Projeto 4 - Fundamentos em Sistemas Inteligentes

Renan Godoi de Medeiros
Matricula: 15/0146612

***Abstract*—O projeto consiste em treinar e testar dois modelos de Redes Neuronais Artificiais, uma MLP (Perceptron Multicamadas), e uma FBR, em uma base de dados indicativos de spam.**

I. Introdução

Este relatório tem por objetivo descrever o projeto da disciplina de Fundamentos em Sistemas Inteligentes e demonstrar as capacidades e limitações dos algoritmos de multicamadas e FBR no âmbito de Sistemas Inteligentes.

Os tópicos foram divididos em 6 partes, o tópico II explica de forma clara os problemas que acarretaram no desenvolvimento do projeto assim como as dificuldades encontradas no mesmo.O tópico IV explica a razão pela qual tentei resolver os problemas citados em II.

Ainda no tópico III eu descrevo os materiais utilizados para a realização do projeto, assim como os métodos que eu utilizei no desenvolvimento do mesmo, em seguida, no tópico V é apresentada de forma clara como o projeto foi desenvolvido e também como ele foi dividido dentro do código do software. No tópico VI descrevo os resultados encontrados após uma série de testes e pesquisas a respeito.E por fim em VII apresento as minhas conclusões a respeito do trabalho, assim como as experiências que este trabalho me proporcionou.

II. Problema

Os problemas que acarretaram durante o projeto foram vários, desde a projeção de como seria feito o software até a sua implementação, pelo fato de existirem poucas informações sobre a utilização de validação cruzada em conjunto do algoritmos de MLP e FBR, tive que fazer certas improvisações para realizar o trabalho da melhor maneira possível.

Tive problemas no processamento das informações do banco de dados do projeto, já que quando o abri para averiguar como estavam a disposição de dados da tabela, vi que os dados já estavam previamente separados em treino e teste(80% para treino e 20% para teste aproximadamente), logo para respeitar a especificação do que foi requisitado neste projeto,juntei as informações que haviam sido divididas e as dividi novamente através de um método para junção de tabelas, e a partir recalculei a parametrização da divisão de dados, e refiz a divisão de teste e treino dos dados fornecidos.

O desempenho dos algortimos foi algo que atrapalhou um pouco o desenvolvimento dos mesmos, o motivo se deve pelo grande problema de que para a validação cruzada, deveria-se usar vários modelos para a apuração de resultados dos algoritmos, e pelo fato da minha máquina não ser tão boa, somado ao fato da IDE demorar significativamente para processar os dados e dar um resultado, muitas vezes o console soltava um erro de compilação depois de tentar fazer o processamento de dados no cálculo dos acertos, e o tempo que isso levava era cerca de 20min até 1h, e consequentemente era difícil descobrir a origem dos erros ocasionados.Entretanto mais tarde, depois de várias horas perdidas, descobri que era o método de divisão de dados, que não estava dividindo os dados proporcionalmente para os fatores utilizados, contudo resolvi este problema criando uma gambiarra, utilizando a mesma proporção para os dois fatores e felizmente funcionou, apesar de tudo.

III. Materiais e Métodos

Produzi este software utilizando a linguagem R, pois as funções e métodos já são disponibilizados para a utilização.Fiz uma pesquisa assídua sobre o assunto em vários fórums, sites e artigos disponíveis na internet, tanto em inglês quanto português.Utilizei uma máquina com Windows para a realização do trabalho, utilizei também a IDE RStudio para facilitar a construção do software, e modelei o programa com a ferramenta de modelagem Asta.

Também fiz uso do github para garantir caso ouvesse algum problema na minha máquina, utilizei as bibliotecas RSNNS, ggplot2, R.matlab que dispõe das funções de MLP e FBR para classificação dos dados, além de funções para geração de matrizes de confusão, assim como leitura de arquivos .mat(ao qual era o formato do banco de dados do projeto).

