

**67660**

**67711**

**67724**

**67649**

**António Manuel Pereira do Anjo**

**Carlos Rafael Cruz Antunes**

**José Francisco Ferreira Alves de Sousa**

**Nuno André da Silva Oliveira**

**Universidade do Minho**

Escola de Engenharia

Licenciatura em Engenharia Informática

**Sistemas de Representação de Conhecimento e Raciocínio**

Ano Letivo de 2014/2015

**Relatório do Exercício 3**

# Índice

[1. Introdução 1](#_Toc419133125)

[2. Descrição das Tarefas 2](#_Toc419133126)

[2.1. Conhecimento Inicial 3](#_Toc419133127)

[2.2. Invariantes 4](#_Toc419133128)

[2.3. Manipulação do Conhecimento 5](#_Toc419133129)

[2.4. Jasper / Aplicação em Java 6](#_Toc419133130)

[3. Conclusões 9](#_Toc419133131)

1. Introdução

Neste relatório explicaremos como desenvolvemos a nossa script de R que nos permitiu resolver o problema proposto no exercício 3, que tem como objetivo utilizar, redes neuronais artificiais para processar grandes quantidades de informação.

Neste exercício foi-nos pedido que fosse encontrada uma rede neuronal artificial ideal que consiga prever o nível de fadiga de um utilizador, dados vários parâmetros. Para isto treinamos a rede previamente com uma amostra dada.

1. Descrição das Tarefas

Neste exercício trabalhamos com redes neuronais artificias, em R, para tentar calcular o nível de fadiga de um utilizador. Isto é conseguido devido ao treino prévio da rede neuronal, que é baseado num conjunto de dados iniciais. Depois da rede estar treinada, calcula-se então a fadiga do utilizador, recebendo alguns 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 *píxeis*) 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 *clicks* 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 *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 calculará a soma da distância entre dois eventos MOUSE\_UP e MOUSE\_DOWN consecutivos;

Estes parâmetros devem estar entre -1 e 1. Assim que os parâmetros sejam introduzidos a rede deve ser capaz de calcular uma aproximação da fadiga do utilizador, que varia de 1 a 7.

O nosso objetivo neste exercício é criar uma rede capaz de oferecer uma aproximação da fadiga, como explicado anteriormente.

* 1. Conhecimento Inicial

De forma a treinar a rede neuronal artificial para que esta possa ser usada para prever valores de fadiga tivemos que a treinar usando o *dataset* dado. Este *dateset* tem oito colunas relativas aos parâmetros de entrada indicados anteriormente uma coluna para o valor da fadiga, e uma última coluna para o tipo de tarefa que o utilizador estava a realizar na altura do teste.

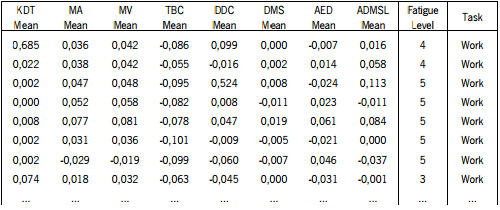


Tabela 1 - Excerto do *dataset* de treino

Verificamos que existem apenas três valores de fatiga iguais a 6 e nenhum com valor 7, logo decidimos remover esses valores da escala de fadiga, para que a rede neuronal seja mais precisa nos seus resultados.

Para além disto, treinamos também a rede neuronal utilizando apenas dois níveis de fadiga: 0 e 1, que correspondem simplesmente a não-cansado e cansado, respetivamente.

Dos 843 registos iniciais, foram usados apenas 600 para treinar a rede. Os restantes são usados para testa-la.

* 1. Testes

Ao longo deste exercício testamos redes com número de nodos e de camadas diferentes para encontrar o menor desvio padrão possível. Estas experiencias encontram-se na Tabela 2 e na Tabela 3. Foi medido o tempo (segundos) de execução de treino da rede, o EMSE (desvio padrão) e o erro da rede, quando possível. Estes parâmetros são melhores quanto menor for o seu valor. Para algumas tentativas não foi possível treinar a rede.

