UNIVERSIDADE DO MINHO

Conhecimento Não Simbólico: Redes Neuronais Artificiais.

Mestrado Integrado em Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio
(2º semestre, Ano Letivo 2016/17)

A75135 Bruno Pereira
A73580 Maria Brito
A27748 Gustavo Andrez
A74634 Rogério Moreira
A76507 Samuel Ferreira

Braga,
2017-05-21

Este relatório tem como objetivo, descrever e explicar todas as etapas da resolução do terceiro exercício prático da Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio, que visa a utilização de sistemas não simbólicos na representação de conhecimento e no desenvolvimento de mecanismos de raciocínio, nomeadamente, Redes Neuronais Artificiais (RNAs) para a resolução de problemas.

Este relatório fornece toda a informação necessária sobre os resultados obtidos e permite a compreensão dos mesmos. Para além da estratégia usada para resolver o problema, são também caracterizadas as decisões tomadas.

Índice

1. Introdução ao Exercício Prático

a. Introdução

O exercício prático descrito neste relatório consiste na utilização de várias RNAs (Redes Neuronais Artificiais) com o objetivo de identificar níveis de fadiga de acordo com uma fórmula que utiliza parâmetros relativos à utilização do rato e do teclado em diversos cenários. Assim, procedeu-se à realização de diversos testes com amostras de dados de aprendizagem e de teste diferentes, assim como com redes neuronais com características distintas. De seguida, foi feita uma seleção das melhores redes para utilizar nos futuros testes com escalas ideais e identificação de existência e inexistência de exaustão.

É de referir que foram utilizados ficheiros com os dados representados de formas diferentes, como por exemplo, dados normalizados, dados não normalizados, bem como dados organizados (organizados aleatoriamente) e dados não organizados.

b. Motivação e Objetivos

O exercício proposto na Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio surge como uma oportunidade para colocar em prática os conhecimentos adquiridos ao longo das aulas práticas e teóricas.

Assim, o principal objetivo consiste na utilização de Redes Neuronais Artificiais para a identificação do nível de fadiga de um ser humano, tendo em conta métricas relativas à utilização do teclado e do rato do computador em diversos cenários de teste.

c. Descrição dos Parâmetros de Input

Neste exercício trabalhamos com redes neuronais artificias, em R, para tentar prever o nível de fadiga de um sujeito. Isto é possível devido ao treino prévio da rede neuronal, que é baseado num conjunto de dados iniciais. Após a rede estar treinada, prevê-se então a fadiga da pessoa, recebendo como parâmetros de entrada:

- **Performance.KDTMean** – tempo médio entre o momento em que a tecla é pressionada para baixo e o momento em que é largada;

- **Performance.MAMean** – aceleração do manuseamento rato em determinado momento. O valor da aceleração é calculado através da velocidade do rato (pixel/milissegundos) sobre o tempo de movimento (milissegundos);

- **Performance.MVMean** – velocidade do manuseamento do rato em determinado momento. A distância percorrida pelo rato (em pixéis) entre uma coordenada C1 (x1; y1) e uma C2 (x2; y2) correspondentes a time1 e time2, sobre o tempo (em milissegundos);

- **Performance.TBCMean** – tempo entre dois cliques consecutivos, entre eventos consecutivos MOUSE_UP e MOUSE_DOWN;

- **Performance.DDCMean** – período de tempo entre dois eventos MOUSE_UP consecutivos;

- **Performance.DMSMean** – distância média em excesso entre o caminho de dois cliques consecutivos;

- **Performance.ADMSLMean** – distância média das diferentes posições do ponteiro entre dois pontos durante um movimento, e o caminho em linha reta entre esses mesmos dois pontos;

- **Performance.AEDMean** – esta métrica é semelhante à anterior, no sentido em calculará a soma da distância entre dois eventos MOUSE_UP e MOUSE_DOWN consecutivos;

d. Estrutura do Relatório

O relatório é constituído por 3 partes. A primeira parte, menciona os principais objetivos e a motivação ao desenvolvimento do exercício prático, bem como uma pequena introdução ao projeto desenvolvido. Na segunda explica-se qual foi o raciocínio do grupo a resolver os vários desafios propostos, assim como se apresentam vários resultados de testes realizados e faz-se a sua análise. Por fim, na terceira parte tecem-se as conclusões finais.

2. Descrição do trabalho

a. Representação dos Dados de Entrada

Os dados iniciais fornecidos pela equipa docente foram tratados recorrendo a funcionalidades do Excel (como, por exemplo, a função ALEATORIO()) e a funções do R (como, por exemplo, *discretize*). Decidiu-se normalizar os dados, por forma a melhorar a sua representação, tornar os valores contínuos das medições menos dispersos, facilitar os cálculos das redes neuronais utilizadas e promover uma melhor compreensão dos resultados. De seguida, explicam-se as várias representações de dados que se utilizaram nos diversos testes com redes neuronais.

i. Dados Normalizados e Aleatórios

Os dados encontram-se organizados de forma aleatória, graças à função ALEATORIO() do Excel. Assim, de forma a proporcionar uma distribuição mais uniforme do que a dos dados originais, utilizou-se esta função para obter dados que estivessem mais distribuídos. Este aspeto é muito importante, uma vez que quando a rede neuronal aprender com um subconjunto de dados, a aprendizagem será mais robusta, pois terá mais casos de estudo diferentes.

Além disso, os dados também se encontram normalizados em valores discretos que estão compreendidos entre 0.1 e 1. Isto foi possível graças à função *discretize* da linguagem R, que dividiu os valores de entrada em 10 intervalos e, de seguida, renomeou-os com os valores mencionados. Assim, os valores de entrada originais, que, devido ao grande número de casas decimais, podemos assumir que são contínuos, tornam-se discretos,oq eu facilita o seu processamento através de redes neuronais.

A *Task*, por apenas permitir três valores diferentes (*work, programming, office*), não foi sujeita à normalização referida acima. Os seus valores podem ser 0.1, 0.2 ou 0.3, que representam *work, programming* e *office*, respetivamente.

Os valores de *output* (níveis de exaustão) também foram normalizados para um intervalo entre 0.1 e 0.6, que indicam o nível de exaustão 1 e 6, respetivamente.

Desta forma, com este conjunto de dados, obtém-se uma representação mais organizada e distribuída e os dados estão normalizados em valores contínuos, que vai possibilitar menores erros de aprendizagem e variação dos resultados previstos.

De seguida, seguem-se os intervalos calculados pelo R para a discretização dos valores iniciais e os respetivos valores que esses intervalos vão tomar nesta representação de dados.

KDT	MA
0.1=>[-1,-0.006366)	0.1=>[-0.75169, -0.16103)
0.2=>[-0.006366, -0.001216)	0.2=>[-0.16103, -0.08019)
0.3=>[-0.001216, -0.000581)	0.3=>[-0.08019, -0.04280)
0.4=>[-0.000581, 0.000113)	0.4=>[-0.04280, -0.01807)
0.5=>[0.000113, 0.001013)	0.5=>[-0.01807, -0.00179)

0.6=>[0.001013, 0.002433)
0.7=>[0.002433, 0.005972)
0.8=>[0.005972, 0.022155)
0.9=>[0.022155, 0.078921)
1.0=>[0.078921, 1)

0.6=>[-0.00179, 0.00293)
0.7=>[0.00293, 0.02095)
0.8=>[0.02095, 0.04661)
0.9=>[0.04661, 0.09338)
1.0=>[0.09338, 1)

MV

0.1=>[-0.49703, -0.07737)
0.2=>[-0.07737, -0.04329)
0.3=>[-0.04329, -0.02070)
0.4=>[-0.02070, -0.00335)
0.5=>[-0.00335, -0.00060)
0.6=>[-0.00060, 0.00110)
0.7=>[0.00110, 0.00830)
0.8=>[0.00830, 0.02792)
0.9=>[0.02792, 0.06463)
1.0=>[0.06463, 1)

TBC

0.1=>[-0.7296433, -0.0553351)
0.2=>[-0.0553351, -0.0177817)
0.3=>[-0.0177817, -0.0073175)
0.4=>[-0.0073175, -0.0036850)
0.5=>[-0.0036850, 0.0000548)
0.6=>[0.0000548, 0.0070465)
0.7=>[0.0070465, 0.0156319)
0.8=>[0.0156319, 0.0368104)
0.9=>[0.0368104, 0.1113655)
1.0=>[0.1113655, 1)

DDC

0.1=>[-0.28060 , -0.04088)
0.2=>[-0.04088, -0.01942)
0.3=>[-0.01942, -0.00912)
0.4=>[-0.00912, -0.00225)
0.5=>[-0.00225, 0.01516)
0.6=>[0.01516, 0.03296)
0.7=>[0.03296, 0.06443)
0.8=>[0.06443, 0.13993)
0.9=>[0.13993, 0.24444)
1.0=>[0.24444, 1)

DMS

0.1=>[-0.300857 , -0.025850)
0.2=>[-0.025850, -0.015673)
0.3=>[-0.015673, -0.009484)
0.4=>[-0.009484, -0.004853)
0.5=>[-0.004853, -0.000832)
0.6=>[-0.000832, 0.003349)
0.7=>[0.003349, 0.11911)
0.8=>[0.11911, 0.025402)
0.9=>[0.025402, 0.059053)
1.0=>[0.059053, 1.00000)

AED

0.1=>[-0.223617, -0.032666)
0.2=>[-0.032666, -0.018896)
0.3=>[-0.018896, -0.011591)
0.4=>[-0.011591, -0.006291)
0.5=>[-0.006291, -0.000396)
0.6=>[-0.000396, 0.003046)
0.7=>[0.003046, 0.011515)
0.8=>[0.011515, 0.023510)
0.9=>[0.023510, 0.060197)
1.0=>[0.060197, 1)

ADMSL

0.1=>[-0.43672, -0.16156)
0.2=>[-0.16156, -0.09049)
0.3=>[-0.09049, -0.04947)
0.4=>[-0.04947, -0.02492)
0.5=>[-0.02492, -0.00581)
0.6=>[-0.00581, 0.02440)
0.7=>[0.02440, 0.06630)
0.8=>[0.06630, 0.12193)
0.9=>[0.12193, 0.25324)
1.0=>[0.25324, 1)

ii.Alguns Dados Normalizados e Aleatórios

Neste caso, apenas os dados relativos à exaustão (*FatigueLevel)* e à tarefa (*Task*) encontram-se normalizados. Os primeiros encontram-se compreendidos entre os valores

0.1 e 0.6, que indicam o primeiro e o sexto nível de exaustão, respetivamente. Aos segundos, para o *work* foi atribuído o valor 0.1, para o *programming*, o valor 0.2 e, por fim, para o *office*, o valor 0.3. Os restantes dados de entrada não se encontram normalizados, sendo, então, valores contínuos entre -1 e 1.

