Universidade do Minho
Departamento de Informática
Mestrado Integrado em Engenharia Informática

Relatório do Exercício 3

Conhecimento não simbólico: Redes Neuronais Artificiais

Sistemas de Representação de Conhecimento e Raciocínio

Grupo de trabalho 20
Ana Fernandes, A74321
Diogo Machado, A75399
Miguel Miranda, A74726
Rui Leite, A75551

Braga, 7 de Março de 2018

Conteúdo

Resumo

Neste terceiro exercício prático pretende-se explorar a utilização de sistemas não simbólicos na representação de conhecimento e desenvolver mecanismos de raciocínio para a resolução de problemas recorrendo a *Redes Neuronais Artificiais* (*RNAs*).

Ao longo deste relatório é apresentado todo o processo desenvolvido por forma a identificar o nível de exaustão e a tarefa em execução. Foi realizado um estudo dos atributos mais significativos para a representação do conhecimento do problema em análise recorrendo ao WEKA e à função `regsubsets`. De seguida, são identificadas as topologias de rede mais adequadas e selecionadas as regras de aprendizagem para treinar as redes.

Por fim, são apresentados os resultados obtidos para cada topologia de rede e de acordo com a regra de aprendizagem, sendo posteriormente analisados tendo em conta o *Root Mean Square Error* (*RMSE*).

1. Introdução

As *Redes Neuronais Artificiais* (*RNAs*) tratam-se de uma estrutura interconectada de unidades computacionais, designadas por neurónios ou nodos, com capacidade de aprendizagem. Estas permitem criar um modelo simplificado do sistema nervoso central do Ser Humano e são passíveis de serem implementadas por *software* ou *hardware*, e capazes de realizar tarefas após um período de treino. [1, 2].

1.1 Treino de uma *RNA*

No treino de uma *RNA* o conhecimento é armazenado nas conexões entre os neurónios, também designadas por sinapses. Cada neurónio i que possua n conexões de entrada (Figura 1.1), multiplica cada um dos sinais x_j que recebe do neurónio j por uma ponderação (ou peso) w_{ij}, associada à respetiva conexão n. O cálculo do valor total a partir dos n estímulos é feito por um integrador que, frequentemente, utiliza a função de adição ($\sum$), embora possam ser usadas outras. Quando o efeito cumulativo das entradas, i.e., o valor calculado pelo integrador, excede um dado valor limite é usada uma função de ativação fa que condiciona o sinal de saída s_i. [1]

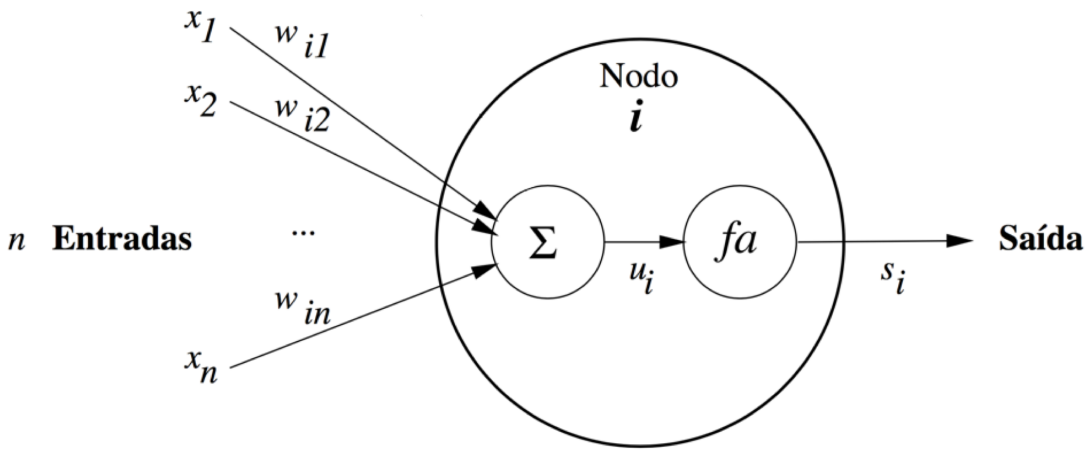


Figura 1.1: Estrutura geral de um neurónio/nodo, adaptado de [1]

1.2 Tipologias de rede

Uma tipologia de rede, ou arquitetura de rede, é entendida como sendo a forma como os nodos se interligam. Existem vários tipos de arquiteturas de *RNAs*, cada uma com as suas características.

Resumidamente, estas podem ser descritas como [1]:

- **Redes *Feedforward*** - uma rede *feedforward* pode ser organizada por camadas e as conexões são sempre unidirecionais (Figura 1.2), não existindo ciclos. Podem ou não ter camadas intermédias, sendo que a sua existência aumenta a capacidade da rede em modelar funções de maior complexidade, revelando-se útil quando o número de nodos na camada de entrada é elevado, porém aumenta de forma exponencial o tempo de aprendizagem.
- **Redes recorrentes** - nas redes recorrentes existem uma ou mais conexões cíclicas. As saídas não são, exclusivamente, em função das conexões entre nodos, i.e. passa-se a estar na presença de um cálculo recursivo que obedecerá a um condição de paragem. O resultado na última iteração é o valor de saída do neurónio.

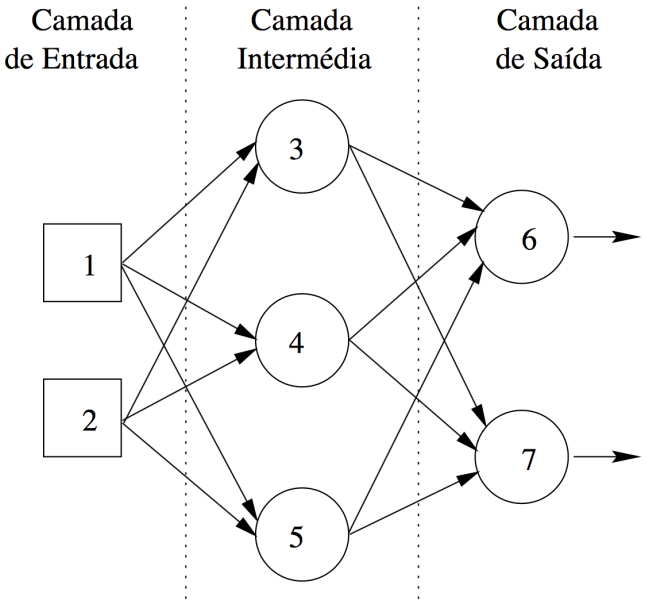


Figura 1.2: Topologia de uma RFMC, de [1]

1.3 Treino e teste de *RNAs* usando R

A linguagem R está assente num ambiente de análise de dados ideal para a definição, treino e teste de *Redes Neuronais Artificiais*. Existem vários *packages* que fornecem métodos e variáveis capazes de caracterizar as RNAs e de retirar toda a informação relevante. Um exemplo disso, que é o usado neste exercício prático, é o *package* NEURALNET. Os métodos usados são explicitados no Capítulo 4 deste relatório.

2. Apresentação do caso de estudo

Neste exercício prático é usada uma *Rede Neuronal Artificial* para a identificação do nível de exaustão e da tarefa em execução por um utilizador na sua interação com um computador.

Os dados que prometem dar suporte a este modelo foram obtidos através dos dispositivos físicos rato e teclado e têm em conta as seguintes biométricas comportamentais:

- ***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 do rato em determinado momento, que corresponde a $\frac{v_{rato}}{t_{movimento}}$, com v_{rato} a velocidade em $^{pixel}/_{ms}$ e $t_{movimento}$ em ms;
- ***Performance.MVMean*** – velocidade do manuseamento do rato em determinado momento. Corresponde a $\frac{d}{t}$, com d a distância percorrida pelo rato, em píxeis, entre uma coordenada $C_1 = (x_1, y_1)$ e uma $C_2 = (x_2, y_2)$ e t, o tempo, em milissegundos;
- ***Performance.TBCMean*** – tempo entre dois *clicks* consecutivos, i.e., 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 *clicks* 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 que calculará a soma da distância entre dois eventos *MOUSE_UP* e *MOUSE_DOWN* consecutivos;
- ***ExhaustionLevel*** – nível subjetivo de exaustão mental;
- ***Performance.Task*** – identificação da tarefa em execução no momento da recolha dos dados.

Os níveis de exaustão mental a usar são uma representação da escala *USAF-SAM Fatigue scale* [3], desenvolvida pelo Dr. Layne Perelli no USA Army, e pode ser interpretada da seguinte forma:

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”.

Todos estes dados são analisados no Capítulo 3 deste relatório.

Motivações e Objetivos

Em todos os exercícios práticos o principal objetivo tem sido colocar em prática todos os conhecimentos adquiridos na Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio. Neste, em particular, são usados conhecimentos baseados na utilização de sistemas não simbólicos na representação de conhecimento e no desenvolvimento de mecanismos de raciocínio. Para isso, serão usadas *Redes Neuronais Artificiais* para a resolução dos problemas propostos.

As *RNAs* trazem vantagens para a representação de conhecimento uma vez que a sua estrutura se baseia no sistema nervoso humano, permitindo obter respostas que se assemelhem às do ser humano.

Uma das principais características das *RNAs* assenta na sua capacidade de aprendizagem, dado que permitem prever *outputs* baseando-se nos dados fornecidos. Possuem auto-organização uma vez que a representação da informação é criada no seu interior, sem que o utilizador a tenha que escolher. Além disto, podem continuar a responder de forma aceitável mesmo que estejam por algum motivo danificadas.

3. Dados

3.1 Análise dos dados

Por forma a identificar o nível de exaustão e a tarefa em execução foi feito um estudo aos dados fornecidos e analisados todos os atributos. Tal como já referido, o conjunto de dados recolhidos resulta da interação humano-computador através dos dispositivos físicos rato e teclado, sendo que para a deteção da exaustão mental recorreu-se à biometria, nomeadamente ao seguinte conjunto de biométricas comportamentais:

- *Performance.KDTMean*
- *Performance.MAMean*
- *Performance.MVMean*
- *Performance.TBCMean*
- *Performance.DDCMean*
- *Performance.DMSMean*
- *Performance.ADMSLMean*
- *Performance.AEDMean*
- *ExhaustionLevel*
- *Performance.Task*

Todas os dados relativos às biométricas de *Performance*, à exceção da *Performance.Task*, já se encontram normalizados no intervalo $[-1, 1]$. Relativamente à *Performance.Task* esta pode ser classificada como: *work*, *office* ou *programming*. A *ExhaustionLevel* toma valores de 1 a 7, sendo o nível 1 classificado como "Totalmente bem" e o nível 7 classificado como "Incapaz de funcionar, pronto a desligar".

