Universidade do Minho

3º Exercício

Mestrado Integrado em Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio

(2º Semestre / 2016-2017)

A74859 João da Cunha Coelho

A74601 José Miguel Ribeiro da Silva

A74748 Luís Miguel Moreira Fernandes

A73959 Pedro João Novais da Cunha

Braga,

Maio de 2017

Resumo

O presente documento tem como objetivo a documentação do terceiro exercício prático da unidade curricular de *Sistemas de Representação de Conhecimento e Raciocínio*. Seguindo o modelo dos dois primeiros exercícios práticos, depois de uma breve introdução, é detalhada a forma como o grupo de trabalho desenvolveu as várias funcionalidades propostas no enunciado do problema. Após a apresentação de alguns resultados, é realizada uma breve crítica ao trabalho realizado.

Índice

Introdução

O trabalho prático descrito neste relatório consiste na utilização de uma ***RNA*** (Rede Neuronal Artificial) com o objetivo de identificar diferentes níveis de fadiga de acordo com uma fórmula que utiliza como parâmetros métricas relativas à utilização do rato e do teclado de um computador.

A linguagem utilizada para desenvolver este trabalho será a linguagem ***R.***

Neste relatório são apresentadas todas as decisões tomadas pelo grupo e conclusões tiradas em relação aos resultados obtidos.

Preliminares

De forma a conseguirmos realizar o trabalho proposto foi necessário, através das aulas da Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio, adquirirmos conhecimentos base sobre *RNA*'s de forma a conhecermos a forma de utilizar o conhecimento que já possuímos sobre algo para obter conclusões sobre novos dados.

Foi ainda necessário conhecermos a linguagem de programação *R*.

Descrição do trabalho e análise dos resultados

1. Descrição Inicial do Trabalho Proposto

Como referido anteriormente este trabalho consiste na utilização de uma *RNA* para a identificação de diferentes níveis de fadiga tendo em conta certas métricas relativas à interação humano - computador. As métricas utilizadas neste estudo são as seguintes:

- **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 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;
- **Performance.Task:** identificação da tarefa que estava ser executada no momento da recolha dos dados.

Para possibilitar a realização deste trabalho foram-nos fornecidos dados respetivos a observações destas métricas e respetivo nível de fadiga. Os dados que recebemos contém 844 observações que podem ser usadas para o treino da nossa *RNA*.

2. Identificação dos Principais Atributos

O primeiro passo do trabalho foi decidir quais os atributos com maior relevância para o cálculo do *FatigueLevel*, da *Performance.Task*, ou destes em conjunto na mesma rede, tendo para isso sido utilizada a biblioteca *leaps* e executadas as seguintes instruções:

```
# Saber quais os 4 atributos de entrada mais importantes para o cálculo no nível de fadiga
regg1 <- regsubsets(FatigueLevel ~ Performance.KDTMean + Performance.MAMean + Performance.MVMean +
                    Performance.TBCMean + Performance.DDCMean + Performance.DMSMean + Performance.AEDMean +
                    Performance.ADMSLMean, exaustao7N, nvmax = 4)

# Saber quais os 4 atributos de entrada mais importantes para o cálculo da Task em exercício
regg2 <- regsubsets(Performance.Task ~ Performance.KDTMean + Performance.MAMean + Performance.MVMean +
                    Performance.TBCMean + Performance.DDCMean + Performance.DMSMean + Performance.AEDMean +
                    Performance.ADMSLMean, exaustao7N, nvmax = 4)

# Saber quais os 4 atributos de entrada mais importantes para uma rede com ambos os neurónios de saída
regg3 <- regsubsets(FatigueLevel + Performance.Task ~ Performance.KDTMean + Performance.MAMean + Performance.MVMean +
                    Performance.TBCMean + Performance.DDCMean + Performance.DMSMean + Performance.AEDMean +
                    Performance.ADMSLMean, exaustao7N, nvmax = 4)
```

Através das instruções em cima mostradas, obtiveram-se os seguintes resultados:

FatigueLevel	**Performance.Task**
Atributos mais significativos	
1º Performance.DDCMean	**1º** Performance.KDTMean
2º Performance.MAMean	**2º** Performance.DMSMean
3º Performance.MVMean	**3º** Performance.DDCMean
4º Performance.KDTMean	**4º** Performance.AEDMean