Por outro lado, todos os dados encontram-se organizados de forma aleatória, que promove uma distribuição mais homogénea dos dados.

iii. Dados Não Normalizados e Aleatórios

Neste caso, os dados de entrada que medem valores do *mouse* e do *keyboard* encontram-se iguais aos originais, isto é, são contínuos (muitas casas decimais) e estão compreendidos entre -1 e 1.

Por outro lado, para a tarefa (*Task*) foram atribuídos números inteiros ao *work, programming* e *office*, sendo eles, 1, 2 e 3, respetivamente. Denote-se que este intervalo de valores inteiros é significativamente maior do que o intervalo acima mencionado (entre 0.1 e 0.3), logo não se encontram normalizados, apenas converteu-se as *strings* para números inteiros correspondentes.

Em relação aos níveis de exaustão, estes também se encontram iguais aos originais, ou seja, cada nível de exaustão está representado com um número inteiro correspondente (ao primeiro nível foi atribuído o número 1, ao segundo foi atribuído o número 2, …).

Desta forma, obtém-se, assim, uma representação dos dados muito próxima dos dados originais, que apenas difere na sua organização, uma vez que estes estão organizados de forma aleatória.

iv. Dados Não Normalizados e Não Aleatórios

A representação dos dados neste caso é muito semelhante à anterior, diferindo na sua organização, uma vez que estes não se encontram organizados de forma aleatória. Por esta razão, é a representação mais próxima da originalmente fornecida.

b. Redes Neuronais Testadas

De forma a abranger o maior número de possibilidades possível, o grupo definiu várias redes neuronais com topologias diferentes para testar os dados de entrada acima mencionados.

i. Rede 1

A primeira rede neuronal definida possui três camadas intermédias. A primeira tem 60 neurónios, a segunda 40 neurónios e a terceira 20 neurónios. Além disso, a função de ativação não está ativa (*linear.output=TRUE)*, logo são permitidas grandes oscilações nos neurónios de *output*.

ii. Rede 2

Esta rede neuronal definida possui três camadas intermédias. A primeira tem 60 neurónios, a segunda 40 neurónios e a terceira 20 neurónios. Além disso, a função de ativação está ativa (*linear.output=FALSE)*, logo não são permitidas grandes oscilações nos neurónios de *output*.

iii.Rede 3

A terceira rede neuronal possui também três camadas intermédias. A diferença é que em cada uma delas, o número de neurónios é o dobro das anteriores, isto é, a primeira camada possui 120 neurónios, a segunda 80 neurónios e a terceira 40 neurónios.

Além disso, a função de ativação não está ativa (*linear.output=TRUE)*, logo são permitidas grandes oscilações nos neurónios de *output*.

iv.Rede 4

A quarta rede neuronal possui também três camadas intermédias, em que a primeira camada possui 6 neurónios, a segunda 4 neurónios e a terceira 2 neurónios.

Além disso, a função de ativação não está ativa (*linear.output=TRUE)*, logo são permitidas grandes oscilações nos neurónios de *output*.

v.Rede 5

A quinta rede neuronal possui duas camadas intermédias, em que a primeira camada possui um total de 10 neurónios e a segunda metade do valor da primeira, ou seja, 5 neurónios.

Além disso, a função de ativação não está ativa (*linear.output=TRUE)*, logo são permitidas grandes oscilações nos neurónios de *output*.

vi.Rede 6

A sexta rede neuronal possui apenas uma camada intermédia com 20 neurónios.

Além disso, a função de ativação não está ativa (*linear.output=TRUE)*, logo são permitidas grandes oscilações nos neurónios de *output*.

c. Escalas

Os níveis de exaustão mental apresentados no enunciado deste trabalho prático estão divididos em sete níveis, segundo a escala *USAFSAM Fatigue scale*, sendo eles os seguintes:

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”.

Um dos desafios propostos era identificar a existência ou não existência de exaustão e para tal efeito, foi necessário converter a escala de 7 níveis para uma escala de 2 níveis. Assim, a conversão consistiu em converter os valores da escala, ou em 0 (não

exausto), ou em 0.1 (exausto). De seguida, apresentam-se os valores da escala *USAFSAM Fatigue scale* e os novos valores que eles vão tomar na escala de dois níveis.

- 0 (não exausto): {1,2,3};
- 0.1 (exausto): {4,5,6,7}.

O outro desafio consistia em definir uma escala ideal para a identificação de exaustão. Para este propósito, o grupo definiu duas novas escalas.

A escala de 3 níveis consiste na seguinte transformação da escala de *USAFSAM Fatigue scale:*

- 0 (não se encontra exausto): {1, 2, 3};
- 0.1 (encontra-se em risco de exaustão): {4,5};
- 0.2 (encontra-se em exausto): {6, 7}.

A escala de 4 níveis consiste na seguinte transformação da escala de *USAFSAM Fatigue scale:*

- 0 (encontra-se num estado físico e mental excelente): {1};
- 0.1 (encontra-se num estado normal): {2, 3};
- 0.2 (encontra-se cansado, mas não em exaustão completa): {4, 5};
- 0.3 (encontra-se num estado de exaustão): {6,7}.

d. Metodologia Seguida

Visando uma maior compreensão do raciocínio seguido pelo grupo no desenvolvimento deste exercício prático, de seguida são enumerados os passos que compõe a metodologia criada:

1. Análise dos dados fornecidos pela equipa docente.
2. Normalização e organização dos dados. Deste passo, temos os dados representados de formas distintas.
3. Divisão do *dataset* em vários subconjuntos para aprendizagem da rede e treino.

	1º Cenário	2º Cenário
Número de dados de aprendizagem	600	244
Número de dados de treino	422	422

4. Definição de redes com topologias diferentes.

	1ª Rede	2ª Rede
Camadas Interiores	(60, 40, 20)	(60, 40, 20)
Linear.output	FALSE	TRUE

5. Verificação dos atributos mais importantes para os vários tipos de dados.
6. Testes com as redes (passo 4), volume de dados (passo 3) e fórmulas diferentes (passo 5)
7. Escolha da rede com a topologia e volume de dados que possuem melhores resultados.
8. Introdução de redes com camadas intermédias diferentes e linear.output = TRUE.

	1ª Rede	2ª Rede	3ª Rede	4ª Rede
Camadas interiores	(120, 80, 40)	(6, 4, 2)	(10, 5)	20

9. Testes com dados do passo 7 e redes do passo 8.
10. Definição de escalas de existência/inexistência de exaustão, três níveis de exaustão e quatro níveis de exaustão.
11. Testes com novas escalas e volume de dados do passo 7, camadas interiores (120, 80, 40) e (60, 40, 20) e alternando valor booleano de linear.output.

e. Análise das Topologias de Rede

i.Eliminação de Representações de Dados

No início do desenvolvimento do exercício prático, o grupo prático analisou os dados fornecidos pela equipa docente e, a partir daí, desenvolveu quatro representações de dados distintas. De forma a selecionar apenas duas representações, foram feitos testes com volume de dados distintos (1º Cenário e 2º Cenário) e analisados os erros de aprendizagem e desvio padrão dos resultados previstos pela rede. Foram utilizadas a primeira e segunda redes, conforme mencionado na Metodologia.

File	nº camadas	nº neurónios	nº parâmetros	Dados aprendizagem	Dados treino	thresh	L.O	error	rmse
1	3	60/40/20	9	600	244	0,01	F	0,0952	0,3085
1	3	60/40/20	9	600	244	0,01	T	0,0797	0,3599
1	3	60/40/20	9	422	422	0,01	F	0,0478	0,3127
1	3	60/40/20	9	422	422	0,01	T	0,0224	0,3424

Figura 1 - Resultados dos Dados Normalizados e Aleatórios

File	nº camadas	nºnodos	nº parâmetros	Dados aprendizagem	Dados treino	thresh	L.O.	error	rmse
2	3	60/40/20	9	600	244	0,01	F	0,07513	0,3377
2	3	60/40/20	9	600	244	0,01	T	0,04168	0,3377
2	3	60/40/20	9	422	422	0,01	F	0,0526	0,3012
2	3	60/40/20	9	422	422	0,01	T	0,01718	0,3786

Figura 2 - Resultados da Task e FatigueLevel Normalizados e Dados Aleatórios

File	nº camadas	nºnodos	º parâmetr	ados aprendizage	Dados treino	thresh	L.O.	error	rmse
3	3	60/40/20	9	600	244	0,01	F	1024,5069	1,862
3	3	60/40/20	9	600	244	0,01	T	não convergiu	não convergiu
3	3	60/40/20	9	422	422	0,01	F	744,5051	1,8253
3	3	60/40/20	9	422	422	0,01	T	0,00459	1,624

Figura 3 - Resultados dos Dados Não Normalizados e Aleatórios

File	nº camadas	nº neurónios	nº parâmetros	Dados aprendizagem	Dados treino	thresh	L.O	error	rmse
2	3	60/40/20	9	600	244	0,01	F	1114,0034	1,6533
2	3	60/40/20	9	600	244	0,01	T	0,0014	1,9045
2	3	60/40/20	9	422	422	0,01	F	808,006	1,7409
2	3	60/40/20	9	422	422	0,01	T	0,0016	1,7959

Figura 4 - Resultados dos Dados Não Normalizados e Não Aleatórios

Tal como se pode verificar, pode-se agrupar os quatro tipos de representação de dados em dois grupos: o primeiro grupo engloba os dados normalizados e aleatórios e os dados aleatórios que contêm a *Task* e a *FatigueLevel* normalizadas e o segundo engloba os dados não normalizados aleatórios e não aleatórios.