Para a realização deste exercício prático foi usado um conjunto de dados, disponibilizado pela equipa docente, constituído por 844 amostras. Foi realizada uma análise aos dados fornecidos e obtidas conclusões relativamente ao *FatigueLevel* e à *Performance.Task*:

FatigueLevel	**Nº Amostras**
1	184
2	257
3	247
4	126
5	26
6	4
7	0

Tabela 3.1: Análise baseada no *FatigueLevel*

Conforme é possível observar na tabela 3.1, a quantidade de dados que refletem níveis de fadiga (*FatigueLevel*) mais elevados são em menor número comparativamente aos dados de nível de fadiga mais baixo. Se considerarmos que a partir

do nível 4 existe efetivamente exaustão, então 81% dos dados fornecidos correspondem à inexistência de exaustão e apenas 19% à existência de exaustão. Isto poder-se-á dever ao facto de quando um ser humano se encontrar cansado não exercer a sua atividade e como tal não ser possível obter dados para níveis de exaustão mais elevados.

Em relação à *Performance.Task*, pode-se ver a distribuição dos dados na tabela 3.2.

Performance.Task	**Nº Amostras**
Work	349
Office	382
Programming	113

Tabela 3.2: Análise baseada na *Performance.Task*

3.2 Processo de normalização dos dados

3.2.1 Normalização dos atributos não numéricos

Apesar de ser possível treinar uma rede neuronal com um qualquer formato de dados, é uma boa prática realizar um pré-processamento dos dados de *input* usados para treinar a rede. Este tratamento prévio permite que o processo de treino da rede se desenrole de forma mais rápida e de forma a exigir menos recursos de memória e a produzir resultados de previsão mais exatos.

Assim, para o conjunto de dados disponibilizado, o primeiro atributo a ser normalizado foi o *Performance.Task*. Como, no conjunto de dados, este registo está sobre a forma de uma *string*, foi criada uma escala numérica que permite converter a designação da tarefa num valor inteiro.

A seguinte tabela procura descrever a conversão linear utilizada.

Tarefa	**Valor**
Work	-1
Office	0
Programming	1

Tabela 3.3: Normalização da *Performance.Task*

3.2.2 Normalização da escala dos atributos

Devido às variações do peso de cada nodo no processo de treino de uma *rede neuronal artificial*, é recomendado que todos os atributos utilizados como *input* de treino da rede estejam dentro de uma escala limitada e bem definida, como por exemplo, contidos no intervalo $[-1, 1]$ ou $[0, 1]$. Apesar deste aspeto não ser expressamente obrigatório, a sua prática facilita o processo de aprendizagem da rede. No conjunto de dados em estudo praticamente todos os atributos já se encontram normalizados num intervalo entre $[-1, 1]$, com exceção do atributo *FatigueLevel* que pode ser um dos 7 níveis possíveis.

Assim, numa primeira fase, foram criadas várias topologias de rede para prever o nível de exaustão, usando a escala inicial de 7 níveis. Posteriormente, com a intenção de melhorar os resultados de previsão obtidos, foi realizada uma diminuição do número de níveis de exaustão, tendo sido apenas considerados dois estados de exaustão: a pessoa apresenta sinais de exaustão, ou não.

A seguinte tabela procura descrever a distribuição dos 7 níveis de cansaço em apenas 2 níveis.

Nível de Exaustão	Escala Inicial	Conversão
Bem	1	0
Responsivo, não no pico	2	0
Normal	3	0
Abaixo do Normal	4	1
Perdendo o foco	5	1
Muito difícil de concentrar	6	1
Incapaz de funcionar	7	1

Tabela 3.4: Normalização do nível de exaustão

3.3 Análise de significância dos dados

Por forma a selecionar os atributos mais vantajosos de acordo com o *output* desejado, foram usadas 2 ferramentas que permitem analisar a dispersão dos dados: WEKA e o *package* LEAPS.

3.3.1 Weka

Optou-se pela utilização do WEKA por se tratar de uma ferramenta que procede à análise computacional e estatística dos dados fornecidos recorrendo a técnicas de mineração de dados, tentando, a partir dos padrões encontrados, gerar possíveis soluções. De acordo com o output pretendido, são fornecidos os atributos que devem ser usados como *input* da *RNA*.

Se considerarmos, por exemplo, que pretendemos criar uma rede com o *FatigueLevel* como nodo de *output*, com escala de 7 níveis e usando o ficheiro dado, obtém-se o seguinte resultado:

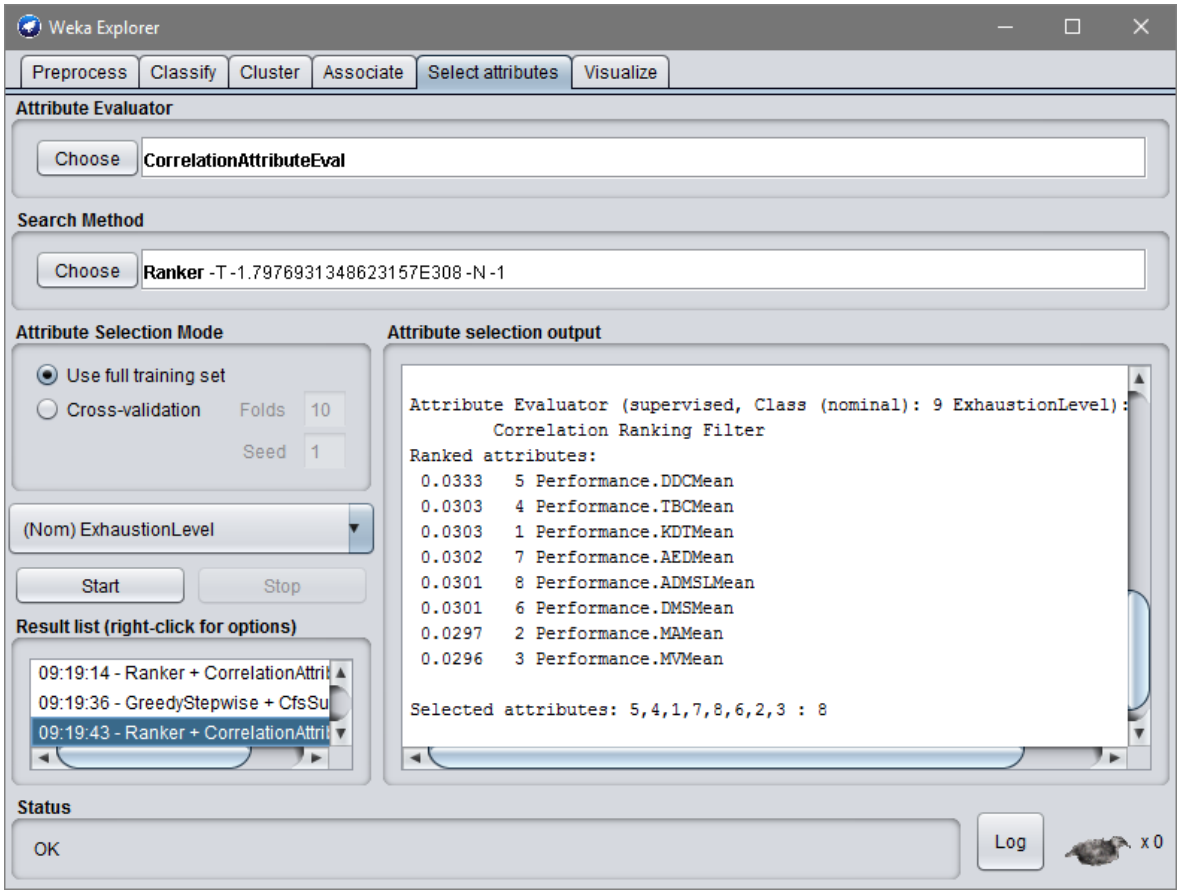


Figura 3.1: Resultado obtido com o Weka para o FatigueLevel não normalizado

3.3.2 Leaps

O *package* `LEAPS` disponibiliza diversas funções que permitem a pesquisa e o tratamento de dados. De entre as diversas funções, optou-se pelo uso da função `regsubsets` que permite a seleção do modelo por busca exaustiva, *forward* ou *backward stepwise* e ainda por substituição sequencial.

Todo o trabalho desenvolvido tem por base os resultados obtidos com esta ferramenta. De seguida apresenta-se um exemplo de utilização do `LEAPS` e também os resultados obtidos utilizando os dados fornecidos, o método de pesquisa *backward selection* e considerando todos os casos, para as duas escala do nível de exaustão e para os seguintes casos de *output*: *FatigueLevel*, *Performance.Task* e *FatigueLevel + Performance.Task*. O resultado obtido pode ser visualizado executando o comando `summary(reggi)`.

FatigueLevel

De seguida são apresentados os resultados tendo como *output* o atributo *FatigueLevel* não normalizado (Figura 3.2) e normalizado (Figura 3.3).

```
> summary(reggi)
Subset selection object
Call: regsubsets.formula(FatigueLevel ~ Performance.KDTMean + Performance.MAMean +
    Performance.MVMean + Performance.TBCMean + Performance.DDCMean +
    Performance.DMSMean + Performance.AEDMean + Performance.ADMSLMean,
    dados, method = "backward")
8 Variables  (and intercept)
                      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
1 subsets of each size up to 8
Selection Algorithm: backward
         Performance.KDTMean Performance.MAMean Performance.MVMean Performance.TBCMean
1  ( 1 ) " "                 " "                " "                " "
2  ( 1 ) " "                 "*"                " "                " "
3  ( 1 ) " "                 "*"                "*"                " "
4  ( 1 ) "*"                 "*"                "*"                " "
5  ( 1 ) "*"                 "*"                "*"                " "
6  ( 1 ) "*"                 "*"                "*"                " "
7  ( 1 ) "*"                 "*"                "*"                "*"
8  ( 1 ) "*"                 "*"                "*"                "*"
         Performance.DDCMean Performance.DMSMean Performance.AEDMean Performance.ADMSLMear
1  ( 1 ) "*"                 " "                " "                " "
2  ( 1 ) "*"                 " "                " "                " "
3  ( 1 ) "*"                 " "                " "                " "
4  ( 1 ) "*"                 " "                " "                " "
5  ( 1 ) "*"                 "*"                " "                " "
6  ( 1 ) "*"                 "*"                " "                "*"
7  ( 1 ) "*"                 "*"                " "                "*"
8  ( 1 ) "*"                 "*"                "*"                "*"
```