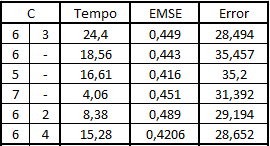


Tabela 3 - Resultados do treino de diferentes redes com 2 níveis de fadiga

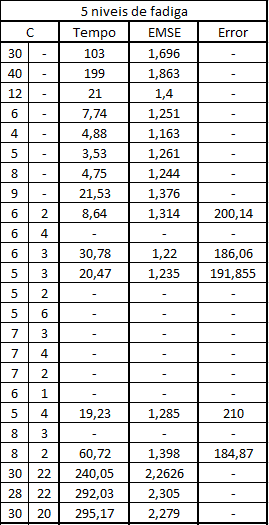


Tabela 2 - Resultados do treino de diferentes redes com 5 níveis de fadiga

Dentro das experiências realizadas, para ambas as redes treinadas com 2 e 5 níveis de fadiga o melhor resultado é uma rede com seis nodos na primeira camada e três nodos na segunda camada.

* 1. Rede Neuronal Artificial

Chegamos então à seguinte rede neuronal artificial:

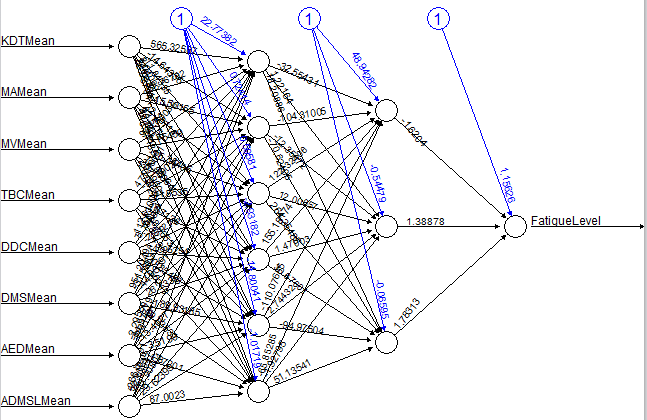


Figura 1 - Gráfico da rede neuronal artificial

* 1. Script em R

Para este exercício usamos duas bibliotecas do R:

library("neuralnet")

library(hydroGOF);

Lê-se o *dataset*:

dataset <- read.csv("C:\\file.csv", header=TRUE, sep=";", dec=".")

Extrai-se os 600 primeiros casos do *dataset* original para um novo *dataset* que será usado para treinar a rede neuronal.

trainset <- dataset[1:600, ]

Guarda-se o restante *dataset* num *dataset* de treino.

testset <- dataset[601:840, ]

Para a criação da rede neuronal definimos como variável objetivo a fatiga, e como variáveis de entrada as restantes. Criamos então uma rede 6,3 e é treinada com o *dataset* de treino.

rede<-neuralnet(FatigueLevel ~ KDTMean + MAMean + MVMean + TBCMean +

DDCMean + DMSMean + AEDMean + ADMSLMean,

trainset, hidden=c(6,3), threshold=0.01)

Com a rede criada corremos os testes na rede. Para isso definimos as variáveis de *input* para o teste.

temp\_test <- subset(testset,select =

c("KDTMean","MAMean","MVMean","TBCMean","DDCMean","DMSMean",

"AEDMean","ADMSLMean"))

Seguidamente submetemos a rede ao teste.

rede.results<-compute(rede,temp\_test)

Recolhe-se os resultados.

results <- data.frame(actual = testset$FatigueLevel, prediction = rede.results$net.result)

E calcula-se o RMSE.

rmse(c(testset$FatigueLevel),c(results$prediction))

1. Conclusões

Assim, através da implementação de conhecimento inicial e seguinte manipulação do mesmo utilizando várias técnicas e mecanismos de raciocínio adequados que nos permitiram trabalhar com conhecimento imperfeito num contexto de uma base de dados de automóveis, foi desenvolvido um sistema de representação de conhecimento e raciocínio com a capacidade de caraterizar um universo de comércio automóvel, assim como uma interface em Java que interaja com o sistema referido.