Do primeiro grupo verifica-se que tanto o erro de aprendizagem e os desvios padrão são baixos. No entanto, no segundo grupo verifica-se que ocorrem erros de aprendizagem extremamente altos e desvios padrão acima de uma unidade. Para este caso de estudo, tais valores não são bons para proceder com estes tipos de representação de dados para a próxima fase de testes.

Deste modo, descarta-se, então, as representações de dados em que estes não estão normalizados. Os valores tão elevados de erros destas representações devem-se ao facto de os valores de entrada correspondentes às medições serem contínuos (dentro do intervalo -1 e 1), assim como os valores representativos da fatiga estejam compreendidos entre 1 e 6 (7, na teoria) e as tarefas terem a si associadas os valores inteiros 1, 2 ou 3 (que correspondem ao *work, programming* e *office*, respetivamente). Assim, pode-se verificar que estes valores estão ambos compreendidos em intervalos de extensão grande.

Consequentemente, os dados que serão sujeitos a mais testes são os dados normalizados e aleatórios e os dados aleatórios com *Task* e *FatigueLevel* normalizados.

ii.Escolha das variáveis mais significativas

Após terem sido eliminados dois ficheiros que apresentam erros muito superiores aos restantes, o objetivo é melhorar os valores obtidos na secção anterior. O primeiro passo para atingir esse fim será escolher as variáveis mais significativas para o problema e remover as restantes, de modo a melhorar a capacidade de aprendizagem das redes.

O *package leaps* do R possui uma função chamada *regsubsets* que, através de procuras extensivas, seleciona os melhores modelos, para um conjunto de dados. Assim, esta função será utilizada para cada um dos conjuntos de dados contidos nos dois ficheiros, para determinar quais as variáveis mais importantes.

Analisemos o caso dos dados normalizados e aleatórios, pois a análise é equivalente para o outro ficheiro. De seguida, segue-se o resultado da aplicação da função *regsubsets* para os dados normalizados e aleatórios.

```
                    Forced in Forced out
Performance.KDTMean       FALSE      FALSE
Performance.MAMean        FALSE      FALSE
Performance.MVMean        FALSE      FALSE
Performance.TBCMean       FALSE      FALSE
Performance.DDCMean       FALSE      FALSE
Performance.DMSMean       FALSE      FALSE
Performance.AEDMean       FALSE      FALSE
Performance.ADMSLMean     FALSE      FALSE
Performance.Task          FALSE      FALSE
1 subsets of each size up to 9
Selection Algorithm: exhaustive
         Performance.KDTMean Performance.MAMean Performance.MVMean Performance.TBCMean Performance.DDCMean
1  ( 1 ) " "                 " "                " "                " "                " "
2  ( 1 ) " "                 "*"                " "                " "                " "
3  ( 1 ) "*"                 "*"                " "                " "                " "
4  ( 1 ) "*"                 "*"                " "                " "                " "
5  ( 1 ) "*"                 "*"                " "                " "                "*"
6  ( 1 ) "*"                 "*"                " "                " "                "*"
7  ( 1 ) "*"                 "*"                " "                "*"                "*"
8  ( 1 ) "*"                 "*"                " "                "*"                "*"
9  ( 1 ) "*"                 "*"                "*"                "*"                "*"
         Performance.DMSMean Performance.AEDMean Performance.ADMSLMean Performance.Task
1  ( 1 ) " "                 " "                " "                "*"
2  ( 1 ) " "                 " "                " "                "*"
3  ( 1 ) " "                 " "                " "                "*"
4  ( 1 ) "*"                 " "                " "                "*"
5  ( 1 ) "*"                 " "                " "                "*"
6  ( 1 ) "*"                 "*"                " "                "*"
7  ( 1 ) "*"                 "*"                " "                "*"
8  ( 1 ) "*"                 "*"                "*"                "*"
9  ( 1 ) "*"                 "*"                "*"                "*"
```

Figura 5 - Resultados da função regsubsets para os dados normalizados e aleatórios

Assim, para as nove variáveis, o *regsubsets* apresenta quais as variáveis mais importantes, mediante o número de variáveis a considerar, ou seja, se o desejado for utilizar três variáveis, o *regsubsets* apresenta as três variáveis com mais impacto na aprendizagem. Neste caso, como temos nove variáveis, queremos saber a ordem de importâncias das nove variáveis. Destas nove variáveis, analisando o resultado do *regsubsets*, podemos definir uma ordem de importância mediante o número de variáveis desejado. De seguida, apresentam-se, para cada ponto, as variáveis ordenadas, decrescentemente, em termos de relevância.

- **1 variável:** Performance.Task;
- **2 variáveis:** Performance.Task, Performance.MAMean;
- **3 variáveis:** Performance.Task, Performance.MAMean, Performance.KDTMean;
- **4 variáveis:** Performance.Task, Performance.MAMean, Performance.KDTMean, Performance.DMSMean;
- **5 variáveis:** Performance.Task, Performance.MAMean, Performance.KDTMean, Performance.DMSMean, Performance.DDCMean ;
- **6 variáveis:** Performance.Task, Performance.MAMean, Performance.KDTMean, Performance.DMSMean, Performance.DDCMean, Performance.AEDMean;
- **7 variáveis:** Performance.Task, Performance.MAMean, Performance.KDTMean, Performance.DMSMean, Performance.DDCMean, Performance.AEDMean, Performance.TBCMean;
- **8 variáveis:** Performance.Task, Performance.MAMean, Performance.KDTMean, Performance.DMSMean, Performance.DDCMean, Performance.AEDMean, Performance.TBCMean, Performance.ADMSLMean;
- **9 variáveis:** conjunto de dados inteiro.

Mediante esta análise, na secção seguinte, utilizar-se-ão fórmulas diferentes para a aprendizagem da rede, que reflitam os resultados obtidos no *regsubsets.*

Assim sendo, a análise dos dados aleatórios com a tarefa e a exaustão normalizadas segue o mesmo registo apresentado para o ficheiro anterior, logo seria redundante repetir os passos mencionados.

Finalmente, apresenta-se o resultado da chamada da função *regsubsets* para os dos dados aleatórios com a tarefa e a exaustão normalizadas.

```
                    Forced in Forced out
Performance.KDTMean       FALSE      FALSE
Performance.MAMean        FALSE      FALSE
Performance.MVMean        FALSE      FALSE
Performance.TBCMean       FALSE      FALSE
Performance.DDCMean       FALSE      FALSE
Performance.DMSMean       FALSE      FALSE
Performance.AEDMean       FALSE      FALSE
Performance.ADMSLMean     FALSE      FALSE
Performance.Task          FALSE      FALSE
1 subsets of each size up to 9
Selection Algorithm: exhaustive
         Performance.KDTMean Performance.MAMean Performance.MVMean Performance.TBCMean Performance.DDCMean Performance.DMSMean Performance.AEDMean
1  ( 1 ) " "                 " "                " "                " "                " "                " "                " "
2  ( 1 ) " "                 " "                " "                " "                "*"                " "                " "
3  ( 1 ) " "                 "*"                "*"                " "                " "                " "                " "
4  ( 1 ) " "                 "*"                "*"                " "                "*"                " "                " "
5  ( 1 ) "*"                 "*"                "*"                " "                "*"                " "                " "
6  ( 1 ) " "                 "*"                "*"                " "                "*"                "*"                " "
7  ( 1 ) "*"                 "*"                "*"                " "                "*"                "*"                " "
8  ( 1 ) "*"                 "*"                "*"                "*"                "*"                "*"                " "
9  ( 1 ) "*"                 "*"                "*"                "*"                "*"                "*"                "*"
         Performance.ADMSLMean Performance.Task
1  ( 1 ) " "                   "*"
2  ( 1 ) " "                   "*"
3  ( 1 ) " "                   "*"
4  ( 1 ) " "                   "*"
5  ( 1 ) " "                   "*"
6  ( 1 ) "*"                   "*"
7  ( 1 ) "*"                   "*"
8  ( 1 ) "*"                   "*"
9  ( 1 ) "*"                   "*"
```

Figura 6 - Resultados da função regsubsets para os dados aleatórios com exaustão e tarefa normalizadas

iii.Análise dos Resultados com Fórmulas Distintas

1. Dados Normalizados e Aleatórios

Após a eliminação das representações de dados com erros e variações de resultados maiores e a verificação dos atributos mais significativos para a aprendizagem das redes, são, finalmente, realizados os testes com fórmulas com um número variado de parâmetros de entrada. Estes testes têm como objetivos a escolha da fórmula mais adequada para este tipo de dados, assim como o melhor volume de dados de aprendizagem e treino.

Deste modo, o grupo decidiu que o número mínimo de parâmetros na fórmula seria 4, fazendo, então, testes com os 4, 5, 6, 7, 8 e 9 parâmetros de entrada mais relevantes, mencionados na secção *ii*. Além disso, ainda se distinguem os resultados de redes com a função de ativação dos neurónios de *output* ativa ou não.

Relembra-se que a rede a ser testada tem três camadas interiores, com 60, 40 e 20 neurónios, respetivamente. A tolerância permitida (*threshold*) é 0.01, uma vez que se pretende a melhor aproximação possível. Também é possível distinguir dois tipos de volumes de dados de aprendizagem e treino diferentes, referentes ao 1º Cenário, que admite 600 registos para aprendizagem e 244 para treino, o que corresponde a cerca de 71% dos registos para aprendizagem e 29% para treino, e ao 2º Cenário, que contém 422 registos para aprendizagem e 422 para treino, ou seja, 50% dos dados são para aprendizagem e os outros 50% são para treino.