Figura 3.2: Resultado obtido para o FatigueLevel não normalizado

```
> summary(reggi)
Subset selection object
Call: regsubsets.formula(FatigueLevel ~ Performance.KDTMean + Performance.MAMean +
    Performance.MVMean + Performance.TBCMean + Performance.DDCMean +
    Performance.DMSMean + Performance.AEDMean + Performance.ADMSLMean,
    dados, method = "backward")
8 Variables  (and intercept)
                      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
1 subsets of each size up to 8
Selection Algorithm: backward
         Performance.KDTMean Performance.MAMean Performance.MVMean Performance.TBCMean
1  ( 1 ) "*"                 " "                " "                " "
2  ( 1 ) "*"                 " "                " "                " "
3  ( 1 ) "*"                 " "                " "                " "
4  ( 1 ) "*"                 "*"                " "                " "
5  ( 1 ) "*"                 "*"                "*"                " "
6  ( 1 ) "*"                 "*"                "*"                "*"
7  ( 1 ) "*"                 "*"                "*"                "*"
8  ( 1 ) "*"                 "*"                "*"                "*"
         Performance.DDCMean Performance.DMSMean Performance.AEDMean Performance.ADMSLMean
1  ( 1 ) " "                 " "                " "                " "
2  ( 1 ) " "                 "*"                " "                " "
3  ( 1 ) " "                 "*"                " "                "*"
4  ( 1 ) " "                 "*"                " "                "*"
5  ( 1 ) " "                 "*"                " "                "*"
6  ( 1 ) " "                 "*"                " "                "*"
7  ( 1 ) "*"                 "*"                " "                "*"
8  ( 1 ) "*"                 "*"                "*"                "*"
```

Figura 3.3: Resultado obtido o FatigueLevel normalizado

Performance.Task

De seguida são apresentados os resultados tendo como *output* o atributo *Performance.Task* normalizado:

```
> summary(reggi)
Subset selection object
Call: regsubsets.formula(Performance.Task ~ Performance.KDTMean + Performance.MAMean +
    Performance.MVMean + Performance.TBCMean + Performance.DDCMean +
    Performance.DMSMean + Performance.AEDMean + Performance.ADMSLMean,
    dados, method = "backward")
8 Variables  (and intercept)
                      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
1 subsets of each size up to 8
Selection Algorithm: backward
         Performance.KDTMean Performance.MAMean Performance.MVMean Performance.TBCMean
1  ( 1 ) "*"                  " "                " "                " "
2  ( 1 ) "*"                  " "                " "                " "
3  ( 1 ) "*"                  " "                " "                " "
4  ( 1 ) "*"                  " "                " "                " "
5  ( 1 ) "*"                  " "                " "                " "
6  ( 1 ) "*"                  " "                " "                "*"
7  ( 1 ) "*"                  " "                "*"                "*"
8  ( 1 ) "*"                  "*"                "*"                "*"
         Performance.DDCMean Performance.DMSMean Performance.AEDMean Performance.ADMSLMean
1  ( 1 ) " "                  " "                " "                " "
2  ( 1 ) " "                  "*"                " "                " "
3  ( 1 ) "*"                  "*"                " "                " "
4  ( 1 ) "*"                  "*"                "*"                " "
5  ( 1 ) "*"                  "*"                "*"                "*"
6  ( 1 ) "*"                  "*"                "*"                "*"
7  ( 1 ) "*"                  "*"                "*"                "*"
8  ( 1 ) "*"                  "*"                "*"                "*"
```

Figura 3.4: Resultado obtido o Performance.Task normalizado

Performance.Task + FatigueLevel

De seguida são apresentados os resultados tendo como *output* o par de atributos *Performance.Task + FatigueLevel* com o atributo *Performance.Task* normalizado e o atributo *FatigueLevel* não normalizado (Figura: 3.5) e normalizado (Figura 3.6):

```
> summary(reggi)
Subset selection object
Call: regsubsets.formula(FatigueLevel + Performance.Task ~ Performance.KDTMean +
    Performance.MAMean + Performance.MVMean + Performance.TBCMean +
    Performance.DDCMean + Performance.DMSMean + Performance.AEDMean +
    Performance.ADMSLMean, dados, method = "backward")
8 Variables  (and intercept)
                     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
1 subsets of each size up to 8
Selection Algorithm: backward
         Performance.KDTMean Performance.MAMean Performance.MVMean Performance.TBCMean
1  ( 1 ) " "                 "*"                " "                " "
2  ( 1 ) " "                 "*"                "*"                " "
3  ( 1 ) " "                 "*"                "*"                " "
4  ( 1 ) " "                 "*"                "*"                " "
5  ( 1 ) " "                 "*"                "*"                " "
6  ( 1 ) " "                 "*"                "*"                " "
7  ( 1 ) " "                 "*"                "*"                "*"
8  ( 1 ) "*"                 "*"                "*"                "*"
         Performance.DDCMean Performance.DMSMean Performance.AEDMean Performance.ADMSLMean
1  ( 1 ) " "                 " "                " "                " "
2  ( 1 ) " "                 " "                " "                " "
3  ( 1 ) "*"                 " "                " "                " "
4  ( 1 ) "*"                 " "                " "                "*"
5  ( 1 ) "*"                 "*"                " "                "*"
6  ( 1 ) "*"                 "*"                "*"                "*"
7  ( 1 ) "*"                 "*"                "*"                "*"
8  ( 1 ) "*"                 "*"                "*"                "*"
```

Figura 3.5: Resultado obtido o FatigueLevel não normalizado

```
> summary(reggi)
Subset selection object
Call: regsubsets.formula(FatigueLevel + Performance.Task ~ Performance.KDTMean +
    Performance.MAMean + Performance.MVMean + Performance.TBCMean +
    Performance.DDCMean + Performance.DMSMean + Performance.AEDMean +
    Performance.ADMSLMean, dados, method = "backward")
8 Variables  (and intercept)
                     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
1 subsets of each size up to 8
Selection Algorithm: backward
         Performance.KDTMean Performance.MAMean Performance.MVMean Performance.TBCMean
1  ( 1 ) " "                 " "                " "                " "
2  ( 1 ) "*"                 " "                " "                " "
3  ( 1 ) "*"                 " "                " "                " "
4  ( 1 ) "*"                 " "                " "                " "
5  ( 1 ) "*"                 " "                "*"                " "
6  ( 1 ) "*"                 "*"                "*"                " "
7  ( 1 ) "*"                 "*"                "*"                " "
8  ( 1 ) "*"                 "*"                "*"                "*"
         Performance.DDCMean Performance.DMSMean Performance.AEDMean Performance.ADMSLMean
1  ( 1 ) " "                 " "                " "                "*"
2  ( 1 ) " "                 " "                " "                "*"
3  ( 1 ) "*"                 " "                " "                "*"
4  ( 1 ) "*"                 " "                "*"                "*"
5  ( 1 ) "*"                 " "                "*"                "*"
6  ( 1 ) "*"                 " "                "*"                "*"
7  ( 1 ) "*"                 "*"                "*"                "*"
8  ( 1 ) "*"                 "*"                "*"                "*"
```

Figura 3.6: Resultado obtido o FatigueLevel normalizado

4. Redes Neuronais Artificiais

4.1 Definição do conjunto dados

Decidiu-se criar *Redes Neuronais* supervisionadas. Deste modo, do conjunto de dados inicial, foram selecionados dois subconjuntos: um para ser utilizado como dados de treino da rede e o outro conjunto para ser utilizado como exemplos de teste, por forma a serem comparados com os resultados de previsão dados pela rede treinada.

Para esta divisão utilizaram-se cerca de dois terços ($^2/_3$) dos dados iniciais para treino e o restante terço ($^1/_3$) para testar a rede, permitindo calcular o erro na capacidade de previsão da rede.

Pelo facto de o conjunto de dados inicial estar organizado pelo parâmetro referente ao nível de fadiga do utilizador, a divisão dos dados nos dois subconjuntos referidos não poderia ser feita de forma direta, utilizando os primeiros $^2/_3$ dos dados para treino e os restantes para teste. Se tal decisão tivesse sido tomada, os dados de teste iriam conter dados pouco significativos.

Deste modo, o conjunto de dados inicial foi dividido em vários subconjuntos, onde cada subconjunto continha apenas as entradas referentes a um determinado nível de fadiga. Estes subconjuntos foram ainda reorganizados de forma aleatória e de cada um deles foram retirados os primeiros $^2/_3$ de entradas para o conjunto de dados de treino e o restante terço para o conjunto de dados de teste. Antes de reunidos estes dois subconjuntos num conjunto final, os dados de cada subconjunto voltaram ainda a ser reorganizados de forma aleatória.

Uma vez que todo o processo de reorganização dos dados foi realizado recorrendo a uma ferramenta externa (EXCEL), a nível de implementação em R foi apenas necessário carregar os dados para uma variável, utilizando a função `read.csv`, e sobre ela definir que entradas serão usadas para treino e para teste. Como no conjunto de dados fornecido para estudo existem 844 registos, pela divisão dos dados já referida, os primeiros 559 ($^2/_3$) foram movidos para treino e os restantes 285 ($^1/_3$) usados para teste.

O seguinte excerto de código procura descrever este procedimento de divisão de dados.

```
dados <- read.csv("dir to file"/exaustao_FatigueLevel_7.csv",
                  header= TRUE, sep = ';', dec = ',')
treino <- dados[1:559,]
teste <- dados[559:844,]
```

Para o processo de treino da rede, é definido o conjunto de nodos de *input*, através de uma fórmula, usada nas iterações de treino dos nodos da rede.

Para executar o treino da rede, com o conjunto de dados de treino e a fórmula definida, é utilizada a função `neuralnet` existente no *package* NEURALNET. Nos campos dos seus argumentos são passados a fórmula, o conjunto de dados a utilizar no processo de treino da rede e ainda alguns parâmetros de controlo, como por exemplo:

- O campo *hidden*, onde se cria topologia da rede indicando o número de camadas intermédias e o número de nodos de cada camada;
- O campo *threshold*, indicando o limite mínimo sobre o qual a rede deve convergir;
- O campo *stepmax*, limitando o número de iterações máximo que a rede pode executar. Acima deste valor, considera-se que a rede não converge para a topologia indicada em tempo aceitável.
- O campo *lifesign*, marcado com o valor `"full"`, para ser possível analisar a evolução do valor do threshold e, caso a rede termine o seu processo de treino, ser possível obter o erro associado a esse threshold.

Para as redes desenvolvidas, o parâmetro threshold, stepmax e lifesign foram utilizados sempre com valores estáticos: `threshold = 0.01; stepmax = 1e+08; lifesign = "full"`.

