De acordo com os resultados obtidos na tabela 3, apresentam-se agora os diagramas associados á rede neuronal obtida e o respetivo gráfico de resultado contendo as soluções geradas pela RNA assim como os valores aproximados na escala definida anteriormente. (Figuras 23 e 24)

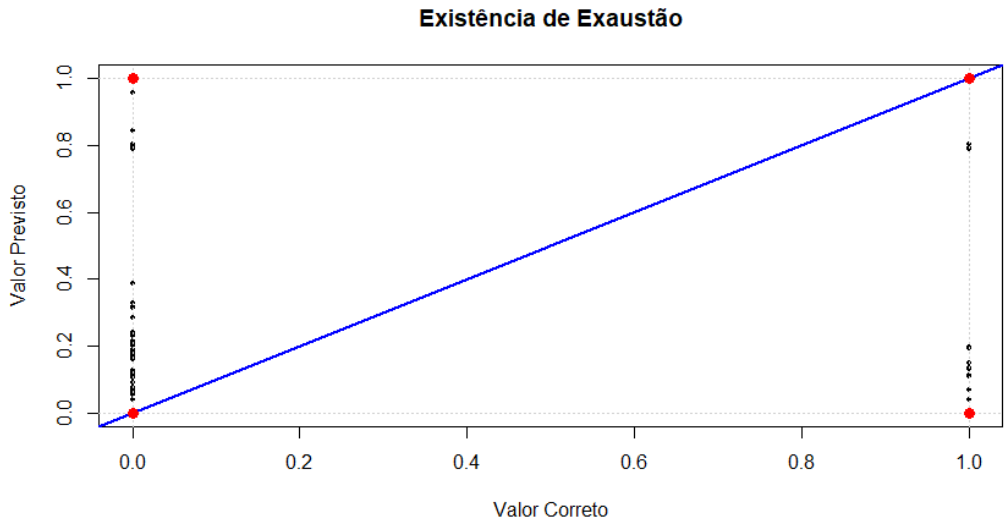

Figura 23 - Caso ótimo para identificar presença de exaustão

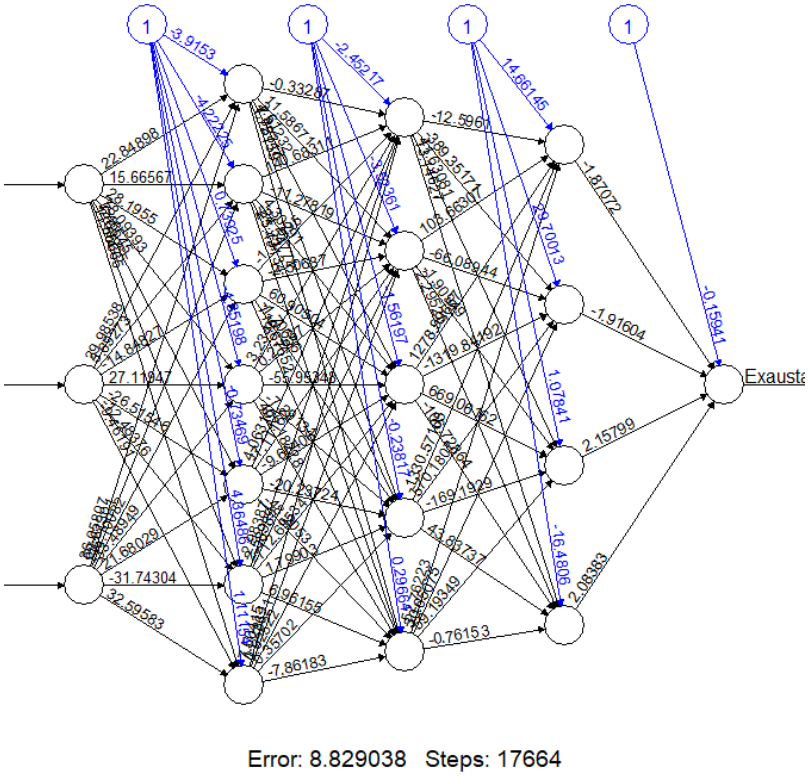

Figura 24 - Rede gerada para o caso de teste de existência de sinais de exaustão

Identificação da escala de Exaustão

Casos treino	Treino com parâmetros ótimos para:	Algoritmo	Nós intermédios	Iterações	Erro de Aprendizagem	RMSE
700	Treino	rprop+	10	5596	8.7899	0.76
700	Treino	rprop+	10, 5	17287	7.49787	1.22
700	Geral	rprop+	10, 5	6938	7.30634	0.802
700	Geral	rprop+	7, 5, 4	18109	8.985274	0.671
700	Treino	rprop-	7, 5, 4	7593	7.14867	0.733
700	Geral	rprop-	7, 5, 4	8537	7.59718	0.896
700	Geral	rprop-	10, 5	3535	7.92202	0.763
700	Geral	rprop-	10	7069	8.78976	0.917

Tabela 4 - Testes efetuados para verificação de existência de exaustão

Através das verificações efetuadas, conclui-se que:

1. Os melhores parâmetros de treino são os dos dados completos. Utilizando parâmetros ótimos para treino a rede não converge, ou tem um erro maior;
2. Uma variação na quantidade de casos significa aumento no erro;
3. O melhor algoritmo foi o rprop+ com um erro associado de 0.671, o algoritmo backprop não foi apresentado porque não convergia e em alguns casos resultava num erro muito superior ao já testado.
4. Introduzir mais que 3 níveis intermédios, causa um aumento no erro.