File	nº camadas	nº neurónios	nº parâmetros	Dados aprendizagem	Dados treino	thresh	L.O	error	rmse
1	3	60/40/20	9	600	244	0,01	F	0,095	0,3085
1	3	60/40/20	9	600	244	0,01	T	0,08	0,3599
1	3	60/40/20	9	422	422	0,01	F	0,048	0,3127
1	3	60/40/20	9	422	422	0,01	T	0,022	0,3424
1	3	60/40/20	4	600	244	0,01	F	0,905	0,2821
1	3	60/40/20	4	600	244	0,01	T	0,643	0,3215
1	3	60/40/20	4	422	422	0,01	F	0,378	0,2988
1	3	60/40/20	4	422	422	0,01	T	0,465	0,3097
1	3	60/40/20	5	600	244	0,01	F	0,278	0,3045
1	3	60/40/20	5	600	244	0,01	T	0,199	0,3401
1	3	60/40/20	5	422	422	0,01	F	0,155	0,2866
1	3	60/40/20	5	422	422	0,01	T	0,103	0,3297
1	3	60/40/20	6	600	244	0,01	F	0,218	0,3303
1	3	60/40/20	6	600	244	0,01	T	0,106	0,3266
1	3	60/40/20	6	422	422	0,01	F	0,085	0,318
1	3	60/40/20	6	422	422	0,01	T	0,065	0,3459
1	3	60/40/20	7	600	244	0,01	F	0,162	0,3177
1	3	60/40/20	7	600	244	0,01	T	0,102	0,3327
1	3	60/40/20	7	422	422	0,01	F	0,083	0,3012
1	3	60/40/20	7	422	422	0,01	T	0,068	0,3389
1	3	60/40/20	8	600	244	0,01	F	0,214	0,2962
1	3	60/40/20	8	600	244	0,01	T	0,107	0,3472
1	3	60/40/20	8	422	422	0,01	F	0,104	0,2948
1	3	60/40/20	8	422	422	0,01	T	0,063	0,3105

Figura 7 - Resultados dos testes para os dados normalizados e aleatórios

O melhor resultado obtido através dos vários testes, que se podem verificar na figura acima, está sublinhado a cor verde.

Desta forma, para os dados normalizados e aleatórios, usando uma rede com três camadas intermédias (com 60, 40, 20 neurónios, respetivamente) e *threshold* igual 0.01, o menor erro de aprendizagem que se encontra é de 0.022. O desvio padrão, por sua vez, toma o valor de 0.3424, ou seja, é aproximadamente 0.3, o que, analisando a figura, verifica-se que não difere dos restantes valores de *rmse*.

O volume de dados usado para este teste foi o de 50% dos registos para aprendizagem e 50% para treino. Além disso, o *linear.output* toma o valor TRUE, ou seja, a função de ativação não está ativa, logo são permitidas oscilações nos neurónios de *output*.

Procedendo, então, a uma análise dos dados obtidos, discrimina-se para as várias fórmulas utilizadas, o volume de dados usado e qual a topologia que obteve melhores resultados.

Para a fórmula com nove parâmetros: com o volume de dados do 1º Cenário obteve-se melhores resultados usando a rede com a função de ativação não ativa, com o volume de dados do 2º Cenário obteve-se melhores resultados usando a rede com função de ativação não ativa (que é o resultado ótimo).

Para as fórmulas com 5, 6, 7 e 8 parâmetros: com o volume de dados do 1º Cenário obteve-se melhores resultados usando a rede com a função de ativação não ativa, com o volume de dados do 2º Cenário obteve-se melhores resultados usando a rede com função de ativação não ativa.

Por outro lado, para a fórmula com 4 parâmetros: com o volume de dados do 1º Cenário obteve-se melhores resultados usando a rede com a função de ativação não ativa, com o volume de dados do 2º Cenário obteve-se melhores resultados usando a rede com função de ativação ativa.

Desta forma, pode-se concluir que, em geral, para as redes e volumes de dados definidos obtém-se melhores resultados com a função de ativação não ativa (*linear.output=TRUE)*. Também se verifica que para a mesma fórmula é melhor que a função de ativação não esteja ativa. Além disso, verificam-se melhores resultados de erro e desvio padrão quando se utiliza o volume de dados de aprendizagem e treino do 2º Cenário (50%/50%).

Analisando o crescente número de parâmetros na fórmula, verifica-se que o erro de aprendizagem diminui com o aumento de entradas na fórmula, uma vez que a fórmula possui, assim, mais dados para processar e aprender melhor a fazer a previsão dos resultados. O valor do desvio padrão mantém-se constante, permanecendo no valor 0.3.

2. Dados aleatórios com Exaustão e Tarefa Normalizados

Mediante os resultados obtidos na secção anterior, foi possível determinar diferentes tipos de fórmulas a serem utilizadas na aprendizagem da rede. A seguinte imagem apresenta os diferentes parâmetros, mediante o número de variáveis desejado.

1	TASK							
2	TASK	DDC						
3	TASK	DDC	MA					
4	TASK	DDC	MA	MV				
5	TASK	DDC	MA	MV	KDT			
6	TASK	DDC	MA	MV	DMS	ADMSL		
7	TASK	DDC	MA	MV	DMS	ADMSL	KDT	
8	TASK	DDC	MA	MV	DMS	ADMSL	KDT	TBC
9	TASK	DDC	MA	MV	DMS	ADMSL	KDT	TBC	AED

Figura 8 - Diferentes parâmetros a serem utilizados nos dados aleatórios e com exaustão e tarefa normalizados

As linhas coloridas apresentam os diferentes parâmetros a serem considerados na aprendizagem. Cada cor originará uma nova fórmula a ser testada, por exemplo, se quisermos considerar 4 parâmetros, a fórmula resultante será:

```
FatigueLevel~Performance.Task+Performance.DDCMean+Performance.MAMean+Perf
ormance.MVMean
```

De seguida, apresentam-se os resultados da aprendizagem da rede com variações ao nível da fórmula, da *flag linear output*, dos dados de aprendizagem e de treino.

File	nº camadas	nºnodos	nº parâmetros	Dados aprendizagem	Dados treino	threshold	L.O.	error	rmse
2	3	60/40/20	9	600	244	0.01	F	0.07513	0.3377
2	3	60/40/20	9	600	244	0.01	T	0.04168	0.3377
2	3	60/40/20	9	422	422	0.01	F	0.0526	0.3012
2	3	60/40/20	9	422	422	0.01	T	0.01718	0.3786
2	3	60/40/20	4	600	244	0.01	F	0.3893	0.3111
2	3	60/40/20	4	600	244	0.01	T	0.5021	0.3437
2	3	60/40/20	4	422	422	0.01	F	0.355	0.31124
2	3	60/40/20	4	422	422	0.01	T	0.05107	0.4394
2	3	60/40/20	5	600	244	0.01	F	0.2039	0.3072
2	3	60/40/20	5	600	244	0.01	T	0.1417	0.3788
2	3	60/40/20	5	422	422	0.01	F	0.09559	0.3105
2	3	60/40/20	5	422	422	0.01	T	0.02441	0.3499
2	3	60/40/20	6	600	244	0.01	F	0.12734	0.3084
2	3	60/40/20	6	600	244	0.01	T	0.08845	0.3884
2	3	60/40/20	6	422	422	0.01	F	0.1138	0.318
2	3	60/40/20	6	422	422	0.01	T	0.09961	0.3703
2	3	60/40/20	7	600	244	0.01	F	0.1587	0.3124
2	3	60/40/20	7	600	244	0.01	T	0.04888	0.3978
2	3	60/40/20	7	422	422	0.01	F	0.06436	0.3282
2	3	60/40/20	7	422	422	0.01	T	0.01302	0.3499
2	3	60/40/20	8	600	244	0.01	F	0.08276	0.3303
2	3	60/40/20	8	600	244	0.01	T	0.04399	0.3745
2	3	60/40/20	8	422	422	0.01	F	0.02892	0.3074
2	3	60/40/20	8	422	422	0.01	T	0.01012	0.3472

Figura 9 - Resultados da aprendizagem da rede para os dados aleatórios e tarefa e exaustão normalizados

A rede mencionada neste caso apresenta três camadas intermédias, tendo a primeira 60 neurónios, a segunda 40 neurónios e a terceira 20 neurónios. O valor de *threshold* é constante, sendo ele 0.01. As variações ocorrem ao nível dos dados, das fórmulas e do *linear output*. Relativamente aos dados, estes foram utilizados de duas maneiras diferentes. A primeira implica 422 dados para aprendizagem e 422 para treino. A segunda divide os dados de modo a que 622 linhas sejam para aprendizagem e 244 para treino. Em relação ao *linear output*, esta *flag* pode tomar o valor T(true), para quando queremos permitir oscilações muito grandes na função de ativação, ou F(False), quando desejamos o contrário.

Geralmente, quando o valor de *linear output* é falso, os erros de aprendizagem são mais elevados do que quando o valor de *linear output* é verdadeiro e isto verifica-se, visto que, como os dados não estão organizados uniformemente, os dados apresentam grandes oscilações e, se não é permitido à função de ativação acompanhar essas oscilações, o provável é que o erro de aprendizagem aumente.

Começando com quatro parâmetros, à medida que se avança para uma fórmula que considera mais parâmetros, o erro de aprendizagem diminui, tanto como o desvio padrão (apesar da diminuição não ser tão óbvia neste caso). O erro diminui, visto que estamos a considerar mais parâmetros que afetam positivamente a aprendizagem da rede.

Para a primeira maneira de representar os dados (600 dados para aprendizagem), quando o valor de *linear output* é *false*, os resultados são melhores (mais baixos), do que quando valor de *linear output* é *true*. O contrário acontece para a segunda maneira de representar dados (422 dados para aprendizagem). A razão de isto acontecer, deve-se ao facto da organização dos dados. Os dados estão organizados de tal maneira que os 600 dados não apresentam qualquer distribuição uniforme (daí os melhores resultados surgirem quando são permitidas oscilações na função de ativação), enquanto que os 422

primeiros dados já apresentam melhor distribuição, pois, quando não são permitidas oscilações na função de ativação, os resultados são melhores.

A organização dos dados apresenta valores que representam melhor o todo nos primeiros 422 dados, pois, para a mesma fórmula, o valor do erro é mais baixo.