O seguinte excerto procura mostrar a criação de uma fórmula e de uma variável, designada por `rna`, sobre a qual estarão os resultados obtidos através do treino de uma rede com três camadas intermédias.

```
formula01 <- Performance.Task ~ Performance.KDTMean
                              + Performance.DMSMean
rna <- neuralnet(formula01, treino,
                 hidden = c(2,1), threshold = 0.01,
                 stepmax = 1e+08  , lifesign = "full")
```

Com a função `compute`, também existente no *package* NEURALNET, são comparadas as respostas dadas pela rede, após o seu treino ter finalizado, com os dados de teste. Será a partir desta comparação entre as previsões da rede e os valores registados que se irá determinar o valor do erro *rmse*.

```
teste.01 <- subset(teste,select = c("Performance.KDTMean",
                                    "Performance.DMSMean"))
rna.resultados <- compute(rna, teste.01)
resultados <- data.frame(atual  = teste$Performance.Task,
                         previsao = rna.resultados$net.result)
rmse(c(teste$Performance.Task), c(resultados$previsao))
```

4.2 Definição das *RNAs*

4.2.1 *RNA* com *FatigueLevel* como *output*

Para a criação das redes capazes de estimar o nível de fadiga do utilizador, através dos seus registos biométricos, foi tido em consideração o nível de significância atribuído a cada um destes parâmetros, conforme os resultados obtidos com o método `regsubsets` existente no *package* LEAPS já referido.

Como com a construção de uma RNA para previsão do nível de Fadiga se procura prever exatamente um dos parâmetros que no conjunto de dados inicial não se encontrava normalizado, foi tomada a decisão de criar inicialmente uma rede neuronal para previsão do nível de fadiga com os 7 níveis iniciais e, posteriormente, repetidos os mesmos procedimentos de forma a prever este parâmetro normalizado.

Assim, para o caso do conjunto de dados com 7 níveis de fadiga (não normalizado), os resultados obtidos relativamente à significância dos atributos já foram apresentados na Figura 3.2. Para o caso do conjunto de dados já normalizado, onde o nível de fadiga foi limitado a um intervalo mais restrito, os resultados encontram-se na Figura 3.3.

Para os dois contextos referidos, foi criado um conjunto de 8 fórmulas. Estas fórmulas foram criadas acrescentando sucessivamente à fórmula anterior um novo parâmetro biométrico, conforme o seu grau de importância atribuído pelo método `regsubset`.

Como a intenção geral deste projeto recai sobre a previsão do nível de Fadiga e da *Task*, na abordagem inicial de criação de redes para prever cada um dos parâmetros individualmente, foi considerado que um parâmetro não seria utilizado para determinar o outro. Assim as entradas relativas ao atributo Performance.Task não serão utilizadas no treino das redes neuronais de previsão do nível de Fadiga.

A titulo de exemplo, para o caso do conjunto de dados com o nível de Fadiga distribuído em dois níveis, as formula criadas seguem a seguinte metodologia:

```
formula1 <- Performence.Task ~ Performance.KDTMean
formula2 <- Performence.Task ~ Performance.KDTMean
                             + Performance.DMSMean
(...)
formula8 <- Performence.Task ~ Performance.KDTMean
                             + Performance.DMSMean
                             + Performance.ADMSLMean
                             + Performance.MAMean
                             + Performance.MVMean
                             + Performance.TBCMean
                             + Performance.DDCMean
                             + Performance.AEDMean
```

Para cada uma destas fórmulas, foram criadas diferentes topologias da rede neuronal, explorando os níveis das camadas intermédias da rede, procurando assim obter um *rmse* mais reduzido.

Para este processo de treino e determinação do erro de previsão da rede, foram utilizadas as funções e procedimentos explicados no tópico anterior.

4.2.2 *RNA* com *Performance.Task* como *output*

Como na formulação das redes neuronais para previsão da Tarefa do utilizador, se considerou que o parâmetro FatigueLevel não seria utilizado na formula, foram apenas utilizados como dados de treino da rede o conjunto de dados com os níveis da Fadiga já normalizados a dois níveis.

De forma semelhante à metodologia utilizada para a criação de redes para prever o nível de Fadiga, foi novamente tido em consideração os níveis de significância atribuídos a cada um dos parâmetros biométricos, conforme os resultados obtidos com o método `regsubsets` do *package* LEAPS. Para este conjunto de dados utilizado, e para a formula usada para estimar o tarefa do utilizador, os resultados obtido pelo método `regsubsets` encontram-se na figura 3.4.

Tal como na previsão do nível de Fadiga, foi novamente criado um conjunto de 8 fórmulas, acrescentando sucessivamente um novo parâmetro biométrico, conforme o seu grau de importância atribuído pelo método `regsubset`. Perante os níveis de significância obtidos, as formula criadas podem ser descritas da seguinte forma:

```
formula1 <- FatigueLevel ~ Performance.KDTMean
formula2 <- FatigueLevel ~ Performance.KDTMean
                         + Performance.DMSMean
(...)
formula8 <- FatigueLevel ~ Performance.KDTMean
                         + Performance.DMSMean
                         + Performance.DDCMean
                         + Performance.AEDMean
                         + Performance.ADMSLMean
                         + Performance.TBCMean
                         + Performance.MVMean
                         + Performance.MAMean
```

4.2.3 *RNA* com *FatigueLevel* e *Performance.Task* como *output*

Com as metodologias descritas nas secções anteriores foram construidas redes neuronais com a capacidade de previsão dos parâmetros Tarefa e Fadiga, analisando-os separadamente. Depois desta abordagem inicial, foi tomada a decisão de criar outras redes com os mesmos casos de treino, mas de forma a tentar prever os dois parâmetros biométricos em simultâneo.

Assim como no procedimento de construção das redes anteriores, na elaboração da formula a usar no processo de treino da rede deu-se ênfase aos níveis de significância dos parâmetros biométricos obtidos com o método `regsubsets`. Estes resultados podem ser consultados na figura 3.5 para o conjunto de dados sem o parâmetro Fadiga normalizado, e na figura 3.6 para o conjunto de dados com a Fadiga limitada a dois níveis.

Para esta abordagem de criação de uma rede neuronal com dois nodos de saída, desenvolvemos novamente várias formulas de treino, considerando gradualmente um novo parâmetro biométrico. Como exemplo, apresentam-se as formulas utilizadas para estimar a Tarefa e a Fadiga, com o conjunto de dado iniciais não normalizado.

```
formula1 <- FatigueLevel + Performance.Task ~
                            Performance.MAMean
formula2 <- FatigueLevel + Performance.Task ~
                              Performance.MAMean
                            + Performance.MVMean
(...)
formula8 <- FatigueLevel + Performance.Task ~
                              Performance.MAMean
                            + Performance.MVMean
                            + Performance.DDCMean
                            + Performance.ADMSLMean
                            + Performance.DMSMean
                            + Performance.AEDMean
                            + Performance.TBCMean
                            + Performance.KDTMean
```

5. Resultados

De seguida são apresentados os resultados obtidos com todos os treinos e testes para cada uma das *RNAs*. É apresentada uma tabela por cada uma das fórmulas testadas, cuja correspondência pode ser consultada no Capítulo 4.

5.1 *RNA* com *FatigueLevel* como *output*

5.1.1 Escala de fadiga com 7 níveis

Tabela 5.1: Fórmula 1

c	rmse	Iter	Erro
2,1	1,1251	731	328,3334
3,2	1,1371	30244	318,1925
2	1,1241	1389	328,2375

Tabela 5.2: Fórmula 2

c	rmse	Iter	Erro
2,1	1,1332	2476	311,31513
3,2	1,1517	26786	298,60031
2	1,1281	7543	311,00624

Tabela 5.3: Fórmula 3

c	rmse	Iter	Erro
3	1,0896	10730	296,4406
3,2	1,1452	15577	293,5584
4,3	1,1077	110046	272,3855

Tabela 5.4: Fórmula 4

c	rmse	Iter	Erro
4	1,1146	79513	286,6090
2,1	1,1054	12711	300,9205
5,4	1,1677	76327	254,5009
3,2	1,1246	43467	287,4130
4,3,1	1,1410	76327	254,5009

Tabela 5.5: Fórmula 5

c	rmse	Iter	Erro
5	1,1358	11076	255,8723
3,2	1,1225	52735	266,4886
5,3,1	1.1551	67264	253.8926

Tabela 5.6: Fórmula 6

c	rmse	Iter	Erro
6	1.1759	24024	241.1201
7,4	1.2660	542030	178.5426
5,3,3	1.3275	471160	212.0553

Tabela 5.7: Fórmula 7

c	rmse	Iter	Erro
6	1.1776	18008	232.86327
5,3	1.1513	346132	223.1376
5,2,1	1.1505	336371	230.21958

Tabela 5.8: Fórmula 8

c	rmse	Iter	Erro
6,4	1.2755	143005	185.5491
9,4	1.2564	154931	147.6020
3,2,1	1.1246	1535380	263.5485

5.1.2 Escala de fadiga com 2 níveis

Tabela 5.9: Fórmula 1

c	rmse	Iter	Erro
4	0,3899	52	41,9480
2,1	0,3898	58	41,9557
4,3	0,3914	228	41,8658
5,4,3	0,3865	45465	38,1218

Tabela 5.10: Fórmula 2

c	rmse	Iter	Erro
4	0,3963	21954	39,3839
3,2	0,3982	8431	39,7337
4,3,2	0,4160	16130	37,5980
6,3	0,4598	32460	37,6974

Tabela 5.11: Fórmula 3

c	rmse	Iter	Erro
4,3	0,4107	10614	37,3112
7,2	0,4017	47265	32,9074
5,3,2	0,3992	2384	39,2725
3,2,1	0,3840	6517	37,7885

Tabela 5.12: Fórmula 4

c	rmse	Iter	Erro
7,2	0,4153	81762	30,1844
4,3,2	0,4316	92976	32,8967
7,5,3,2	0,5009	109673	25,9902

Tabela 5.13: Fórmula 5

c	rmse	Iter	Erro
4,3	0,4775	4888	30,2811
7,4	0,5391	56219	25,0211
5,3	0,4767	108223	27,0177
5,4	1,0031	8370	33,7759
8,4,2	0,4778	9676	21,6044

Tabela 5.14: Fórmula 6

c	rmse	Iter	Erro
5,3,2,1	0,4109	1310	38,1397
5,4	0,5151	76604	24,9555
7,4	0,5075	70822	24,8928
8,5	0,5440	30872	15,5149
6,4,2	0,4133	1016	37,3536

Tabela 5.15: Fórmula 7

c	rmse	Iter	Erro
5,3	0,4605	8695	25,5009
7,5	0,5652	54877	17,4617
7,4,2	0,5286	72574	11,6993

Tabela 5.16: Fórmula 8

c	rmse	Iter	Erro
6,4	0,4983	35349	23,2496
6,5	0,5162	51935	22,6710
8,4	0,5299	15191	20,7259
7,4,2	0,5159	7820	17,7890

5.2 *RNA* com *Performance.Task* como *output*

Tabela 5.17: Fórmula 1

c	rmse	Iter	Erro
3	0.6744	17462	118.0752
2,1	0.6386	3839	121.1681
3,2	0.6573	29215	118.4982