Dividi como foi requisitado, 90% para treino e 10% para teste, apesar de não ter encontrado exemplos para me basear de validação cruzada relacionados a este algoritmo, consiguir fazer o projeto baseando-me em projetos anteriores que já haviam sido construidos.

IV. Porque

As razões pelas quais quis resolver este problema foi meu interesse na implementação do projeto, pois de certa forma inteligência artificial implementada na prática é algo que não é visto muito no curso, e se formos analisar, praticamente inteligência artificial é utilizada em muitas áreas da tecnologia da informação, desde pequenos sites, até coisas mais complexas, como as novas arquiteturas de placas gráficas fabricadas pela empresa Nvidia a qual possui uma inteligência artificial responsável por ajudar a renderizar os gráficos de jogos e melhorar a imersão dos jogadores.

A existência de várias soluções para o problema deste projeto, foi um dos fatores que me incentivou em minhas escolhas, pois os modelos de redes neurais MLP e FBR possui uma aplicabilidade imensa, logo é possível reunir muitas

soluções para diferentes linguagens, inclusive linguagens que não são necessáriamente voltadas a estatística.

O fator interessante que me intrigaram na construção deste projeto foi a utilização de redes neurais, algo que teoricamente melhora a precisão das classificações, pois é possível variar por exemplo a classificação de um nível de camada para outro, enquanto uma camada faz uma classificação do tipo Knn por exemplo outra faz do tipo floresta randômica, o que abre uma grande fronteira de possibilidades para implementações.

Apesar dos algoritmos FBR e MLP possuirem seus defeitos e talvez não ser a melhor escolha para a solução desse problema, já que existem muitos algoritmos capazes de resolver este problema de forma talvez até mais eficiente, a proposta deste trabalho além de mostrar qual das duas é a melhor opção para utilização, também é mostrar que a FBR e MLP podem ser úteis de diversas formas.

V. Implementação

Para a implementação, fiz um reuso da arquitetura do projeto anterior, porém com algumas mudanças em algumas funcionalidades.

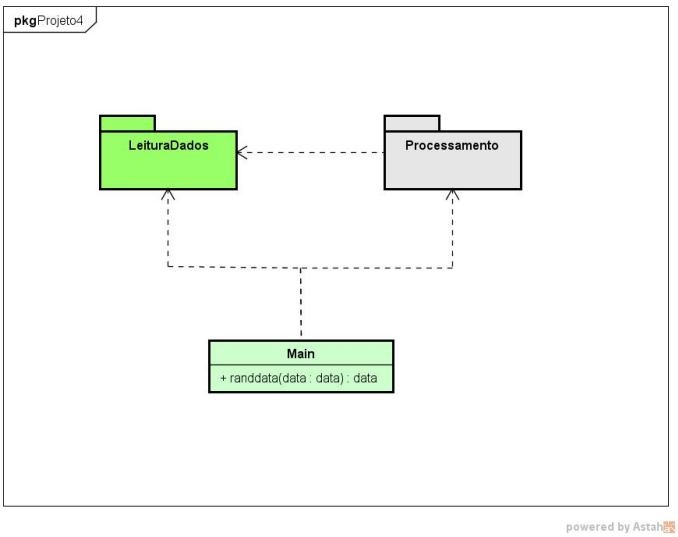

Fig. 1. Diagrama geral

A arquitetura deixei como estava no projeto anterior, a main depende do processamento e da leitura de dados, enquanto o processamento é dependente da leitura para que possa ser feita a análise de classificação do algoritmo.