No entanto, existe uma exceção, sendo o valor ótimo para a rede. A regra geral afirmava que à medida que se aumenta o número de parâmetros da fórmula, tanto o erro como o desvio padrão apresentam resultados inferiores. No entanto, quando utilizamos 8 parâmetros (não considerando Performance.AEDMean), tanto o valor do erro de aprendizagem, como o desvio padrão é superior quando se utilizam mais parâmetros, o que nos leva a concluir que o parâmetro Performance.AEDMean não influencia positivamente a aprendizagem da rede. Os melhores valores para a rede estão representados na tabela superior com uma linha verde.

iv.Testes com Topologias Diferentes

1. Dados Normalizados e Aleatórios

Escolhida a fórmula e volume de dados mais adequados, procedeu-se à realização de testes com redes com um número diferente de camadas intermédias e neurónios. Deste modo, como mencionado na Metodologia, foram utilizadas mais quatro redes além da que foi utilizada na secção anterior.

A primeira rede corresponde ao valor ótimo obtido no teste anterior. A segunda e terceira redes, por sua vez, possuem três camadas interiores. A segunda possui 120 neurónios na primeira camada, 80 na segunda e 40 na terceira, ou seja, o mesmo número de camadas que a primeira rede, porém com o dobro do número de neurónios. A terceira, no entanto, possui apenas 6 neurónios na primeira camada, 4 na segunda e 2 na terceira, de forma a exemplificar uma rede com três camadas, mas um número reduzido de neurónios.

O número de camadas interiores é alterado na quarta rede, que possui duas camadas intermédias, com 10 e 5 neurónios, respetivamente. A quinta rede possui apenas uma camada interior com 20 neurónios. Todos os outros parâmetros são iguais aos do resultado ótimo da secção anterior para este tipo de dados.

1	3	60/40/20	9	422	422	0,01	T	0,048	0,3127
1	3	120/80/40	9	422	422	0,01	T	0,016	0,3172
1	3	6/4/2	9	422	422	0,01	T	1,428	0,2783
1	2	10/5	9	422	422	0,01	T	0,936	0,2722
1	1	20	9	422	422	0,01	T	0,59	0,3051

Figura 10 - Resultados dos testes com camadas intermédias diferentes para os dados normalizados e aleatórios

Analisando, então, os resultados obtidos, verifica-se que para a segunda rede o erro de aprendizagem decresce para mais de metade, porém o desvio padrão não sofre a mesma redução, uma vez que aumenta 0.0045. Conclui-se, então, que o uso de uma rede com três camadas e o dobro de neurónios não promove um melhoramento significativo dos resultados. No entanto, para os testes com diferentes escalas usar-se-ão estas duas redes.

Para a terceira rede, que possui um número reduzido de neurónios, o erro de aprendizagem é muito elevado, ficando acima de uma unidade, logo esta rede é rejeitada de seguida.

Para a quarta rede (duas camadas, com 10 e 5 neurónios, respetivamente), verifica-se que o erro se situa a pouco de uma unidade (rondando os 0.9), logo regista-se um valor muito elevado também, o que leva à rejeição desta rede para aprendizagem.

Por fim, a quinta rede (uma camada com 20 neurónios) verifica um erro que ronda os 0.5, logo este valor continua a ser excessivo, sendo esta rede também rejeitada.

2. Dados aleatórios com Tarefa e Exaustão normalizadas

Após terem sido testados diversos parâmetros e diferentes volumes de dados, foi descoberta uma tipologia de rede com bons resultados de erro de aprendizagem e desvio padrão. Nesta secção, a única alteração à rede anterior passa pela mudança do número de camadas intermédias. A topologia anterior possuía três camadas intermédias com 60 neurónios na primeira camada, 40 na segunda e 20 na última. Para efetuar novos testes, utilizaremos 4 novos tipos de arquitetura de rede, sendo eles:

- **3 camadas intermédias:** 120 neurónios na primeira, 80 neurónios na segunda e 40 neurónios na terceira camada;
- **3 camadas intermédias:** 6 neurónios na primeira, 4 neurónios na segunda e 2 neurónios na terceira camada;
- **2 camadas intermédias:** 10 neurónios na primeira e 5 neurónios na segunda camada intermédia;
- **1 camada intermédia:** 20 neurónios nesta camada.

De seguida, apresentam-se os resultados para as diferentes camadas referidas anteriormente.

File	nº camadas	nºnodos	nº entrada	Dados aprendizagem	Dados treino	threshold	L.O.	error	rmse
2	3	60/40/20	8	422	422	0.01	T	0.01012	0.3472
2	3	120/80/40	8	422	422	0.01	T	0.02903	0.4786
2	3	6/4/2	8	422	422	0.01	T	1.5723	0.2687
2	2	10/5	8	422	422	0.01	T	1.0901	0.2783
2	1	20	8	422	422	0.01	T	1.0812	0.2923

Figura 11 - Resultados da alteração das topologias de redes com mudança das camadas intermédias

Como seria expectável, as últimas três topologias representadas apresentam um valor de erro de aprendizagem muito superior às outras, pois possuem poucos neurónios para processar os dados recebidos como *input*. Apesar do valor do desvio padrão ser inferior (à primeira camada representada a verde), a diferença é pequena e torna-se desnecessário considerar esta topologia, visto que o melhor valor do erro é de 0.01, cem vezes menor que o erro de aprendizagem das últimas três topologias.

Relativamente à rede que possui 3 camadas intermédias (120/80/40), o erro de aprendizagem aumentou para (aproximadamente) o triplo, tal como o desvio padrão que também aumentou substancialmente.

Assim sendo, mesmo alterando o número de camadas intermédias e o número de neurónios nelas contidas, a melhor topologia de rede continua a ser a descoberta na secção anterior. Verificou-se ainda que, mesmo duplicando o número de neurónios, a capacidade de aprendizagem não aumenta para o dobro. Neste caso, a capacidade de aprendizagem piorou quando se multiplicou o número de neurónios da rede ideal.

V. Identificação de Existência ou Inexistência de Exaustão

1. Dados Normalizados e Aleatórios

Para a identificação de existência ou inexistência de exaustão foram usadas duas redes com três camadas intermédias (60, 40, 20 e 120, 80, 40) e variou-se o valor booleano do *linear.output*.

1	3	60/40/20	9	422	422	0,01	T	0.04239	0.0424
1	3	120/80/40	9	422	422	0,01	T	0.0813	0.0808
1	3	60/40/20	9	422	422	0,01	F	0,3226	0.0424
1	3	120/80/40	9	422	422	0,01	F	0.39905	0.04243

Figura 12 - Resultados dos testes da identificação de exaustão para os dados normalizados e aleatórios

Analisando os valores obtidos, verifica-se que quando o *linear.output* é falso, ou seja, não são permitidas oscilações nos neurónios de *output*, o erro de aprendizagem é mais alto (cerca de 0.3) do que quando a função de ativação não é ativa (neste caso, atinge-se valores de erro de aprendizagem de cerca de 0.04 e 0.08). Os valores do desvio padrão são extremamente baixos para ambos os casos, não ultrapassando os 0.08.

2. Dados aleatórios e a exaustão e a tarefa randomizados

Para determinar a existência de exaustão, utilizando a estratégia descrita na apresentação das escalas, e com a rede neuronal obtida na secção anterior, foi possível obter os seguintes valores de erro de aprendizagem e desvio padrão.

File	nº camadas	nºnodos	nº parâmetros	Dados aprendizagem	Dados treino	threshold	L.O.	error	rmse
2	3	60/40/20	6	422	244	0.01	F	0.3233	0.0424
2	3	60/40/20	6	422	244	0.01	T	0.2893	0.0424
2	3	120/80/40	6	422	244	0.01	F	0.4	0.04244
2	3	120/80/40	6	422	244	0.01	T	0.2769	0.0646

Figura 13-Resultados dos testes da identificação de exaustão para os dados aleatórios e a exaustão e a tarefa randomizados

Analisando os valores obtidos, verificamos que para quando o *linear output* apresenta valores *true*, o erro de aprendizagem é inferior a quando o *linear output* apresenta o valor *false.* Também conseguimos analisar que quando a rede apresenta a seguinte composição intermédia 120/80/40, os resultados são mais elevados, do que a rede que apresenta 60/40/20 neurónios.

Assim sendo, a melhor solução encontrada acontece para a rede que tem 60/40/20 neurónios e o linear output é *true*, pois apresenta melhores resultados. Apesar do erro de aprendizagem da rede de 120/80/40 neurónios e linear output *true* ser ligeiramente menor, o seu desvio padrão é muito superior, o que não a torna a melhor rede.

vi.Testes com Escala de 3 Níveis de Exaustão

1. Dados Normalizados e Aleatórios

Os testes da escala de 3 níveis de exaustão foram feitos com redes com três camadas intermédias (60, 40, 20 e 120, 80, 40) e variou-se o valor booleano do *linear.output*.

1	3	60/40/20	9	422	422	0,01	T	0.1513	0.0651
1	3	120/80/40	9	422	422	0,01	T	0.01992	0,113
1	3	60/40/20	9	422	422	0,01	F	0.3644	0.04326
1	3	120/80/40	9	422	422	0,01	F	0.44454	0.04326

Figura 14 - Resultados dos testes da escala de 3 níveis para os dados normalizados e aleatórios

Analisando os resultados obtidos, verifica-se que novamente o erro de aprendizagem é menor para as redes com *linear.output = FALSE*, uma vez que neste caso atinge valores de 0.019 quando o número de neurónios é 120, 80, 40. Por outro lado, quando a função de ativação é ativa, o erro ronda os 0.3 e 0.4, que é significativamente mais elevado. Os valores de *rmse* também se mantêm relativamente baixos, sendo que para o caso do erro de 0.019, o *rmse* é de 0.113.

Conclui-se, então, que o melhor resultado é atingido com a rede com camadas intermédias de (120, 80, 40) e *linear.output*=TRUE.

2. Dados aleatórios e a exaustão e a tarefa randomizados

Para determinar a melhor escala representativa de exaustão, utilizando a estratégia descrita na apresentação das escalas, na parte da escala com três níveis, e com a rede neuronal obtida na secção anterior, foi possível obter os seguintes valores de erro de aprendizagem e desvio padrão.