Tabela 5.18: Fórmula 2

c	rmse	Iter	Erro
5	0.7055	55645	99.6197
2,1	0.6846	5898	117.7770
3,2	0.6564	68090	108.8796

Tabela 5.19: Fórmula 3

c	rmse	Iter	Erro
5	0.6848	9733	104.2138
7	0.6375	11625	120.0549
3,2	0.6538	23071	114.1109
4,2	0.6523	9832	91.67299

Tabela 5.20: Fórmula 4

c	rmse	Iter	Erro
5	0.7078	14368	95.5090
5,3	0.7120	36871	88.0172
2,1	0.6509	74120	112.3220
5,3,1	0.6550	20665	118.4699
3,2	0.6320	44939	94.4944

Tabela 5.21: Fórmula 5

c	rmse	Iter	Erro
5	0.7470	10094	91.9761
6	0,6691	19653	90.4069
5,3	0.7639	28048	79.8679
6,4	0.7680	63181	67.0353
5,3,2	0.7319	192962	80.6823

Tabela 5.22: Fórmula 6

c	rmse	Iter	Erro
5	0.7449	11442	82.3924
5,3	0.7884	142623	66.0236
5,3,2	0.7988	191987	67.9086

Tabela 5.23: Fórmula 7

c	rmse	Iter	Erro
6	0.7588	26981	76.9434
6,4	0.8175	25904	52.8933
7,5,3,1	0.6530	716	101.3081

Tabela 5.24: Fórmula 8

c	rmse	Iter	Erro
5,3	0.7928	29041	63.3931
5,3,1	0.6673	2922	109.9543
7,5,3,1	0.8043	74648	60.04124

5.3 *RNA* de teste do nível de fadiga e da tarefa em execução

5.3.1 Escala de fadiga com 7 níveis

Tabela 5.25: Fórmula 1

c	rmseF	rmseT	Iter	Erro	Média rmse
4	1,1325	0,6972	87915	450,5821	0,9149
2,1	1,1111	0,6783	2031	447,2148	0,8947

Tabela 5.26: Fórmula 2

c	rmseF	rmseT	Iter	Erro	Média rmse
4	1,1343	0,6952	9921	426,6351	0,9148
2,2	1,1230	0,6992	248815	437,4696	0,9111
4,3	1,1262	0,6416	201438	409,9026	0,8839

Tabela 5.27: Fórmula 3

c	rmseF	rmseT	Iter	Erro	Média rmse
4	1,1231	0,6670	12975	417,0761	0,8951
3,2	1,1462	0,6928	13139	420,8837	0,9195
4,3	1,1659	0,7160	234986	398,7105	0,9410
4,3,2	1,1384	0,7110	89382	402,2870	0,9247

Tabela 5.28: Fórmula 4

c	rmseF	rmseT	Iter	Erro	Média rmse
4	1,1315	0,6885	10786	394,0055	0,9100
3,2	1,1670	0,6976	61823	407,6308	0,9323
4,3	1,1854	0,6833	13792	397,5780	0,9344

Tabela 5.29: Fórmula 5

c	rmseF	rmseT	Iter	Erro	Média rmse
4	1,1237	0,7063	17779	397,4410	0,9150
3,2	1,1596	0,6630	125172	386,5234	0,9113
4,3	1,1857	0,6981	50619	383,5194	0,9419
4,3,1	1,1274	0,6986	28178	384,3425	0,9130

Tabela 5.30: Fórmula 6

c	rmseF	rmseT	Iter	Erro	Média rmse
5,3	1,1594	0,68	61933	351,5118	0,9197
5,3,1	1,2386	0,6748	588291	321,3543	0,9567
7,4,2,1	1,3251	0,7002	779837	295,2649	1,01265

Tabela 5.31: Fórmula 7

c	rmseF	rmseT	Iter	Erro	Média rmse
5,3	1,1762	0,6607	60698	342,5786	0,91845
5,3,1	1,1485	0,6641	32332	360,71261	0,9063
7,3,1	1,1718	0,6942	62503	346,8574	0,9330

Tabela 5.32: Fórmula 8

c	rmseF	rmseT	Iter	Erro	Média rmse
6,3	1,2563	0,6754	537611	296,008	0,96585
7,4,2	1,3472	0,6949	270567	276,7875	1,02105
7,4,2,1	1,1759	0,6894	3852	339,645	0,93265

5.3.2 Escala de fadiga com 2 níveis

Tabela 5.33: Fórmula 1

c	rmseF	rmseT	Iter	Erro	Média rmse
2	0,3869	0,6939	15337	157,5461	0,5404
3,2	0,3927	0,6936	117669	156,2822	0,5431
3,2,1	0,3877	0,6920	15910	156,3508	0,5398

Tabela 5.34: Fórmula 2

c	rmseF	rmseT	Iter	Erro	Média rmse
3	0,3870	0,6868	27221	154,7609	0,5369
3,2	0,4214	0,6977	44170	153,6185	0,5595
3,2,1	0,3922	0,7029	4024	163,3466	0,5475

Tabela 5.35: Fórmula 3

c	rmseF	rmseT	Iter	Erro	Média rmse
4	0,3903	0,6843	328495	152,4073	0,5373
3,2	0,3888	0,7092	100974	149,0733	0,5490
3,2,1	0,3895	0,6918	16594	143,3463	0,5406

Tabela 5.36: Fórmula 4

c	rmseF	rmseT	Iter	Erro	Média rmse
3	0,3835	0,6917	1266	147,7253	0,5376
3,2	0,3873	0,6548	125996	139,0436	0,5210
4,3,2	0,3841	0,7110	70091	137,7081	0,5475

Tabela 5.37: Fórmula 5

c	rmseF	rmseT	Iter	Erro	Média rmse
5	0,3853	0,6663	10047	126,8236	0,5258
5,3	0,3872	0,6637	23831	113,7560	0,5254
6,3	0,3884	0,6926	81979	99,1960	0,5405

Tabela 5.38: Fórmula 6

c	rmseF	rmseT	Iter	Erro	Média rmse
6	0,3984	0,6863	20998	119,9995	0,54235
5,4	0,4094	0,6381	32506	100,9852	0,52375
5,4,1	0,3944	0,7041	12984	113,0544	0,54925

Tabela 5.39: Fórmula 7

c	rmseF	rmseT	Iter	Erro	Média rmse
6	0,3966	0,6058	9623	106,9555	0,5012
5,4	0,3929	0,6978	53212	98,7469	0,5453
4,3,2	0,3881	0,5859	57375	112,6273	0,4870

Tabela 5.40: Fórmula 8

c	rmseF	rmseT	Iter	Erro	Média rmse
7	0,4290	0,6193	47880	114,4950	0,52415
8,6	0,4003	0,7526	80403	79,2273	0,57645

6. Análise dos resultados

Depois e obtidos todos os resultados que se pretendiam, é necessário analisá-los e procurar obter conclusões sobre cada um deles e sobre qual a melhor tipologia e fórmula da *Rede Neuronal Artificial*.

Numa primeira instância pretende-se, para cada uma das *RNAs* consideradas, concluir sobre a fórmula que melhores resultados trás e qual a tipologia. Numa segunda instância, sobre qual a *RNA* que melhor responde à determinação do nível de exaustão mental e à tarefa em execução.

6.1 *RNA* com *FatigueLevel* como *output*

Em relação à *Rede Neuronal Artificial* em que o *output* é somente o nível de exaustão, chegou-se à conclusão que:

O melhor resultado com escala de 7 níveis é: uma rede com três nodos de *input* (*Performance.DDCMean*, *Performance.MAMean* e *Performance.MVMean*) e 2 camadas intermédias, com 4 e 3 nodos, respetivamente (Figura 6.1). O teste obtido resultou num *rmse* de $1,1077$ e foi calculado em 110046 iterações.