Diferentemente dos projetos anteriores, a main agora possui um método *randdata(data)* que recebe um valor data como parâmetro, correspondente ao banco de dados, e retorna um valor data com dados separados em teste e treino que por sua vez foram processados randomicamicamente.

```
randdata = function(data){
  dP <- cbind(data$P.train,data$P.test)
  dT <- cbind(data$T.train,data$T.test)
  kdat <- list()
  dPv <- sample(ncol(dP), 0.9*ncol(dP), replace =
      TRUE)
  kdat$P.train <- dP[,dPv]
  kdat$P.test <- dP[,-dPv]
  kdat$T.train <- t(as.matrix(dT[,dPv]))
  kdat$T.test <- t(as.matrix(dT[,-dPv]))
  kdat
}
```

Código : Método de randomização de dados

o algoritmo não é muito complexo, apesar de ter sido um pouco difícil implementá-lo, eu utilizei o método *sample*, para fazer a divisão de dados, como especifiquei no problema em II, eu tive que juntar e dividir os valores novamente, pois os dados lidos pelo módulo de leitura já estavam separados, e retornei os dados com a divisão correta de treino e teste, utilizando a variável dPv como base.

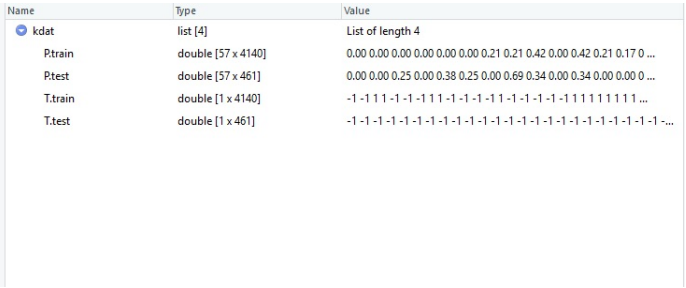

Fig. 2. Valores da estrutura de dados

Como pode ser avaliado na imagem 2 que representa a estrutura de dados de retorno do método kdata, a estrutura de dados data, é divida basicamente em 4 elementos principais, P.test, T.test, P.train, T.train, aos quais P e T se referem especificamente aos valores probabilísticos dos dados e os valores classificatórios.Train e Test como pode-se presumir, são os valores aos quais destinam-se a treino e a teste.

```
data = leitor$readfile()

cvalues <- 0:10

for (i in 0:11){
  rdata <- randdata(data)
  cvalues[i] = processor$ProcessarFBR(rdata
      ,40,1000)*100
}

cvaluesmlp <- 0:10

for (i in 0:11){
  rdata <- randdata(data)
  cvaluesmlp[i] = processor$ProcessarMLP(rdata
      ,40,1000)*100
}

plot(cvalues, type="o", col="blue",xlab = "modelos
    ",ylab = "Taxa de acerto %", main = "Radial
    Basis Function x Multi Layer Perceptron",ylim=
    c(0,80))

lines(cvaluesmlp, type="o", pch=22, lty=2, col="
    red")

legend(c(0,20), legend=c("RBF","MLP"),
       col=c("blue","red"), lty=1:2, cex=0.5)
```

Código : Main

A implementação da main ficou desta forma, com o leitor fazendo a leitura do arquivo e armazenando esta leitura em uma variável data, ao qual é utilizada durante o processamento dos valores nos loops, onde são feitas as validações cruzadas do algoritmo, são várias data diferentes sendo processadas pelo processador e com seus dados de retorno armazenados em dois arrays(cvalues e cvaluesmlp) aos quais serão utilizados para produzir o laudo de resultados da validação. O motivo pelo qual eu multipliquei por 100 o retorno dos valores de cada processo, é por causa que os valores retornados são valores decimais (0 1), e para ter uma visão mais clara de resultados, eu os multipliquei por 100. No final será construído um gráfico comparativo entre os modelos da MLP e da FBR, com as taxas de acerto que cada modelo conseguiu produzir, juntamente a relação de modelos com as taxas.

