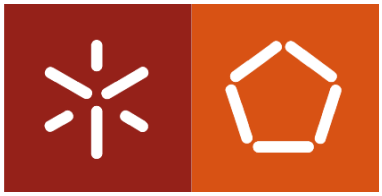


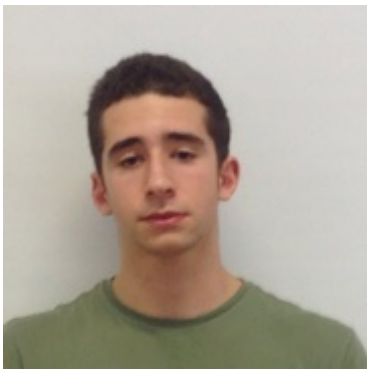
UNIVERSIDADE DO MINHO
Mestrado Integrado em Engenharia Informática
Sistemas de Representação de Conhecimento e Raciocínio

TRABALHO PRÁTICO 3

Conhecimento não simbólico: Redes Neuronais Artificiais.

Grupo 31:

Gonçalo Pereira : A74413



António Silva : A73827

André Diogo : A75505

Braga, 21 de Maio de 2017

Conteúdo

Introdução

O objetivo deste trabalho terá sido o de utilização de sistemas não simbólicos de representação de conhecimento (nomeadamente, Redes Neuronais Artificiais), para a análise de níveis e identificação de coeficientes de fadiga, de modo a estudar e averiguar a possibilidade de existência de exaustão num utilizador, consoante diversos fatores.

Como tal, e para que seja possível uma correta caracterização, foi feita uma normalização inicial dos dados fornecidos (os que ainda não estavam normalizados). Esta normalização deve-se ao facto de que os dados, devem, de modo a que a rede neuronal funcione corretamente, estar todos numa escala semelhante.

Deste modo, e após esta breve introdução, será feita, então, uma análise de como foi interpretado o problema em questão.

Estudo dos atributos mais significativos para a representação do conhecimento do problema em análise, com dados não normalizados

De modo a estudar os atributos mais significativos para representação do conhecimento, procedemos à utilização de várias das bibliotecas da plataforma `R Studio`, destacando as seguintes:

```
> library("neuralnet", lib.loc="~/R/win-library/3.3")

> library("hydroGOF", lib.loc="~/R/win-library/3.3")

> library("leaps", lib.loc="~/R/win-library/3.3")
```

De seguida, carregamos o ficheiro original para obtenção dos melhores parâmetros, associámos cerca de metade dos dados para treino e outra metade dos dados para teste.

```
> exaustao <- read.csv("./exaustao.csv")

> treino <- exaustao[1:444,]

> teste <- exaustao[445:844,]
```

Por fim, bastou apenas utilizar o comando `regsubsets` para verificar quais os atributos mais significativos (optámos por escolher apenas os três principais para futuras análises, visto que, como verificado nas aulas práticas, um número superior a esse não interfere no valor final):

```
> res <- regsubsets(ExhaustionLevel ~ Performance.KDTMean + Performance.MAMean +
Performance.MVMean + Performance.TBCMean + Performance.DDCMean + Performance.DMSMean +
Performance.AEDMean + Performance.ADMSLMean, exaustao, nvmax = 3)

         Performance.KDTMean Performance.MAMean Performance.MVMean Performance.TBCMean
1  ( 1 ) " "                 " "                " "                " "
2  ( 1 ) " "                 "*"                " "                " "
3  ( 1 ) " "                 "*"                "*"                " "
         Performance.DDCMean Performance.DMSMean Performance.AEDMean Performance.ADMSLMean
1  ( 1 ) "*"                 " "                 " "                " "
2  ( 1 ) "*"                 " "                 " "                " "
3  ( 1 ) "*"                 " "                 " "                " "
```

Obtendo, assim, como primeiros três valores, o `Performance.DDCMean`, o `Performance.MAMean` e o `Performance.MVMean`.

Identificação da(s) topologia(s) de rede mais adequada(s) e seleção das regras de aprendizagem para treinar a(s) rede(s), com dados não normalizados

Como já nos foi possível a averiguação dos atributos mais significativos, procedemos, então, à criação da rede neuronal adequada ao problema.

Optámos, desde já, por uma rede neuronal com 6, 4 e 2 nodos. É uma das tipologias que dados mais precisos dá (pois segue uma estrutura em pirâmide invertida, isto é, à medida que a rede se aproxima do nodo de saída, os nodos intermédios vão diminuíndo em número). Assim, executamos o seguinte comando no `R Studio`:

```
> rnacredito <- neuralnet(ExhaustionLevel ~ Performance.DDCMean +
Performance.MAMean + Performance.MVMean, treino, hidden = c(6,4,2),
lifesign = "full", linear.output = FALSE, threshold = 0.1)

hidden: 6, 4, 2    thresh: 0.1    rep: 1/1    steps:      30 error: 830.58472
time: 0.09 secs
```

Com a obtenção destes valores, suspeitámos da eventual não normalização de valores que poderiam ser necessários (como, por exemplo, o nível de exaustão, que estaria entre 1 e 7, quando poderia estar entre 0 e 1, ou até mesmo a não utilização de `Performance.Task` por não estar no formato numérico). Continuámos, porém, com a execução da rede neuronal, de modo a obter o `RMSE` (*Root-mean-square deviation*).