FatigueLevel + Performance.Task

1º Performance.MAMean

2º Performance.MVMean
3º Performance.DDCMean
4º Performance.ADMSLMean

3. Conversão de valores não inteiros

Após o estudo dos atributos mais significativos para cada tipo de output, foi necessário proceder à conversão dos valores da *Performance.Task* para *integers* uma vez que as redes neuronais não processam *Strings*, para isso foram usadas as seguintes instruções em R, executadas sobre o *dataset exaustao7N* anteriormente lido:

```
exaustao7N$Performance.Task[exaustao7N$Performance.Task == "Work"] <- 1
exaustao7N$Performance.Task[exaustao7N$Performance.Task == "office"] <- 2
exaustao7N$Performance.Task[exaustao7N$Performance.Task == "programming"] <- 3
exaustao7N$Performance.Task <- parse_integer(exaustao7N$Performance.Task)
```

4. Identificação dos Níveis de Fadiga

A proposta inicial, dada no enunciado do problema, referenciava uma escala com 7 níveis para a fadiga, sendo eles:

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.

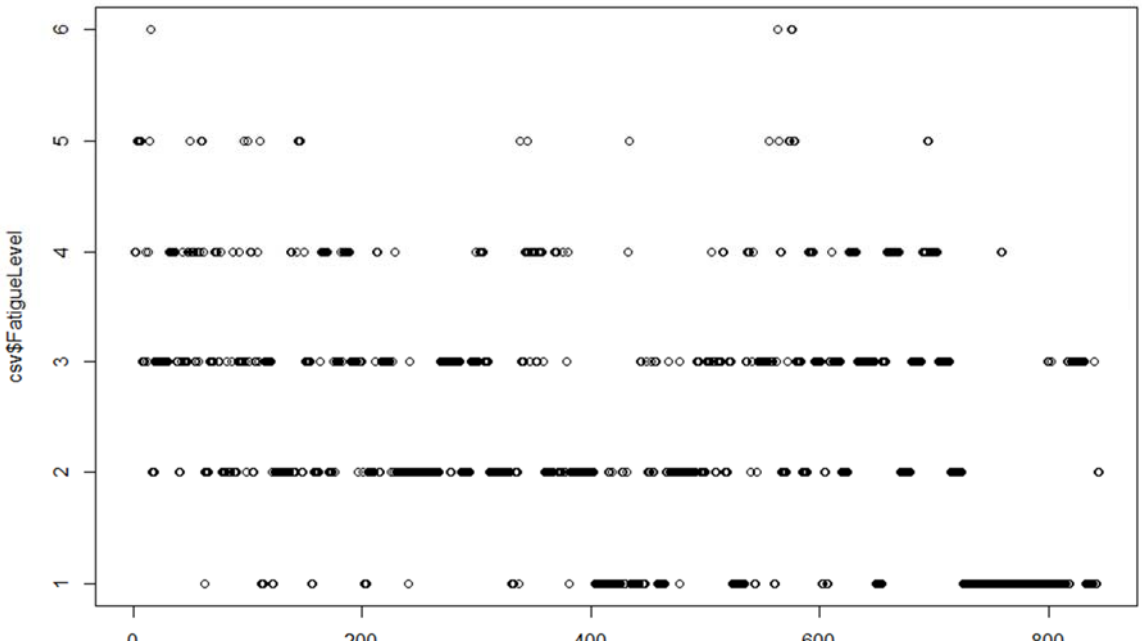
Foi então proposto que a rede neuronal conseguisse ainda dar resultados aos cenários em que a escala de fadiga consistia apenas na existência ou ausência de fadiga, bem como encontrar a melhor escala de identificação de exaustão.