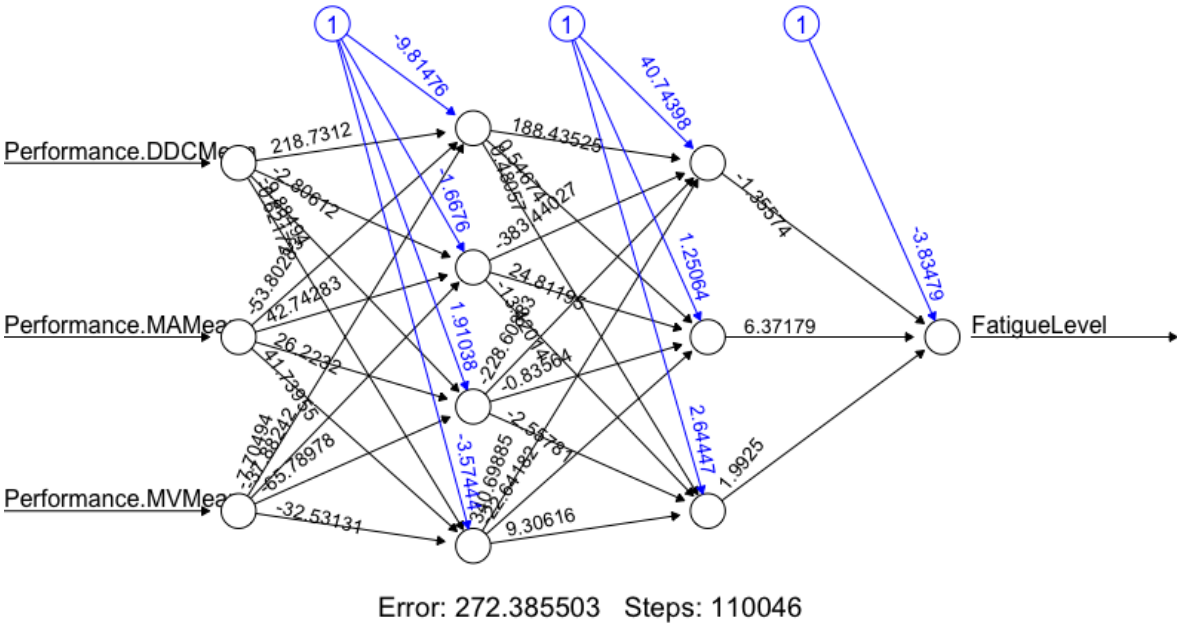


Figura 6.1: *RNA* com *FatigueLevel* como *output*

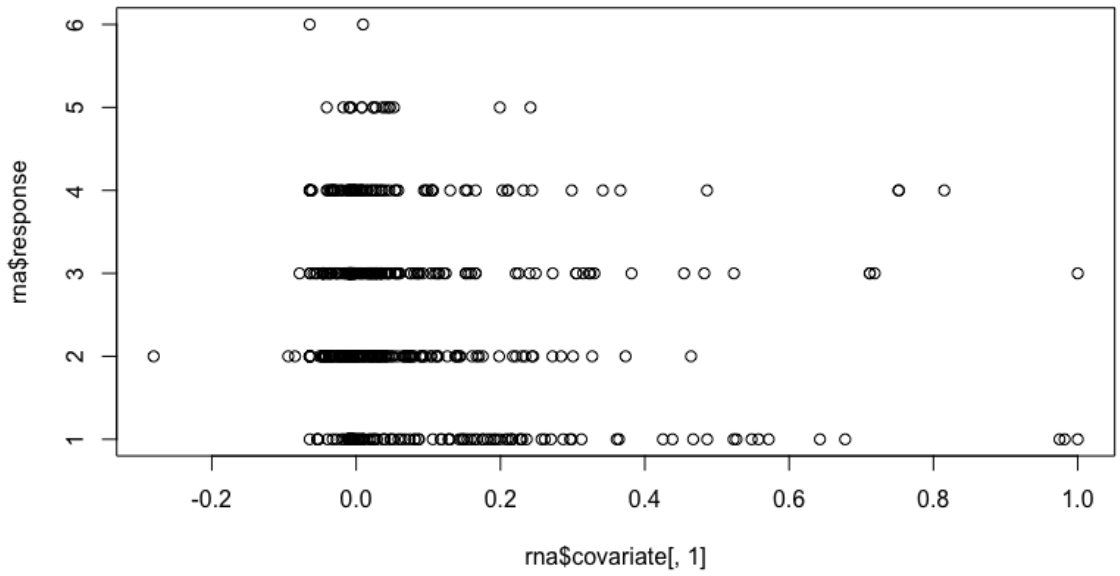


Figura 6.2: Covariância dos resultados da *RNA* com *FatigueLevel* como *output*, numa escala de 7 níveis

O melhor resultado com escala de 2 níveis é: uma rede com três nodos de *input* (*Performance.KDTMean*, *Performance.DMSMean* e *Performance.ADMSLMean*) e 3 camadas intermédias, com 3, 2 e 1 nodos, respetivamente (Figura 6.3). O teste obtido resultou num *rmse* de $0,3840$ e foi calculado em 6517 iterações.

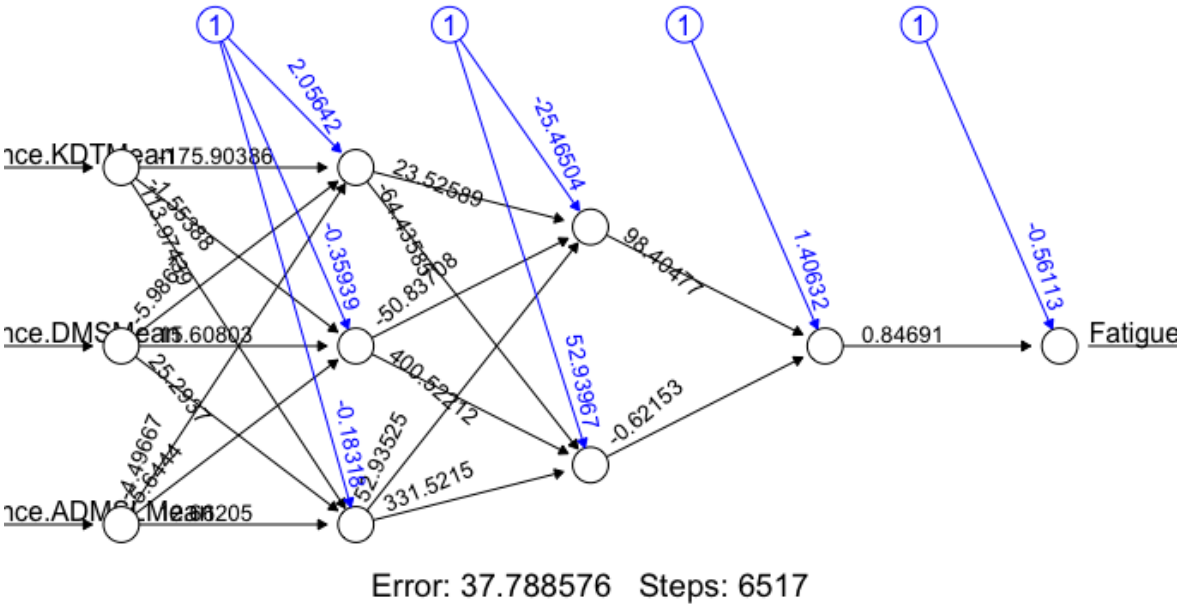


Figura 6.3: *RNA* com *FatigueLevel* como *output*

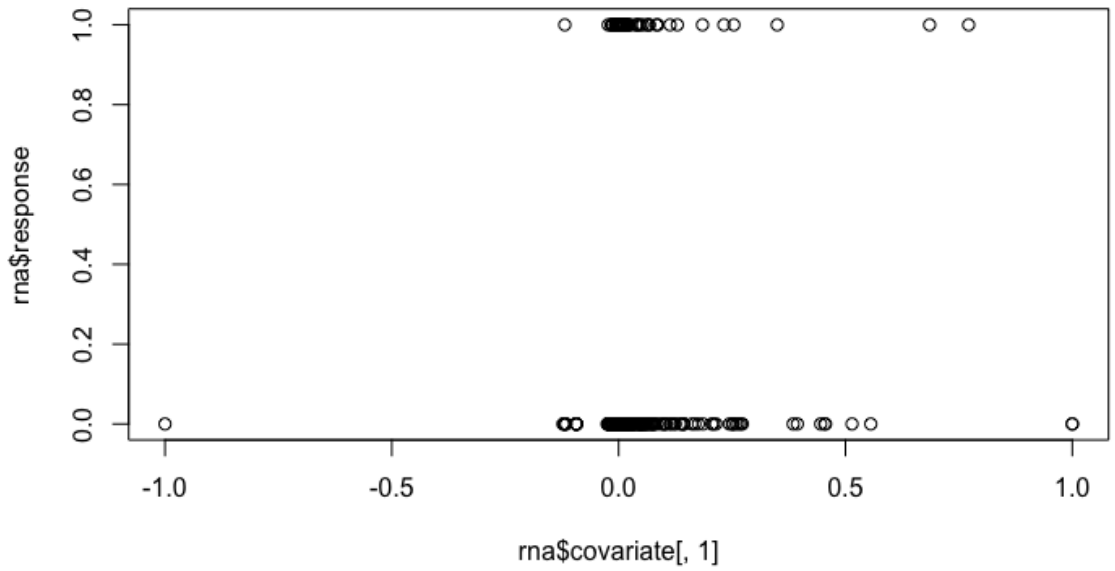


Figura 6.4: Covariância dos resultados da *RNA* com *FatigueLevel* como *output*, numa escala de 2 níveis

Pode-se concluir que a melhor *RNA* é: a rede em que a exaustão vem expressa em 2 níveis. Este resultado já era esperado, uma vez que, para além de não existirem dados suficientes para a rede aprender sobre os níveis mais altos de Fadiga, é natural que, tendo uma escala bastante mais reduzida, a evolução e a aprendizagem sejam mais fácil.

Em relação à dispersão dos resultados, é possível verificar, pelas Figuras 6.2 e 6.4, que a rede treinada com os dados da exaustão numa escala de 7 níveis produz resultados bastante mais dispersos em relação à rede treinada com os dados da exaustão numa escala de 2 níveis.

6.2 *RNA* com *Performance.Task* como *output*

Em relação à *Rede Neuronal Artificial* em que o *output* é somente a tarefa em execução, chegou-se à conclusão que:

O melhor resultado é: uma rede com 4 nodos de *input* (*Performance.KDTMean*, *Performance.DMSMean*, *Performance.DDCMean* e *Performance.AEDMean*) e 2 camadas intermédias, com 4 e 3 nodos, respetivamente (Figura 6.5). O teste obtido resultou num *rmse* de $0,6320$ e foi calculado em 44939 iterações.