De acordo com os resultados obtidos na tabela 4, apresentam-se agora os diagramas da rede neuronal obtida com o algoritmo que obteve um *rmse* menor, e o respetivo gráfico que representa a comparação entre os resultados que a rede neuronal devia dar e os que esta realmente deu. (Figuras 25 e 26)

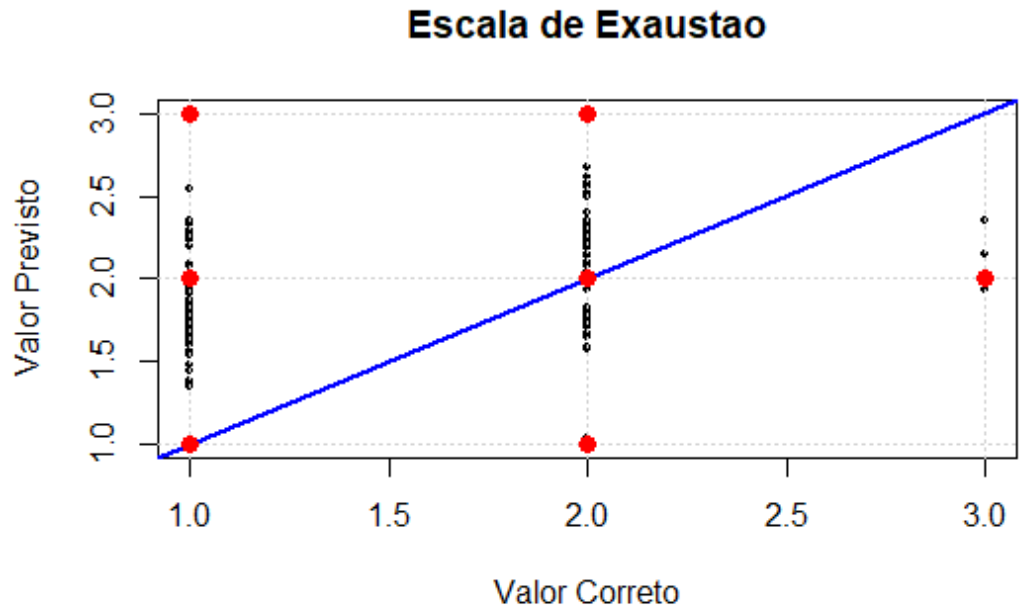

Figura 25 - Caso ótimo para identificar níveis de exaustão

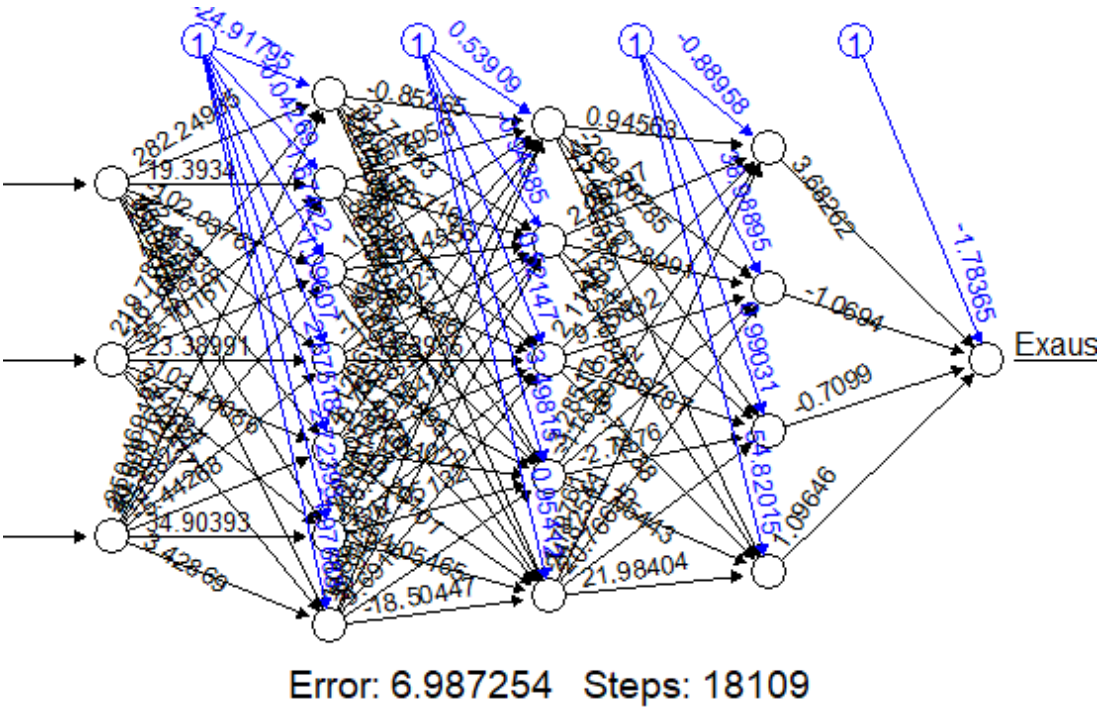

Figura 26 - Caso ótimo para identificar escala de exaustão

Conclusão

Neste trabalho foi solicitada a criação de redes neuronais para determinar qual a tarefa a executar por uma pessoa, ou o seu nível de exaustão. Assim, foram criadas várias redes neuronais, para determinar a tarefa, os 7 níveis de exaustão, a existência ou não de exaustão e para a melhor escala para a exaustão.

Foram encontradas algumas dificuldades até chegar a um resultado aceitável devido ás grandes discrepâncias entre o que as redes davam como resultado ás *queries* efetuadas e o seu resultado real.

Tendo isso em conta, o resultado final, foi conseguido de forma aceitável, sendo que as redes treinadas têm um erro associado a estas mesmo que consideramos o mínimo possível testado.