File	nº camadas	nºnodos	nº parâmetros	Dados aprendizagem	Dados treino	threshold	L.O.	error	rmse
2	3	60/40/20	6	422	244	0.01	F	0.3554	0.04327
2	3	60/40/20	6	422	244	0.01	T	0.309	0.04327
2	3	120/80/40	6	422	244	0.01	F	0.445	0.04327
2	3	120/80/40	6	422	244	0.01	T	0.2879	0.04327

Figura 15-Resultados dos testes da identificação da melhor escala de 3 níveis para os dados aleatórios e a exaustão e a tarefa randomizados

Analisando os resultados, verificamos que para quando o *linear output* apresenta valores *true*, o erro de aprendizagem é inferior a quando o *linear output* apresenta o valor *false.* Também conseguimos analisar que quando a rede apresenta a seguinte composição intermédia 120/80/40, os resultados são mais elevados, do que a rede que apresenta 60/40/20 neurónios, independentemente do valor do *linear output*.

Assim sendo, a melhor solução encontrada acontece para a rede que tem 120/80/40 neurónios e o linear output é *true*, pois apresenta claramente os melhores

resultados, visto que todos partilham o mesmo valor de desvio padrão e o erro desta rede é o mais baixo.

vii.Testes com Escalas de 4 Níveis de Exaustão

1. Dados Normalizados e Aleatórios

Os testes da escala de 4 níveis de exaustão foram feitos com redes com três camadas intermédias (60, 40, 20 e 120, 80, 40) e variou-se o valor booleano do *linear.output*.

1	3	60/40/20	9	422	422	0,01	T	0.03774	0.1249
1	3	120/80/40	9	422	422	0,01	T	0.03965	0.1763
1	3	60/40/20	9	422	422	0,01	F	0.2851	0.1145
1	3	120/80/40	9	422	422	0,01	F	2.9779	0.1144

Figura 16 - Resultados dos testes da escala de 4 níveis para dados normalizados e aleatórios

Analisando os resultados dos testes realizados, verifica-se que mais uma vez o erro de aprendizagem é significativamente menor para quando a função de ativação está desativada (rondando os 0.03), do que para quando não são permitidas oscilações nos neurónios. Neste caso, o valor do desvio padrão é constante para todos os casos (nunca variando além dos 0.1).

Conclui-se, então, que o melhor resultado é obtido quando a rede tem camada interiores (60, 40, 20) e *linear.output*=TRUE.

2. Dados aleatórios e a exaustão e a tarefa randomizados

Para determinar a melhor escala representativa de exaustão, utilizando uma escala com quatro níveis e com a rede neuronal obtida na secção anterior, foi possível obter os seguintes valores de erro de aprendizagem e desvio padrão.

File	nº camadas	nºnodos	nº parâmetros	Dados aprendizagem	Dados treino	threshold	L.O.	error	rmse
2	3	60/40/20	6	422	244	0.01	F	0.6286	0.1144
2	3	60/40/20	6	422	244	0.01	T	0.02363	0.22
2	3	120/80/40	6	422	244	0.01	F	2.98	0.1144
2	3	120/80/40	6	422	244	0.01	T	0.01691	0.3211

Figura 17-Figura 5-Resultados dos testes da identificação da melhor escala de 4 níveis para os dados aleatórios e a exaustão e a tarefa randomizados

Analisando os resultados, verificamos que para quando o *linear output* apresenta valores *true*, o erro de aprendizagem é bastante inferior a quando o *linear output* apresenta o valor *false.* Também conseguimos analisar que não ocorreu uma correlação entre o erro de aprendizagem e o número de neurónios, pois o mesmo linear output um erro de aprendizagem aumenta, quando ocorre o aumento dos neurónios, enquanto que existe outro caso em que o contrário acontece.

Assim sendo, a melhor solução encontrada acontece para a rede que tem 120/80/40 neurónios e o linear output é *true*, pois apresenta o menor erro de aprendizagem. Apesar da rede com 60/40/20 neurónios apresentar um desvio padrão

inferior, não é o suficiente para essa rede ser escolhida, visto que o erro de aprendizagem é o dobro da ideal.

viii.Identificação dos Sete Níveis de Exaustão

Os dados fornecidos pela equipa docente mostram uma variedade muito escassa de níveis de exaustão diferentes. Por exemplo, os primeiros dois níveis (nível 1 e nível 2) ocupam mais de 50% do total de resultados, enquanto que o nível cinco é encontrado em 26 registos e o nível seis apenas é encontrado em 4 em registos. Não há registos do sétimo nível de exaustão.

Deste modo, o grupo decidiu dividir a gama de seis níveis fornecidas, em sete níveis distintos. Assim, procedeu à soma dos valores de todos os atributos (excetuando o nível de fatiga) de cada registo. A partir daí, foram calculados os valores máximo e mínimo desse conjunto de valores. O valor máximo identificado é oito e o mínimo é 1.4, logo calculou-se a diferença entres estes valores e, de seguida, dividiu-se o resultado por 7 (níveis).

Assim, identificou-se cada um dos intervalos obtidos por um nível de exaustão, sendo que:

[1.4, 2.34] -> Nível 7
[2.34, 3.28] -> Nível 6
[3.28, 4.22] -> Nível 5
[4.22, 5.16] -> Nível 4
[5.16, 6.1] -> Nível 3
[6.1, 7.04] -> Nível 2
[7.04, 8] -> Nível 1

Deste modo, o grupo conseguiu identificar sete níveis diferentes de exaustão agrupados entre valores.

Utilizando a representação de dados em que estes estão normalizados e aleatórios, testou-se a nova escala com as redes de três camadas (60, 40, 20 e 120, 80, 40), a fórmula completa e alternando os valores *linear.output*, obtiveram-se os seguintes resultados:

1	3	60/40/20	9	422	422	0,01	T	0.12044	0.3879
1	3	120/80/40	9	422	422	0,01	T	0.01996	0.3934
1	3	60/40/20	9	422	422	0,01	F	0.1564	0.3879
1	3	120/80/40	9	422	422	0,01	F	0.12342	0.3879

Figura 18 - Resultados dos testes da escala de 7 níveis para dados normalizados e aleatórios

Analisando os resultados, pode-se verificar que o erro de aprendizagem é superior quando os neurónios de *output* não podem sofrer oscilações muito significativas. Quando a função de ativação não está ativa, por outro lado, verifica-se que o erro, especialmente na rede (120, 80, 40) é muito reduzido. O valor *rmse* permanece constante.

Deste modo, conclui-se que o melhor resultado é obtido com a rede de três camadas (120, 80, 40) quando a função de ativação se encontra desativada.

Por outro lado, se a fórmula utilizada contiver apenas quatro parâmetros, sendo eles a DMSMean, a TBCMean, a DDCMean e a KDTMean, obtém-se os seguintes resultados:

1	3	60/40/20	9	422	422	0,01	T	0.11562	0.4190
1	3	120/80/40	9	422	422	0,01	T	0.03692	0.4527
1	3	60/40/20	9	422	422	0,01	F	0.107	0.4105
1	3	120/80/40	9	422	422	0,01	F	0.07959	0.4167

Figura 19 - Resultados dos testes com a escala de 7 níveis para os dados normalizados e aleatórios

Analisando os resultados obtidos, existe um maior equilíbrio entre os valores de erro de aprendizagem obtidos para os dois valores de *linear.output*, uma vez que não é claro que o valor FALSE é pior do que o valor TRUE, ou vice-versa. Assim, dentro dos dois testes para o valor TRUE, o melhor resultado obteve-se quando se usou a rede neuronal (120, 80, 40), pois ronda os 0.03. Por outro lado, quando o valor é FALSE, obteve-se o melhor resultado quando também se usou a rede neuronal (120, 80, 40), com um erro de 0.08. O valor de *rmse* permaneceu praticamente constante com estes valores.

Escolhendo o melhor resultado de entre os dois acima mencionados, identificamos que a função de ativação desativada resulta em menores erros.

ix.Reconhecimento da tarefa em execução

Um dos objetivos propostos no enunciado do trabalho prático era identificar a tarefa em execução, mediantes os dados recebidos. Para materializar isto é necessário definir uma fórmula que estime o valor da Performance.Task, sendo ela:

```
Performance.Task ~
Performance.KDTMean+Performance.MAMean+Performance.MVMean+Performance.TBC
Mean+Performance.DDCMean+Performance.DMSMean+Performance.AEDMean+Performa
nce.ADMSLMean+FatigueLevel
```

De seguida, apresentam-se os resultados da aprendizagem de várias redes diferentes, em que os valores que são alterados são os dados de aprendizagem e de treino e o *linear output*.

File	nº camadas	nºnodos	nº parâmetros	Dados aprendizagem	Dados treino	threshold	L.O.	error	rmse
3	3	60/40/20	9	600	244	0.01	F	0.05157	0.2364
3	3	60/40/20	9	600	244	0.01	T	0.04541	0.2221
3	3	60/40/20	9	422	422	0.01	F	0.06226	0.2258
3	3	60/40/20	9	422	422	0.01	T	0.01885	0.233

Figura 20 - Resultados da aprendizagem da Task

Analisando os valores obtidos, podemos desde logo concluir que todas as redes apresentaram um erro de aprendizagem bastante baixo. Para os mesmos dados de aprendizagem e treino, alternando só o *linear output*, quando o valor do *linear output* é *true*, tanto o erro de aprendizagem, como o desvio padrão são inferiores. Para o mesmo valor do *linear output* não podemos tirar conclusões em concreto, pois para um determinado tipo de dados o erro é mais baixo quando o *linear output* é true, mas para o outro tipo de dados o erro é mais baixo quando o valor do *linear output* é *false.*

A melhor rede para a aprendizagem do reconhecimento da tarefa em execução é a representada na última linha da tabela, ou seja, quando existem 422 dados tanto para aprendizagem como para treino e quando o valor de *linear output* é igual a *true.* O erro de aprendizagem desta rede é o mais baixo, no entanto o desvio padrão não o é. Apesar de

existirem redes com desvio padrão mais baixo, a diferença não compensa, pois circula por volta das centésimas, enquanto que a diferença do erro de aprendizagem varia nas décimas.

3. Conclusões

Terminado o 3º trabalho prático, o grupo avalia o seu desempenho como bastante positivo. O grupo conseguiu atingir todos os objetivos propostos no enunciado do trabalho e acredita tê-los atingido com distinção.