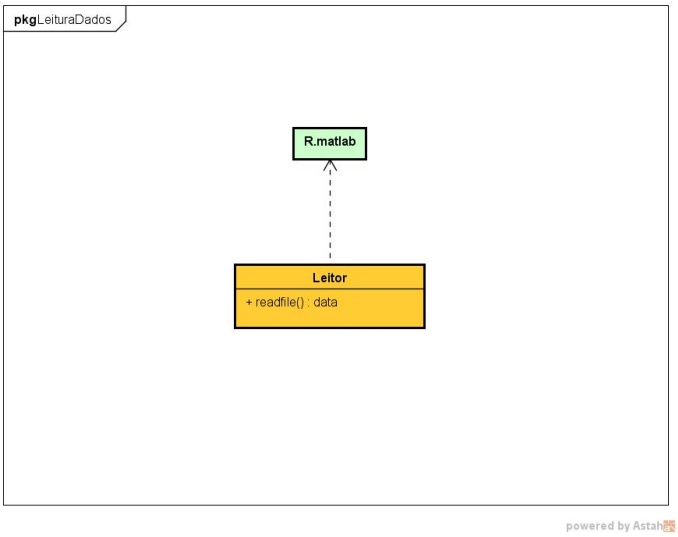

Fig. 3. Esquematização da leitura dos dados

Começando pela leitura representado na figura 3,assim como no projeto anterior o arquivo em formato .mat contendo as tabelas juntamente com os dados que serão lidos pelo leitor que fará a leitura de todos os dados contidos nas tabelas através da função *readfile()*, e retornará um valor do tipo dado, com os dados lidos.Porém diferentemente do projeto anterior, ele precisará da biblioteca R.mat ao qual conterá um método para realizar a leitura dos dados.

```
leitor <- list(
  readfile = function(){
    data <- readMat(file.choose())
    data
  }
)
```

Estrutura e código do módulo de leitura dos dados e seus métodos

O código do leitor como pode ser lido acima, não foi tão complexo de ser elaborado, diferentemente do código anterior, eu precisei apenas do método certo para a realização do método de leitura, sem muitos parâmetros ou alterações diferentemente do que já havia sido feito em projetos anterioes.

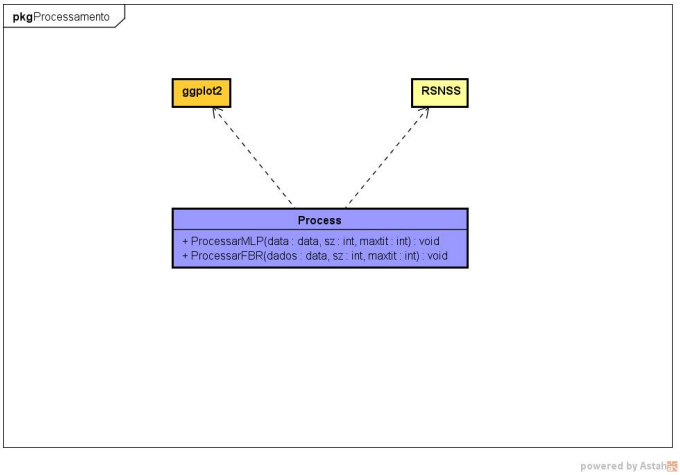

Fig. 4. Esquematização da processamento dos dados

Para o processamento dos dados na figura 4,ouve mudanças nas suas características eu descrevi duas operações para os quais o algoritmo faria, que foram *ProcessarFBR(dados,sz,maxtit)* basicamente responsável por processar o algoritmo das redes FBR ou RBF, mais conhecido como radial basis function e o método *ProcessarMLP(dados,sz,maxtit)*, que da mesma forma que a anterior é responsável por processar as redes neurais entretanto com o algoritmo de multu camadas de perceptrons, que também pode ser chamado de MLP em inglês que seria Multilayer perceptron que diferentemente do método anterior é caracterizado por utilizar várias camadas para realizar sua classificação.