Para o caso da existência ou ausência de fadiga foi criado um novo *dataset* com o nome *exaustaoBin*, processando o *dataset exaustao7N* através da função *binFat(dataset)*, da seguinte forma:

```r
# Conversão do dataset de input para 2 níveis de exaustão (ter ou não ter)
binFat <- function(dataset) {

    exaustaoBin <- dataset
    exaustaoBin$FatigueLevel[exaustaoBin$FatigueLevel <= 3] <- 0
    exaustaoBin$FatigueLevel[exaustaoBin$FatigueLevel > 3] <- 1

    return (exaustaoBin)
}

# Criação de um dataset com o nível de fadiga binário
exaustaoBin <- binFat(exaustao7N)
```

Da mesma forma foi criada uma função que fizesse a conversão para a melhor escala de medição de exaustão, *bestFat(dataset)*, que mais uma vez cria um novo *dataset* com o nome *exaustaoBest*, processando o *dataset exaustao7N*.

A melhor escala de medição de fadiga foi encontrada através da análise da distribuição dos dados do *dataset* original:



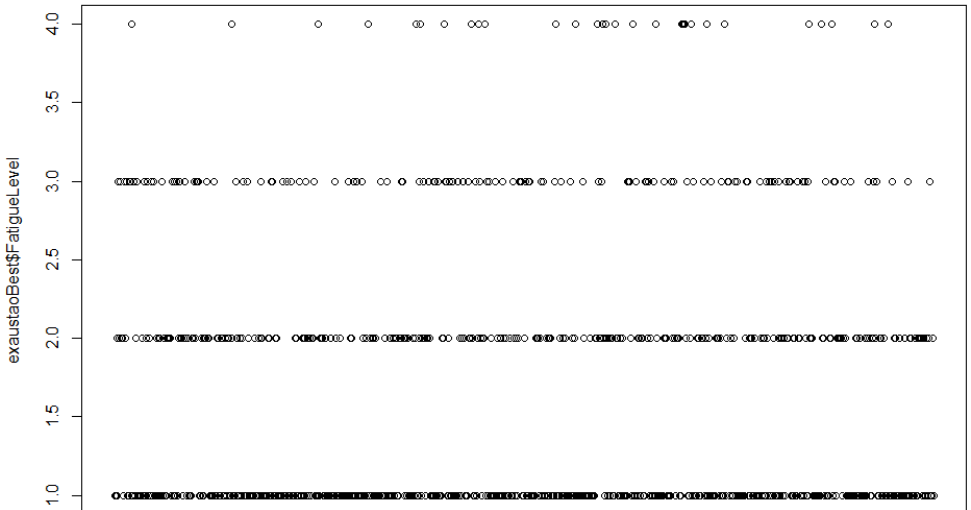
Foi então tomada a decisão de agrupar os níveis 1 e 2 num nível só, bem como os níveis 5, 6 e 7, deixando o nível 3 e o nível 4 como níveis separados tal como no *dataset* original produzindo o seguinte:

```
# Conversão do dataset para o melhor agrupamento de níveis de fadiga
bestFat <- function(dataset) {

    exaustaoBest <- dataset
    exaustaoBest$FatigueLevel[exaustaoBest$FatigueLevel < 3] <- 1
    exaustaoBest$FatigueLevel[exaustaoBest$FatigueLevel == 3] <- 2
    exaustaoBest$FatigueLevel[exaustaoBest$FatigueLevel == 4] <- 3
    exaustaoBest$FatigueLevel[exaustaoBest$FatigueLevel > 4] <- 4

  return(exaustaoBest);
}

# Criação de um dataset com o nível de fadiga na melhor escala
exaustaoBest <- bestFat(exaustao7N)
```

Desta forma há apenas 4 grandes níveis de fadiga, sendo que agora os dados estão mais agrupados, havendo por isso mais casos de treino para certos níveis que anteriormente não acontecia.

Como se pode constatar pela imagem da distribuição dos dados do nível de fadiga no *dataset* original, apesar de existir na escala o nível 7 – "Incapaz de funcionar, pronto a desligar", não existe nenhuma entrada em que o nível de fadiga seja o 7, desta forma a rede nunca irá aprender a responder este valor.

Com a melhor escala que o grupo identificou, os 4 grandes níveis de fadiga ficam um pouco melhor distribuídos, tornando assim a aprendizagem de cada um mais fácil e correta, produzindo assim o gráfico de distribuição:



5. Treino da Rede Neuronal

Após as considerações anteriormente mencionadas, foi então processada a função que produzirá o treino da rede neuronal artificial.