Relativamente à análise das redes neuronais, o grupo fez questão de testar as mais variadas hipóteses e combinações, bem como diversos tipos de ficheiros, analisando todos casos ao pormenor e documentando todo o processo.

Além disto, o grupo conseguiu obter uma resposta para o problema da identificação através dos 7 níveis de exaustão, quando nos dados só estavam representados 6 níveis, utilizando um tipo de aprendizagem não supervisionada.

Outro ponto positivo consistiu no aumento do conhecimento da linguagem R, linguagem desconhecida ao grupo, antes da UC correspondente.

Em suma, com a análise profunda de cada rede testada foram analisados os resultados obtidos e identificados os melhores resultados, tendo o grupo alargado o seu conhecimento sobre redes neuronais artificiais.

Bibliografia

Referências Bibliográficas

[Analide, 2011] ANALIDE, César, NOVAIS, Paulo, NEVES, José,
"Sugestões para a Elaboração de Relatórios", Relatório Técnico, Departamento de Informática, Universidade do Minho, Portugal, 2011.

Script em R

```
# Import das bibliotecas necessárias
library("neuralnet")
library("hydroGOF")
library("arules")

# Leitura do ficheiro de dados
dataset <- read.csv("C:\\Users\\perei\\Desktop\\exaustao-norm-rand.csv",
header=TRUE, sep=";", dec=".")

# Discretização dos valores dos atributos  -- Para o relatório
#dataset$Performance.KDTMean <-
discretize(dataset$Performance.KDTMean,method="frequency", categories=10)
#dataset$Performance.MAMean <-
discretize(dataset$Performance.MAMean,method="frequency", categories=10)
#dataset$Performance.MVMean <-
discretize(dataset$Performance.MVMean,method="frequency", categories=10)
#dataset$Performance.TBCMean <-
discretize(dataset$Performance.TBCMean,method="frequency", categories=10)
#dataset$Performance.DDCMean <-
discretize(dataset$Performance.DDCMean,method="frequency", categories=10)
#dataset$Performance.ADMSLMean <-
discretize(dataset$Performance.ADMSLMean,method="frequency",
categories=10)
#dataset$Performance.DMSMean <-
discretize(dataset$Performance.DMSMean,method="frequency", categories=10)
#dataset$Performance.AEDMean <-
discretize(dataset$Performance.AEDMean,method="frequency", categories=10)
#write.csv(dataset, "exaustao-discretizada.csv")

# Divisão dos dados para aprendizagem em vários conjuntos com dimensões
distintas
dados1 <- dataset[1:600,] # 633 registos
dados2 <- dataset[1:422,] # 422 registos
dados3 <- dataset[1:211,] # 211 registos
dados4 <- dataset[1:140,] # 140 registos

# Divisão dos dados para aprendizagem em vários conjuntos com dimensões
distintas
treino1 <- dataset[601:844,] # 211 registos
treino2 <- dataset[423:844,] # 422 registos
treino3 <- dataset[212:844,] # 634 registos
treino4 <- dataset[141:844,] # 704 registos


sub41 <-subset(treino1,
select=c("Performance.Task","Performance.DDCMean","Performance.MAMean","P
erformance.MVMean","FatigueLevel"))
sub42 <-subset(treino2,
select=c("Performance.Task","Performance.DDCMean","Performance.MAMean","P
erformance.MVMean","FatigueLevel"))

sub51 <-subset(treino1,
select=c("Performance.Task","Performance.DDCMean","Performance.MAMean","P
erformance.MVMean","Performance.KDTMean","FatigueLevel"))
```

```
sub52 <-subset(treino2,
select=c("Performance.Task","Performance.DDCMean","Performance.MAMean","P
erformance.MVMean","Performance.KDTMean","FatigueLevel"))

sub61 <-subset(treino1,
select=c("Performance.Task","Performance.DDCMean","Performance.MAMean","P
erformance.MVMean","Performance.DMSMean","Performance.ADMSLMean","Fatigue
Level"))
sub62 <-subset(treino2,
select=c("Performance.Task","Performance.DDCMean","Performance.MAMean","P
erformance.MVMean","Performance.DMSMean","Performance.ADMSLMean","Fatigue
Level"))

sub71 <-subset(treino1,
select=c("Performance.Task","Performance.DDCMean","Performance.MAMean","P
erformance.MVMean","Performance.DMSMean","Performance.ADMSLMean","Perform
ance.KDTMean","FatigueLevel"))
sub72 <-subset(treino2,
select=c("Performance.Task","Performance.DDCMean","Performance.MAMean","P
erformance.MVMean","Performance.DMSMean","Performance.ADMSLMean","Perform
ance.KDTMean","FatigueLevel"))

sub81 <-subset(treino1,
select=c("Performance.Task","Performance.DDCMean","Performance.MAMean","P
erformance.MVMean","Performance.DMSMean","Performance.ADMSLMean","Perform
ance.KDTMean","Performance.TBCMean","FatigueLevel"))
sub82 <-subset(treino2,
select=c("Performance.Task","Performance.DDCMean","Performance.MAMean","P
erformance.MVMean","Performance.DMSMean","Performance.ADMSLMean","Perform
ance.KDTMean","Performance.TBCMean","FatigueLevel"))

# Diferentes camadas intermédias
hidden1 <- c(60, 40, 20)
hidden2 <- c(120, 80, 40)
hidden3 <- c(6, 4, 2)
hidden4 <- c(10,5)
hidden5 <- c(20)

# Diferentes tolerâncias (thresholds)
threshold1 <- 0.01
threshold2 <- 0.1

# Diferentes fórmulas
formula1 <- FatigueLevel ~
Performance.KDTMean+Performance.MAMean+Performance.MVMean+Performance.TBC
Mean+Performance.DDCMean+Performance.DMSMean+Performance.AEDMean+Performa
nce.ADMSLMean+Performance.Task
formula4NNR <- FatigueLevel ~
Performance.Task+Performance.DDCMean+Performance.MAMean+Performance.MVMea
n
formula5NNR <- FatigueLevel ~
Performance.Task+Performance.DDCMean+Performance.MAMean+Performance.MVMea
n+Performance.KDTMean
formula6NNR <- FatigueLevel ~
Performance.Task+Performance.DDCMean+Performance.MAMean+Performance.MVMea
n+Performance.DMSMean+Performance.ADMSLMean
formula7NNR <- FatigueLevel ~
Performance.Task+Performance.DDCMean+Performance.MAMean+Performance.MVMea
n+Performance.DMSMean+Performance.ADMSLMean+Performance.KDTMean
```

```
formula8NNR <- FatigueLevel ~
Performance.Task+Performance.DDCMean+Performance.MAMean+Performance.MVMea
n+Performance.DMSMean+Performance.ADMSLMean+Performance.KDTMean+Performan
ce.TBCMean

formulaTask <- Performance.Task ~
Performance.KDTMean+Performance.MAMean+Performance.MVMean+Performance.TBC
Mean+Performance.DDCMean+Performance.DMSMean+Performance.AEDMean+Performa
nce.ADMSLMean+FatigueLevel

formula2NR <- FatigueLevel ~
Performance.Task+Performance.MAMean+Performance.KDTMean+Performance.DMSMe
an
formula3NR <- FatigueLevel ~
Performance.Task+Performance.MAMean+Performance.KDTMean+Performance.DMSMe
an+Performance.DDCMean
formula4NR <- FatigueLevel ~
Performance.Task+Performance.MAMean+Performance.KDTMean+Performance.DMSMe
an+Performance.DDCMean+Performance.AEDMean
formula5NR <- FatigueLevel ~
Performance.Task+Performance.MAMean+Performance.KDTMean+Performance.DMSMe
an+Performance.DDCMean+Performance.AEDMean+Performance.TBCMean
formula6NR <- FatigueLevel ~
Performance.Task+Performance.MAMean+Performance.KDTMean+Performance.DMSMe
an+Performance.DDCMean+Performance.AEDMean+Performance.TBCMean+Performanc
e.ADMSLMean

formula7NR <- FatigueLevel ~
Performance.Task+Performance.DDCMean+Performance.MAMean+Performance.MVMea
n
formula8NR <- FatigueLevel ~
Performance.Task+Performance.DDCMean+Performance.MAMean+Performance.MVMea
n+Performance.KDTMean
formula9NR <- FatigueLevel ~
Performance.Task+Performance.DDCMean+Performance.MAMean+Performance.MVMea
n+Performance.DMSMean+Performance.ADMSLMean
formula10NR <- FatigueLevel ~
Performace.Task+Performance.DDCMean+Performance.MAMean+Performance.MVMean
+Performance.KDTMean+Performance.DMSMean+Performance.ADMSLMean
formula11NR <- FatigueLevel ~
Performace.Task+Performance.DDCMean+Performance.MAMean+Performance.MVMean
+Perfomance.TBCMean+Performance.KDTMean+Performance.ADMSLMean+Performance
.DMSMean

sub71NR <- subset(treino1,
select=c("Performance.Task","Performance.MAMean","Performance.DDCMean","P
erformance.MVMean","FatigueLevel"))
sub72NR <- subset(treino2,
select=c("Performance.Task","Performance.MAMean","Performance.DDCMean","P
erformance.MVMean","FatigueLevel"))
sub81NR <- subset(treino1,
select=c("Performance.Task","Performance.MAMean","Performance.DDCMean","P
erformance.MVMean","Performance.KDTMean","FatigueLevel"))
sub82NR <- subset(treino2,
select=c("Performance.Task","Performance.MAMean","Performance.DDCMean","P
erformance.MVMean","Performance.KDTMean","FatigueLevel"))
sub91NR <- subset(treino1,
select=c("Performance.Task","Performance.MAMean","Performance.DDCMean","P
erformance.MVMean","Performance.DMSMean","Performance.ADMSLMean","Fatigue
Level"))
```

```
sub92NR <- subset(treino2,
select=c("Performance.Task","Performance.MAMean","Performance.DDCMean","P
erformance.MVMean","Performance.DMSMean","Performance.ADMSLMean","Fatigue
Level"))

sub1NR <- subset(treino1,
select=c("Performance.Task","Performance.MAMean","Performance.KDTMean","P