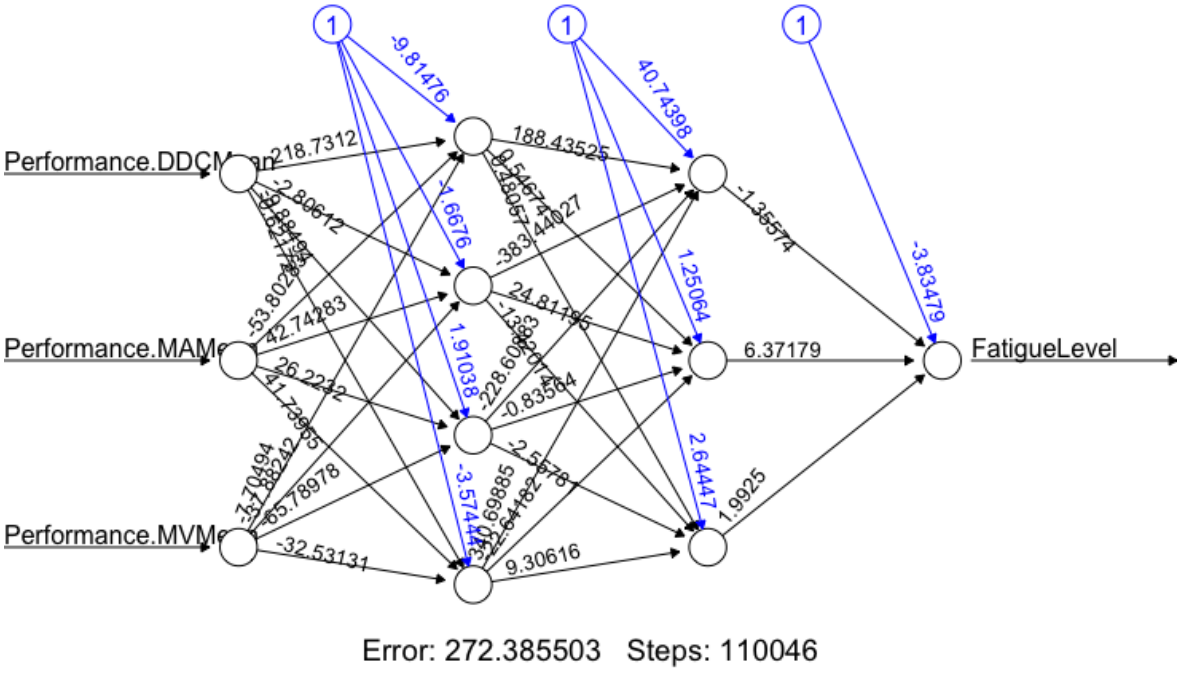


Figura 6.5: *RNA* com *Performance.Task* como *output*

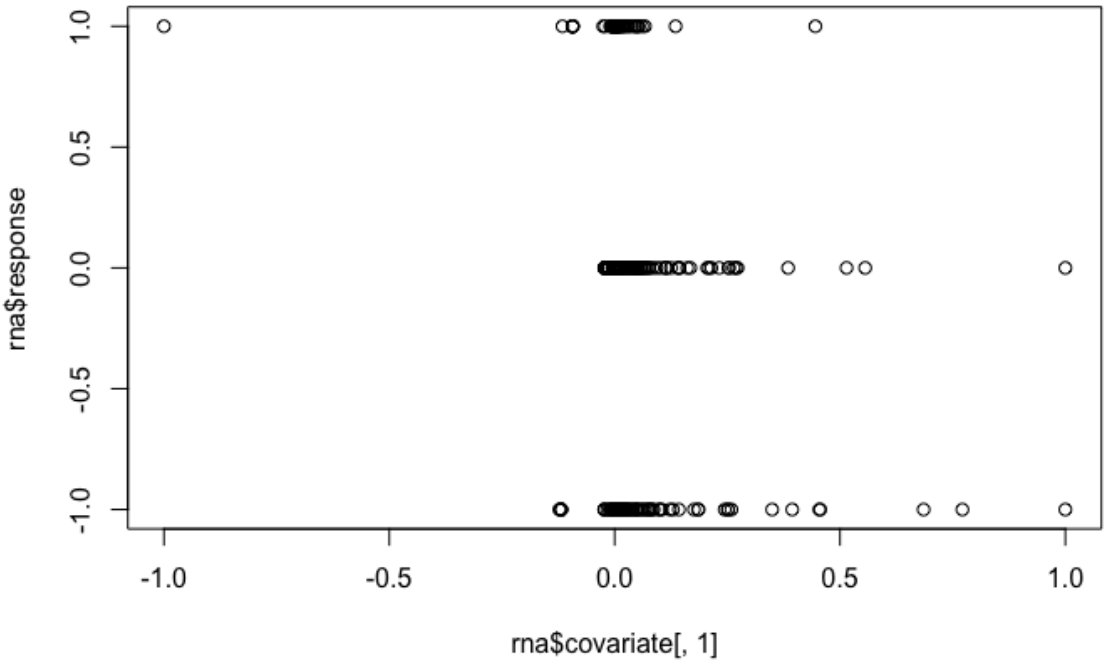


Figura 6.6: Covariância dos resultados da *RNA* com *Performance.Task* como *output*

6.3 *RNA* com *FatigueLevel* e *Performance.Task* como *output*

Em relação à *Rede Neuronal Artificial* em que o *output* é o nível de exaustão e também a tarefa em execução, chegou-se à conclusão que:

O melhor resultado com escala de 7 níveis é: uma rede com 2 nodos de *input* (*Performance.MAMean* e *Performance.MVMean*) e 2 camadas intermédias, com 4 e 3 nodos, respetivamente (Figura 6.7). O teste obtido resultou num *rmse* em relação à Fadiga de $1,1262$ e em relação à Tarefa de $0,6416$, resultando num *rmse* médio de $0,8839$, e foi calculado em 201438 iterações.

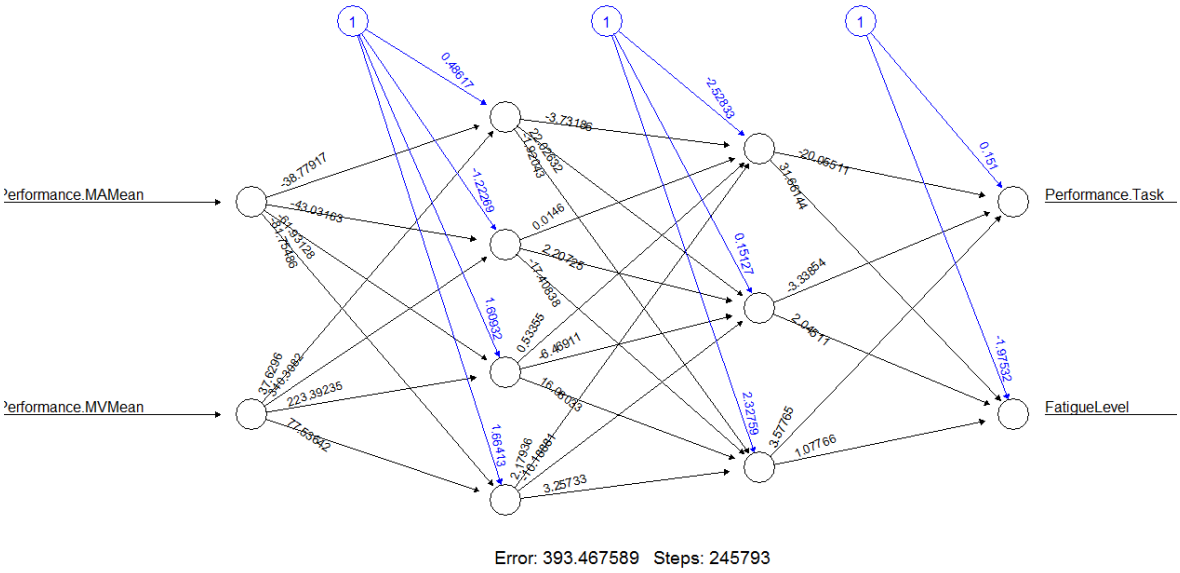
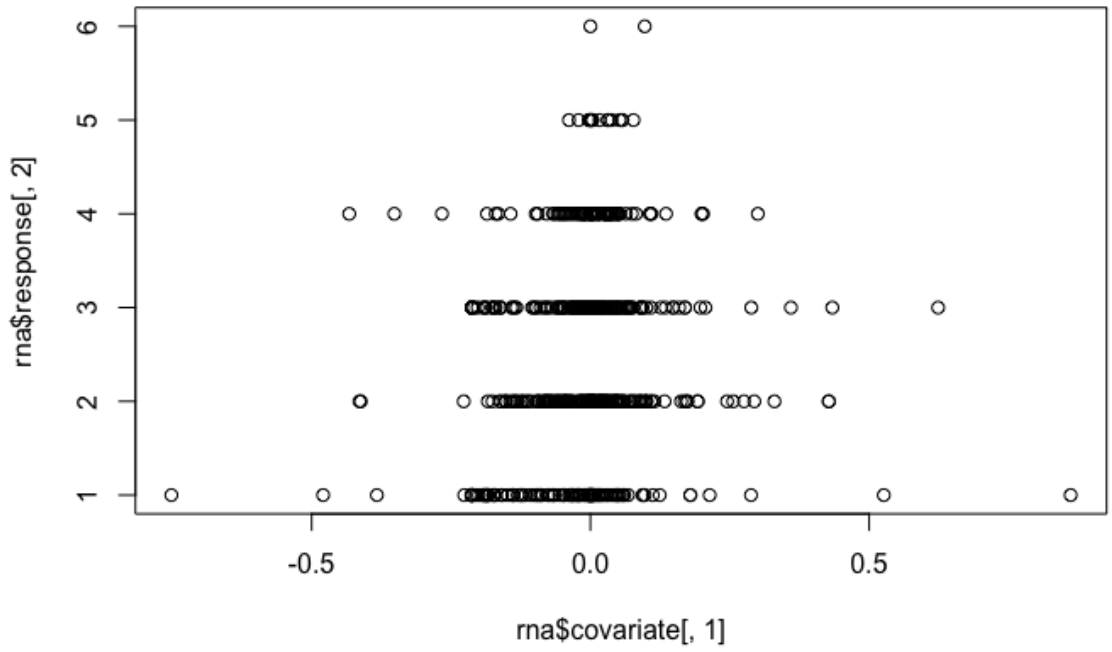
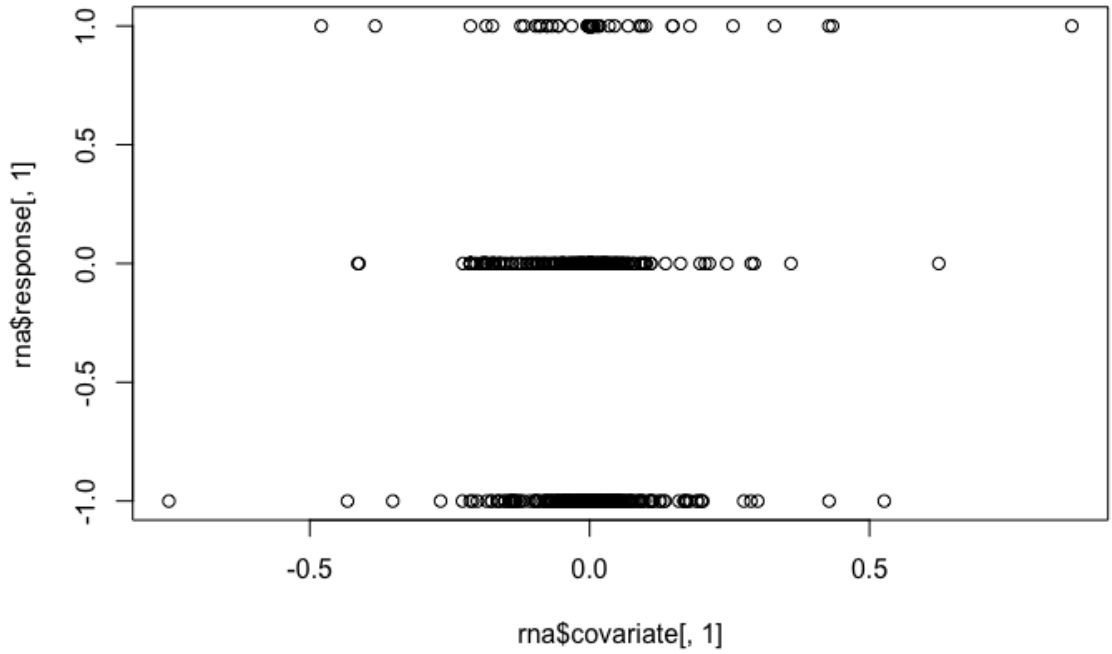


Figura 6.7: *RNA* com *FatigueLevel* e *Performance.Task* como *output*, com 7 níveis de escala



(a) *FatigueLevel*



(b) *Performance.Task*

Figura 6.8: Covariância dos resultados da *RNA* com *FatigueLevel* e *Performance.Task* como *output*, numa escala de 7 níveis

O melhor resultado com escala de 2 níveis é: uma rede com 7 nodos de *input* (*Performance.ADMSLMean*, *Performance.KDTMean*, *Performance.AEDMean*, *Performance.MVMean*, *Performance.MAMean*, *Performance.DMSMean* e *Performance.DDCMean*) e 3 camadas intermédias, com 4, 3 e 2 nodos, respetivamente (Figura 6.9). O teste obtido resultou num *rmse* em relação à Fadiga de $0,3881$ e em relação à Tarefa de $0,5859$, resultando num *rmse* médio de $0,4870$, e foi calculado em 57375 iterações.