A função criada *neuralTrain()* recebe então como argumentos:

trainset - conjunto de dados com os quais a rede irá ser treinada;

hidden – quantidade de camadas intermédias e nº de neurónios em cada uma destas;

testset – conjunto de teste com os quais irá a rede ser testada;

A função pode ainda receber alguns argumentos opcionais, visto estes terem um valor *default*:

***algorithm* –** algoritmo a usar no treino da rede neuronal, que tem como *default* a opção "rprop+";

threshold – critério de pausa do treino da rede, 0.1 como *default*;

formula – modelo da rede, descreve o input e output da rede. A formula *default* é: **Performance.Task + FatigueLevel ~ Performance.MAMean + Performance.MVMean + Performance.DDCMean + Performance.ADMSLMean**

```
neuralTrain <- function(trainset, hidden, testset, algorithm = "rprop+", threshold = 0.1,
                        formula = Performance.Task + FatigueLevel ~ Performance.MAMean + Performance.MVMean +
                                                                    Performance.DDCMean + Performance.ADMSLMean)
```

Função neuralTrain() completa:

```r
neuralTrain <- function(trainset, hidden, testset, algorithm = "rprop+", threshold = 0.1,
                        formula = Performance.Task + FatigueLevel ~ Performance.MAMean + Performance.MVMean +
                                                                    Performance.DDCMean + Performance.ADMSLMean) {

    if (algorithm == "backprop") {
        nNet <- neuralnet(formula, trainset, algorithm = algorithm, learningrate = 0.1, hidden = hidden, threshold = threshold, linear.output = FALSE)
    }
    else {
        nNet <- neuralnet(formula, trainset, algorithm = algorithm, hidden = hidden, threshold = threshold, lifesign = "full")
    }

    aux <- strsplit(as.character(formula), "~")

    teste <- getSubset(aux, testset)

    nNet$results <- compute(nNet, teste)

    if (aux[2] == "FatigueLevel + Performance.Task" || aux[2] == "Performance.Task + FatigueLevel") {
        resultsT <- data.frame(actual = testset$Performance.Task, prediction = nNet$results$net.result)
        resultsF <- data.frame(actual = testset$FatigueLevel, prediction = nNet$results$net.result)
        nNet$prediction1 <- round(resultsT$prediction.1)
        nNet$prediction2 <- round(resultsT$prediction.2)
        print("FatigueLevel:")
        nNet$rmse1 <- rmse(c(resultsF$actual), c(nNet$prediction1))
        print(nNet$rmse1)
        print("Performance.Task:")
        nNet$rmse2 <- rmse(c(resultsT$actual), c(nNet$prediction2))
        print(nNet$rmse2)
    }
    else {
        if (aux[2] == "FatigueLevel") {
            results <- data.frame(actual = testset$FatigueLevel, prediction = nNet$results$net.result)
            nNet$prediction <- round(results$prediction)
            print("FatigueLevel:")
            nNet$rmse <- rmse(c(results$actual), c(nNet$prediction))
            print(nNet$rmse)
        }
        else {
            results <- data.frame(actual = testset$Performance.Task, prediction = nNet$results$net.result)
            nNet$prediction <- round(results$prediction)
            print("Performance.Task:")
            nNet$rmse <- rmse(c(results$actual), c(nNet$prediction))
            print(nNet$rmse)
        }
    }

    return (nNet)
}
```

A função utiliza ainda uma função auxiliar criada pelo grupo, a função *getSubset()*, definida da seguinte forma:

```r
# Função que retorna o subset de teste dependendo da fórmula escolhida
getSubset <- function(aux, testset) {

    if (aux[2] == "FatigueLevel") {
        teste01 <- subset(testset, select = c("Performance.DDCMean", "Performance.MAMean", "Performance.MVMean", "Performance.KDTMean"))
    }
    else {
        if (aux[2] == "Performance.Task") {
            teste01 <- subset(testset, select = c("Performance.KDTMean", "Performance.DMSMean", "Performance.DDCMean", "Performance.AEDMean"))
        }
        else {
            teste01 <- subset(testset, select = c("Performance.MAMean", "Performance.MVMean", "Performance.DDCMean", "Performance.ADMSLMean"))
        }
    }
    return (teste01)
}
```

6. *Trainset*'s e *Testset*'s

Foram então criados *trainset*'s e *testset*'s para servirem de treino e teste para as redes neuronais.