erformance.DMSMean","FatigueLevel"))
sub2NR <- subset(treino2,
select=c("Performance.Task","Performance.MAMean","Performance.KDTMean","P
erformance.DMSMean","FatigueLevel"))
sub3NR <- subset(treino1,
select=c("Performance.Task","Performance.MAMean","Performance.KDTMean","P
erformance.DMSMean","Performance.DDCMean","FatigueLevel"))
sub4NR <- subset(treino2,
select=c("Performance.Task","Performance.MAMean","Performance.KDTMean","P
erformance.DMSMean","Performance.DDCMean","FatigueLevel"))
sub5NR <- subset(treino1,
select=c("Performance.Task","Performance.MAMean","Performance.KDTMean","P
erformance.DMSMean","Performance.DDCMean","Performance.AEDMean","FatigueL
evel"))
sub6NR <- subset(treino2,
select=c("Performance.Task","Performance.MAMean","Performance.KDTMean","P
erformance.DMSMean","Performance.DDCMean","Performance.AEDMean","FatigueL
evel"))
sub7NR <- subset(treino1,
select=c("Performance.Task","Performance.MAMean","Performance.KDTMean","P
erformance.DMSMean","Performance.DDCMean","Performance.AEDMean","Performa
nce.TBCMean","FatigueLevel"))
sub8NR <- subset(treino2,
select=c("Performance.Task","Performance.MAMean","Performance.KDTMean","P
erformance.DMSMean","Performance.DDCMean","Performance.AEDMean","Performa
nce.TBCMean","FatigueLevel"))
sub9NR <- subset(treino1,
select=c("Performance.Task","Performance.MAMean","Performance.KDTMean","P
erformance.DMSMean","Performance.DDCMean","Performance.AEDMean","Performa
nce.TBCMean","Performance.ADMSLMean","FatigueLevel"))
sub10NR <- subset(treino2,
select=c("Performance.Task","Performance.MAMean","Performance.KDTMean","P
erformance.DMSMean","Performance.DDCMean","Performance.AEDMean","Performa
nce.TBCMean","Performance.ADMSLMean","FatigueLevel"))


# Função que determina a existência ou ausência de
cansaço/fatiga/exaustão
determinaExaustao <- function(dataset)
{
  # 0 - Não existe exaustão
  # 1 - Existe exaustão
  # Dos níveis 1 a 3, considera-se que não existe exaustão
  # Dos níveis 4 a 7, considera-se que existe exaustão

  auxSet <- dataset

  # Se (exaustão <= 0.3), então exaustão = 0
  # Se não exaustão = 1
  auxSet$FatigueLevel <- ifelse(auxSet$FatigueLevel <= 0.3, 0, 0.1)

  return (auxSet);
}
```

```
# Função que determina a exaustão em três níveis distintos
escala3niveis <- function(dataset)
{
  # 0 - Não se encontra exausto (níveis 1, 2 e 3)
  # 0.5 - Encontra-se em risco de exaustão (níveis 4 e 5)
  # 1 - Encontra-se em exaustão (níveis 6 e 7)

  auxSet <- dataset

  # Se (exaustão >= 0.6), então exaustão = 1
  # Se não { Se (exaustão <= 0.3), então exaustão = 0
  #          Se não exaustão = 0.5
  #        }
  auxSet$FatigueLevel <- ifelse(auxSet$FatigueLevel >= 0.6, 0.2,
                                ifelse(auxSet$FatigueLevel <= 0.3, 0,
0.1))

  return (auxSet)
}

# Função que determina a exaustão em quatro níveis distintos
escala4niveis <- function(dataset)
{
  # 0 - Encontra-se num estado físico e mental excelente (nível 1)
  # 0.3 - Encontra-se num estado normal (níveis 2 e 3)
  # 0.6 - Encontra-se cansado, mas não em exaustão completa (níveis 4 e
5)
  # 1 - Encontra-se num estado de exaustão (níveis 6 e 7)

  # Guardam-se os dados numa variável, de modo a não perder os dados
originais
  auxSet <- dataset

  # Se (exaustão >= 0.6), então exaustão = 1
  # Se não { Se (exaustão >= 0.4), então exaustão = 0.6
  #          Se não { Se (exaustão >= 0.2), então exaustão = 0.3
  #                   Se não exaustão = 0
  #                 }
  #        }
  #
  auxSet$FatigueLevel <- ifelse(auxSet$FatigueLevel >= 0.6, 0.3,
                                ifelse(auxSet$FatigueLevel >= 0.4, 0.2,
                                       ifelse(auxSet$FatigueLevel >= 0.2,
0.1, 0)))

  return (auxSet)
}

# Função que treina a rede com as características fornecidas e testa os
dados de treino
exaustao <- function(dados, treino, hidden, formula, t, algoritmo)
{
  # Treino da rede com os parâmetros passados como input
  net <- neuralnet(formula,dados, hidden=hidden, threshold=t,
algorithm=algoritmo, lifesign="full", linear.output = FALSE)
  # Falta acrescentar o linear.output

  # Guardamos o input de treino numa variável distinta, à qual retiramos
o output, de modo a ser possível
```

```
  # testar os dados e, de seguida, compará-los com os originais
  input <- treino
  input$FatigueLevel <- NULL

  # Testamos os dados de input com a rede já treinada
  net$res <- compute(net, input)

  # Guardamos os resultados obtidos num dataframe
  res <- data.frame(atual = treino$FatigueLevel, prev =
net$res$net.result)

  # Arredonda-se os resultados obtidos e guarda-se numa nova variável
  net$prev <- round(res$prev)
  # Acrescentar casas decimais??

  # Comparação entre output original e output depois do treino
  net$rmse <- rmse(c(treino$FatigueLevel), c(net$prev))

  return (net)
}


id7niveis <- function(dataset)
{
  auxSet <- dataset

  v <-
auxSet$Performance.KDTMean+auxSet$Performance.MAMean+auxSet$Performance.M
VMean+auxSet$Performance.TBCMean+auxSet$Performance.DDCMean+auxSet$Perfor
mance.DMSMean+auxSet$Performance.AEDMean+auxSet$Performance.ADMSLMean+aux
Set$Performance.Task

  auxSet$FatigueLevel <- ifelse(v <= 2.34, 0.7,
                                ifelse(v <= 3.28, 0.6,
                                       ifelse(v <= 4.22, 0.5,
                                              ifelse(v <= 5.16, 0.4,
                                                     ifelse(v <= 6.1,
0.3,
                                                            ifelse(v <=
7.04, 0.2, 0.1))))))

  return (auxSet)
}


task <- function(dados, treino, hidden, formula, t, algoritmo)
{
  # Treino da rede com os parâmetros passados como input
  net <- neuralnet(formula,dados, hidden=hidden, threshold=t,
algorithm=algoritmo, lifesign="full", linear.output = TRUE)
  # Falta acrescentar o linear.output

  # Guardamos o input de treino numa variável distinta, à qual retiramos
o output, de modo a ser possível
  # testar os dados e, de seguida, compará-los com os originais
  input <- treino
  input$Performance.Task<- NULL

  # Testamos os dados de input com a rede já treinada
  net$res <- compute(net, input)
```

```
  # Guardamos os resultados obtidos num dataframe
  res <- data.frame(atual = treino$Performance.Task, prev =
net$res$net.result)

  # Arredonda-se os resultados obtidos e guarda-se numa nova variável
  net$prev <- round(res$prev)
  # Acrescentar casas decimais??

  # Comparação entre output original e output depois do treino
  net$rmse <- rmse(c(treino$Performance.Task), c(net$prev))

  return (net)
}

qualTarefa <-function(net, kdt, ma, mv, tbc, ddc, dms, aed, admsl,
exaustion)
{
  teste1<-data.frame(Performance.KDTMean=kdt,Performance.MAMean=ma,
Performance.MVMean=mv,Performance.TBCMean=tbc,
                     Performance.DDCMean =ddc,
Performance.DMSMean=dms,Performance.AEDMean=aed,
Performance.ADMSLMean=admsl,
                     FatigueLevel=exaustion)
  net.results<-compute(net, teste1)
  print(round(net.results$net.result,digits =1))
}

# Resultados vão ficar armazenados numa variável. De forma a verificar o
valor de rmse, executar View(res1$rmse).
res1 <- task(dados2, treino2, hidden1, formulaTask, threshold1, "rprop+")
qualTarefa(res1, 0.1, 0.2, 0.5, 0.2, 0.1, 0.2, 0.3, 0.2, 0.3)

# Resultados vão ficar armazenados numa variável. De forma a verificar o
valor de rmse, executar View(res1$rmse).
#r <- id7niveis(dataset)
#dr1 <- r[1:422,]
#tr1 <- r[423:844,]
#dr2 <- subset(dr1, select=c("FatigueLevel","Performance.DMSMean",
"Performance.TBCMean", "Performance.DDCMean", "Performance.KDTMean"))
#tr2 <- subset(tr1, select=c("FatigueLevel","Performance.DMSMean",
"Performance.TBCMean", "Performance.DDCMean", "Performance.KDTMean"))
#formulaA <- FatigueLevel ~
Performance.DMSMean+Performance.TBCMean+Performance.DDCMean+Performance.K
DTMean
#res7 <- exaustao(dr2, tr2, hidden2, formulaA, threshold1, "rprop+")
#res1 <- exaustao(dados2, treino2, hidden5, formula1, threshold1,
"rprop+")
#ex <- determinaExaustao(dataset) # Novo dataset com exaustão 0 ou 1
#d1 <- ex[1:422,] # Novos dados de aprendizagem
#t1 <- ex[423:844,] # Novos dados de treino
#res2 <- exaustao(d1, t1, hidden2, formula1, threshold1, "rprop+")
#escala3n <- escala3niveis(dataset)
#escala4n <- escala4niveis(dataset)
#d3 <- escala3n[1:422,]
#t3 <- escala3n[423:844,]
#res3 <- exaustao(d3, t3, hidden2, formula1, threshold1, "rprop+")
#escala4n <- escala4niveis(dataset)
#d4 <- escala4n[1:422,]
#t4 <- escala4n[423:844,]
```