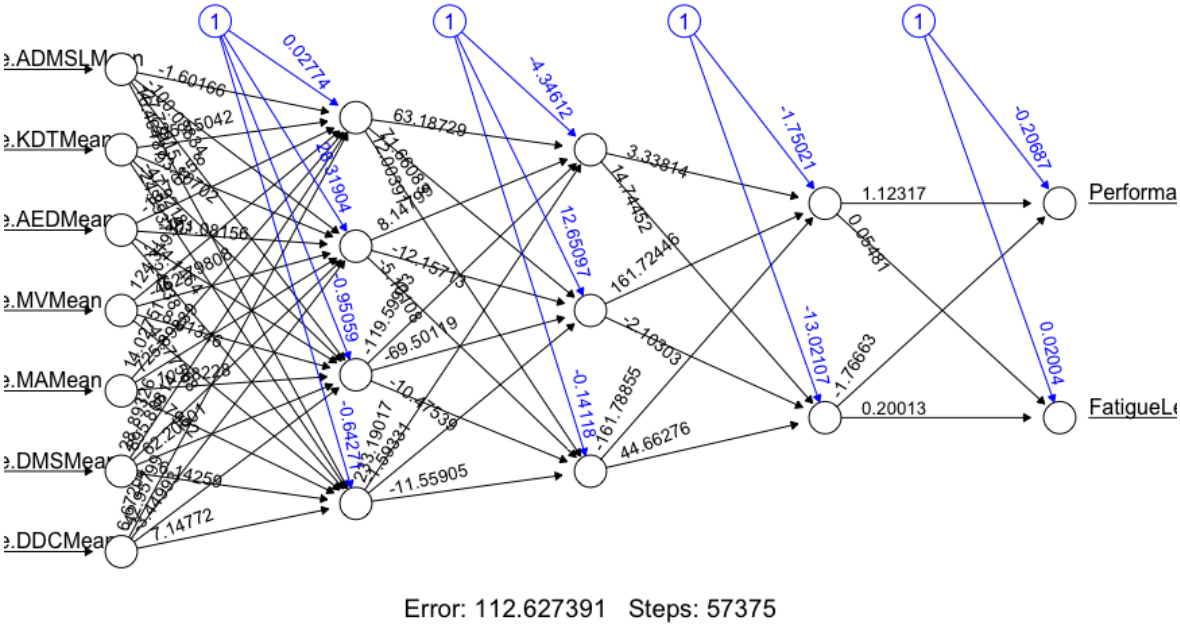
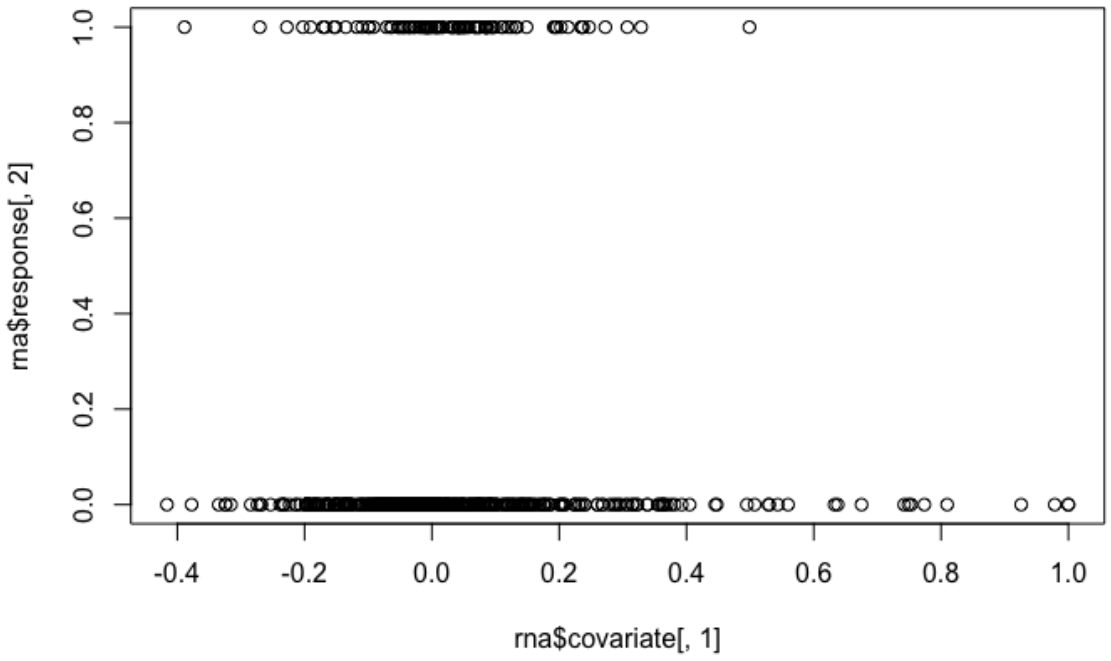


Figura 6.9: *RNA* com *FatigueLevel* e *Performance.Task* como *output*, com 2 níveis de escala

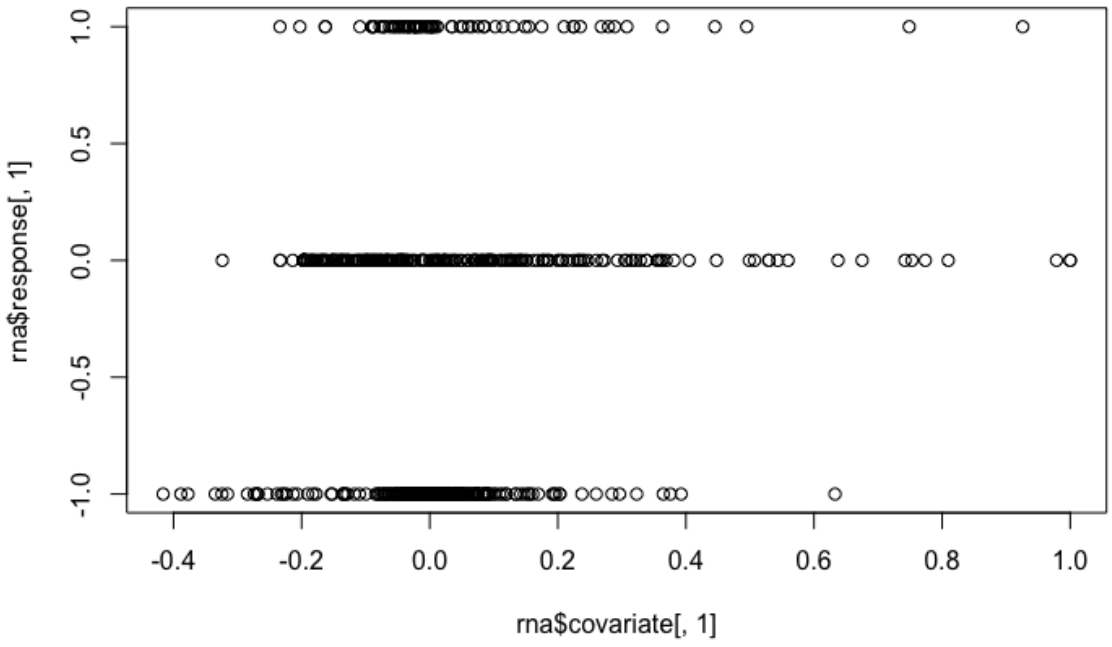
Pode-se concluir que a melhor *RNA* é a rede em que a exaustão vem expressa em 2 níveis. Tal já era esperado, uma vez que, para além de não existirem dados suficientes para a rede aprender sobre os níveis mais altos de Fadiga, é natural que, tendo uma escala bastante mais reduzida, a evolução e a aprendizagem seja mais fácil.

A média do *rmse* entre a melhor rede do *FatigueLevel* como output e a melhor rede da *Performance.Task* como output é igual a $\frac{0,3849+0,6320}{2} = 0,508$ que é superior ao *rmse* da melhor rede com dois *outputs*, que é igual a $0,4870$. Assim, conclui-se que é preferível calcular o nível de exaustão mental e a tarefa em execução de forma conjunta e não separada.

Em relação à dispersão dos resultados, pode-se ver que: quando a rede é treinada com dados em que o nível de exaustão está expresso numa escala de 7 níveis, a covariância dos resultados fica contida num intervalo de $[-0.5, 0.5]$, quer para o *FatigueLevel*, quer para o *Performance.Task*; quando a rede é treinada com dados em que o nível de exaustão está expresso numa escala de 7 níveis, a covariância dos resultados fica contida num intervalo de $[-0.4, 1.0]$, para ambos os *outputs*, sendo a dispersão superior neste caso.



(a) *FatigueLevel*



(b) *Performance.Task*

Figura 6.10: Covariância dos resultados da *RNA* com *FatigueLevel* e *Performance.Task* como *output*, numa escala de 2 níveis

Conclusão e aspetos a melhorar

Como já foi referido anteriormente ao longo deste relatório, o principal objetivo deste exercício prático é adquirir conhecimentos sobre o funcionamento da definição, treino e teste de *Redes Neuronais Artificiais*. Consideramos que esse objetivo foi cumprido, na medida que se percebemos a importância da normalização dos dados de treino/teste da rede, da escolha dos atributos (os mais significativos) e também da escolha da tipologia da rede.

Consideramos que este paradigma da representação de conhecimento não simbólico é importante de ser explorado e pode trazer vantagens na resolução de determinados problemas, comparativamente a outros paradigmas de programação, por exemplo.

Em termos de aspetos a melhorar, tem-se o número de testes a realizar e também a quantidade da dados referentes a níveis altos de fadiga. Apesar de compreensível que não seja possível de se obter dados quando o ser humano se encontra com altos níveis de exaustão, o baixo número e diversidade dos dados impede que a *RNA* aprenda como desejado. Deste modo, gostaríamos de ter tido mais tempo para efetuar mais testes e com mais variedade para que, assim, nos trouxessem resultados mais conclusivos.

Bibliografia

[1] CORTEZ, P., AND NEVES, J. Redes neuronais artificiais. *Universidade do Minho, Braga, Portugal* (2000).

[2] DA CRUZ, A. J. R. *Data mining via redes neuronais artificiais e máquinas de vectores de suporte*. PhD thesis, Universidade do Minho, 2007.

[3] PERELLI, L. P. Fatigue stressors in simulated long-duration flight. effects on performance, information processing, subjective fatigue, and physiological cost. Tech. rep., DTIC Document, 1980.