Temos então, para o caso da rede neuronal que irá atuar sobre o *dataset* *exaustao7N* (*dataset* original):

```
# vários trainsets para o dataset exaustao7N
trainset1a <- exaustao7N[1:100, ]
trainset1b <- exaustao7N[1:200, ]
trainset1c <- exaustao7N[1:400, ]
trainset1d <- exaustao7N[1:600, ]

# vários testsets para o dataset exaustao7N
testset1a <- exaustao7N[601:nrow(exaustao7N), ]
testset1b <- exaustao7N[601:751, ]
testset1c <- exaustao7N[601:651, ]
```

Para a rede que irá atuar sobre o *dataset* binário *exaustaoBin*, foram criados os seguintes:

```
# vários trainsets para o dataset2
trainset2a <- exaustaoBin[1:100, ]
trainset2b <- exaustaoBin[1:200, ]
trainset2c <- exaustaoBin[1:400, ]
trainset2d <- exaustaoBin[1:600, ]

# vários testsets para o dataset2
testset2a <- exaustaoBin[601:nrow(exaustaoBin), ]
testset2b <- exaustaoBin[601:751, ]
testset2c <- exaustaoBin[601:651, ]
```

Por fim, foram da mesma forma criados *trainset*'s e *testset*'s para a rede que irá atuar sobre o *dataset* com os 4 níveis de fadiga (melhor escala encontrada), *exaustaoBest*:

```
# vários trainsets para o dataset3
trainset3a <- exaustaoBest[1:100, ]
trainset3b <- exaustaoBest[1:200, ]
trainset3c <- exaustaoBest[1:400, ]
trainset3d <- exaustaoBest[1:600, ]

# vários testsets para o dataset3
testset3a <- exaustaoBest[601:nrow(exaustaoBest), ]
testset3b <- exaustaoBest[601:751, ]
testset3c <- exaustaoBest[601:651, ]
```

7. Análise de Resultados

Passamos agora à análise dos resultados obtidos, decidiu-se fazer variar apenas o algoritmo e o *hidden*, mantendo o mesmo *trainset* e *testset* para cada um dos treinos e testes realizados.

Começamos então por realizar testes sobre o *dataset* original *exaustao7N*, que tem os 7 níveis de fadiga.

FatigueLevel

	Algoritmos	Nº nodos Camada 1	Nº nodos Camada 2	Erro
1	rprop+	8	4	1.231
2	backprop	8	4	1.847
3	rprop-	8	4	1.256
4	rprop+	10	5	1.199
5	backprop	10	5	1.847
6	rprop-	10	5	1.262

Performance.Task

	Algoritmos	Nº nodos Camada 1	Nº nodos Camada 2	Erro
1	rprop+	8	4	0.733
2	backprop	8	4	1.073
3	rprop-	8	4	0.749
4	rprop+	10	5	0.789
5	backprop	10	5	1.073
6	rprop-	10	5	0.763

FatigueLevel + Performance.Task

	Algoritmos	Nº nodos Camada 1	Nº nodos Camada 2	Erro	
				FatigueLevel	Performance.Task
1	rprop+	8	4	1.175	0.687
2	backprop	8	4	1.847	1.073
3	rprop-	8	4	1.220	0.707
4	rprop+	10	5	1.231	0.710
5	backprop	10	5	1.847	1.073
6	rprop-	10	5	1.172	0.684

Analisando os dados recolhidos, podemos verificar que a rede que melhor responde aos testes realizados é a rede com os dois neurónios de output, seguindo o algoritmo rprop- e com 2 camadas intermédias, com 10 e 5 neurónios respetivamente.