De seguida, selecionamos um subconjunto do conjunto de dados de teste considerando apenas os parâmetros `Performance.MAMean, Performance.MVMean e Performance.DDCMean` e utilizámos a rede neuronal anterior para calcular uma previsão do nível de exaustão. Por fim, arredondámos os valores da previsão para inteiros de modo a extrair o nível de exaustão previsto.

```
> teste.01 <- subset(teste, select = c("Performance.MAMean"
,"Performance.MVMean","Performance.DDCMean"))

> rnacredito.resultados <- compute(rnacredito, teste.01)

> resultados <- data.frame(atual = teste$ExhaustionLevel,
previsao = rnacredito.resultados$net.result)

> resultados$previsao <- round(resultados$previsao, digits=0)
```

Criadas então todas as condições para que pudéssemos proceder ao cálculo do `RMSE`, executámos o seguinte comando:

```
> rmse(c(teste$ExhaustionLevel),c(resultados$previsao))
[1] 1.756416807
```

Valor que, como suspeitado anteriormente, devido ao enormíssimo erro, nos levou a confirmar as nossas expectativas: os valores tabelados podem, ainda, ser mais normalizados. De igual modo, procedemos então a tal normalização.

Normalização dos dados iniciais

Como existiam duas tabelas com valores não normalizados (`ExhaustionLevel` e `Performance.Task`), procedemos, como tal, à normalização dos mesmos.

No que toca ao `ExhaustionLevel`, optámos pela transformação na fração mais adequada. Isto é, como temos sete níveis de exaustão de 1 a 7, dividimos tal numeração em sete frações, de modo a perfazer um intervalo de [0;1] com sete elementos. Assim, tal escala ficou do seguinte modo:

- 1 -> 0/6
- 2 -> 1/6
- 3 -> 2/6
- 4 -> 3/6
- 5 -> 4/6
- 6 -> 5/6
- 7 -> 6/6

Já em relação ao `Performance.Task`, avaliámos cada um dos valores subjetivamente. Existindo valores em formato `string` tais como `Work, Office, Programming`, decidimos que estes estavam por ordem decrescente de atividades propícias ao cansaço. Remetemos, então, à mesma lógica de criar um intervalo entre [0;1] para representar estes elementos. Assim:

- Programming -> 0/2
- Office -> 1/2
- Work -> 2/2

Como todos os valores já estarão normalizados, resta-nos agora a reformalização do estudo e nova tentativa de criação de rede neuronal.

Estudo dos atributos mais significativos para a representação do conhecimento do problema em análise, com dados normalizados

Com as bibliotecas já carregadas, efetuámos de seguida o carregamento dos novos dados (após "limpar"os dados previamente guardados no `R Studio`).

```
> exaustao <- read.csv("./exaustao_normalizado.csv")

> treino <- exaustao[1:444,]

> teste <- exaustao[445:844,]
```

De igual modo, executamos o comando `regsubsets` para averiguação dos atributos mais significativos, desta vez com o método `backward`, de modo a assinalar todos os atributos e qual o seu impacto final na rede:

```
> res <- regsubsets(ExhaustionLevel ~ Performance.KDTMean + Performance.MAMean
+ Performance.MVMean + Performance.TBCMean + Performance.DDCMean + Performance.DMSMean
+ Performance.AEDMean + Performance.ADMSLMean, exaustao, method = "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 ) "*"                 "*"                " "                "*"
         Performance.Task
1  ( 1 ) "*"
2  ( 1 ) "*"
3  ( 1 ) "*"
4  ( 1 ) "*"
5  ( 1 ) "*"
6  ( 1 ) "*"
7  ( 1 ) "*"
8  ( 1 ) "*"
```

Obtendo, neste caso, como primeiros três valores, o `Performance.Task`, o `Performance.MAMean` e o `Performance.MVMean`, valores diferentes dos iniciais.

Identificação da(s) topologia(s) de rede mais adequada(s) e seleção das regras de aprendizagem para treinar a(s) rede(s), com dados normalizados

Como já nos foi possível uma nova averiguação dos atributos mais significativos, procedemos, então, à nova criação da rede neuronal adequada ao problema.

Utilizámos, também, uma rede neuronal com 6, 4 e 2 nodos, como explicado anteriormente:

```
>  rnacredito <- neuralnet(ExhaustionLevel ~ Performance.MAMean + Performance.MVMean +
Performance.Task, treino, hidden = c(6,4,2),
lifesign = "full", linear.output = FALSE, threshold = 0.1)

hidden: 6, 4, 2    thresh: 0.1    rep: 1/1    steps:      11
error: 7.2625  time: 0.01 secs
```

Com estes novos valores, concluímos, desde já, que o erro é bastante inferior ao obtido inicialmente (mais de 100x abaixo do valor inicial), o que, de início, é já uma melhoria significativa.

Por fim, resta-nos então calcular o `RMSE` de novo:

```
> teste.01 <- subset(teste, select = c("Performance.MAMean"
,"Performance.MVMean","Performance.Task"))

> rnacredito.resultados <- compute(rnacredito, teste.01)

> resultados <- data.frame(atual = teste$ExhaustionLevel,
previsao = rnacredito.resultados$net.result)

> resultados$previsao <- round(resultados$previsao, digits=0)
```

Criadas então todas as condições para que pudéssemos proceder ao cálculo do `RMSE`, executámos o seguinte comando:

```
> rmse(c(teste$ExhaustionLevel),c(resultados$previsao))
[1] 0.3094574105
```

Assim, e finalmente, obtivémos um valor 6 vezes melhor do que o que fora obtido de ínicio.

Conclusão

Com a execução deste trabalho registou-se que a normalização dos dados para obtenção de resultados corretos é fundamental, especialmente para um volume de dados em grande escala, possibilitando a averiguação correta de resultados finais (ou, pelo menos, bastante aproximada ao valor que deveria ser correto). Foi também possível concluir que, em quase todos os casos, o tipo de tarefa a ser executada exerce uma grande influência no nível de exaustão, sendo que, como tabelado, os valores obtidos quando `Performance.Task` equivale a `Work`, apresentam um nível de exaustão bastante elevado.