De seguida foram realizados testes para o segundo *dataset* pedido, o *dataset* binário *exaustaoBin*:

FatigueLevel

	Algoritmos	Nº nodos Camada 1	Nº nodos Camada 2	Erro
1	rprop+	8	4	0.384
2	backprop	8	4	0.384
3	rprop-	8	4	0.384
4	rprop+	10	5	0.457
5	backprop	10	5	0.389
6	rprop-	10	5	0.470

Performance.Task

	Algoritmos	Nº nodos Camada 1	Nº nodos Camada 2	Erro
1	rprop+	8	4	0.668
2	backprop	8	4	1.073
3	rprop-	8	4	0.768
4	rprop+	10	5	0.760
5	backprop	10	5	1.073
6	rprop-	10	5	0.698

FatigueLevel + Performance.Task

	Algoritmos	Nº nodos Camada 1	Nº nodos Camada 2	Erro	
				FatigueLevel	Performance.Task
1	rprop+	8	4	0.384	0.704
2	backprop	8	4	0.384	1.073
3	rprop-	8	4	0.379	0.637
4	rprop+	10	5	0.415	0.861
5	backprop	10	5	0.384	1.073
6	rprop-	10	5	0.395	0.701

No caso do *dataset exaustaoBin*, podemos verificar que, mais uma vez, o melhor é a rede com dois neurónios de output, utilizando rprop- e *hidden* com (10, 5).

Por último foram realizados treinos e testes para o *dataset* com a melhor escala para o nível de fadiga, *exaustaoBest*:

FatigueLevel

	Algoritmos	Nº nodos Camada 1	Nº nodos Camada 2	Erro
1	rprop+	8	4	0.977
2	backprop	8	4	1.082
3	rprop-	8	4	0.995
4	rprop+	10	5	0.996
5	backprop	10	5	1.082
6	rprop-	10	5	1.053

Performance.Task

	Algoritmos	Nº nodos Camada 1	Nº nodos Camada 2	Erro
1	rprop+	8	4	0.747
2	backprop	8	4	1.073
3	rprop-	8	4	0.721
4	rprop+	10	5	0.730
5	backprop	10	5	1.073
6	rprop-	10	5	0.768

FatigueLevel + Performance.Task

	Algoritmos	Nº nodos Camada 1	Nº nodos Camada 2	Erro	
				FatigueLevel	Performance.Task
1	rprop+	8	4	0.903	0.687
2	backprop	8	4	1.083	1.073
3	rprop-	8	4	0.954	0.724
4	rprop+	10	5	1.000	0.695
5	backprop	10	5	1.083	1.073
6	rprop-	10	5	0.919	0.665

Para finalizar, foram então analisados os dados recolhidos utilizando o *dataset exaustaoBest* e podemos concluir que apesar de, mais uma vez, a rede mais correta ser a que apresenta dois neurónios de output, neste caso o ideal seria usar rprop+ com duas camadas intermédias com 8 e 4 neurónios respetivamente.

De salientar que muitas variáveis poderiam ter sido tomadas em consideração, por exemplo, foram utilizados apenas 3 algoritmos para testes quando existem 5 (decidiu-se deixar de fora os algoritmos “sag” e “slr”), o *hidden* pode tomar imensos valores nas camadas intermédias, bem como ter várias camadas intermédias e decidiu-se apenas usar 2 camadas intermédias e apenas testar com 8 e 4 ou 10 e 5 nodos.

Além destes referidos, também os *trainset*’s e *testset*’s poderiam variar, podendo encontrar *trainset*’s que permitissem às redes apresentar melhores resultados.

Conclusões e Sugestões

A realização deste projeto foi uma forma de interiorizarmos toda a matéria lecionada nas aulas desta Unidade Curricular sobre o tema de Redes Neuronais e aplicarmos os conhecimentos aprendidos anteriormente.

Não foi um trabalho de muito fácil realização dado tratar-se de um tema que dá aso a muitas possibilidades diferentes de resolução e a muita subjetividade, trazendo assim alguma confusão na altura da tomada de decisões, levando a que estas tenham sido encontradas maioritariamente por tentativa-erro.

Pessoalmente, consideramos todo o conceito das Redes Neuronais um conceito interessante e útil para os mais variados fins.

Referências

- “Data Analysis Using R”, César Analide, José Neves;

- “Sugestões para a Redacção de Relatórios Técnicos, Cesar Analide, Paulo Novais, José Neves.