Igualmente ao projeto anterior, este por sua vez herda 3 bibliotecas para a classe de processamento, aos quais possuem as funções *mlp* e *rbf* que foram essenciais para a construção do projeto.

```
processor <- list(
  ProcessarFBR = function(dados,sz,maxtit){
    yt <- t(as.matrix(dados$T.train))
    xt <- t(as.matrix(dados$P.train))

    modelo <- rbf(x = xt, y = yt, size = sz, maxit
        = maxtit, linOut = TRUE)
    predteste <- sign(predict(modelo, t(as.matrix(
        dados$P.test))))

    mean(predteste == t(dados$T.test))
  },
  ProcessarMLP = function(dados,sz,maxtit){
    #https://rdrr.io/cran/RSNNS/man/mlp.html

    yt <- t(as.matrix(dados$T.train))
    xt <- t(as.matrix(dados$P.train))

    modelo <- mlp(x = xt, y = yt, size = sz, maxit
        = maxtit, linOut = TRUE)
```

```
    predteste <- sign(predict(modelo, t(as.matrix(
        dados$P.test))))

    mean(predteste == t(dados$T.test))
  }
)
```

Estrutura e código do módulo de processamento das redes e seus métodos

A lógica funcional do código não é tão complexa, ela utiliza os parâmetros *P.train* e *T.train* como pode ser visto com detalhes na figura 2, para a realização das redes, como os métodos *rbf* e *mlp* só podem ser utilizados exclusivamente se as linhas das estruturas das matrizes forem iguais, logo se faz necessário equilibrar a quantidade de linhas tanto da estrutura classificatória como a probabilística, assim utilizei a transposta dessas estruturas convertidas em matrizes, para respeitar os parâmetros dos métodos.O retorno como pode ser observado, é feito pelo método mean que retorna o qual parecido são a previsão de teste do algoritmo, com os resultados esperados, e ele retorna esse valor(que corresponderia a taxa de acerto) para o chamador do método, ao qual seria a main no caso.

Eu tentei gerar uma rede neural, com neurônios controlados para fazer uma validação cruzada com mais congruência, entretanto, esses métodos aos quais utilizei para processar as redes, não dão a opção para fazer o controle dos níveis e neurônios da rede, e mesmo fazendo várias pesquisas, os métodos que encontrei que poderiam fazer esse tipo de coisa, infelizmente não tinham suporte para FBR e MLP no R, logo estive limitado ao que esses métodos me ofereciam, e tive que utilizar outros parâmetros como base para fazer a validação.

VI. Resultados

Os resultados me surpreenderam, foram totalmente inesperados, principalmente quando fiz a comparação entre os dois métodos de classificação utilizando redes neurais, a taxa de acerto de nenhum dos modelos testados chegaram a 100%, as médias foram razoavelmente boas, como pode ser observado na imagem da figura 5, onde é mostrado o comparativo do acerto entre dois modelos de redes neurais.

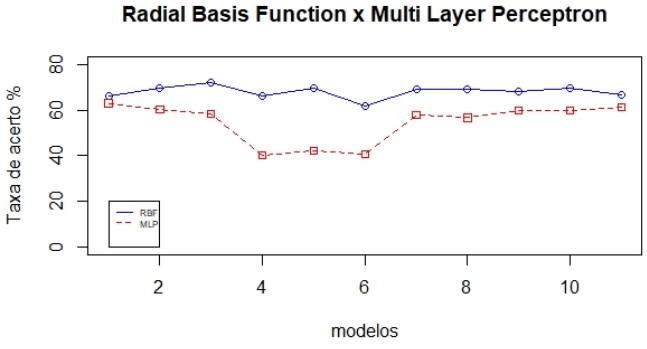

Fig. 5. Taxas de acerto FBR x MLP

Foram processados ao todo 10 modelos de redes neurais, sendo que 3 deles aparentemente ficaram abaixo da média esperada de acerto, é possível observar também que os modelos FBR fizeram a melhor classificação, a média de todos os modelos FBR no geral ficaram acima de 60%, entretanto as taxas dos modelos MLP ficaram bem abaixo da média, chegando a taxas de 40% de acerto, e para minha surpresa os modelos MLP que tiveram as maiores taxas beiraram perto das taxas mínimas de acerto dos modelos FBR, portanto é notório que a FBR teve um desempenho melhor comparado ao MLP.

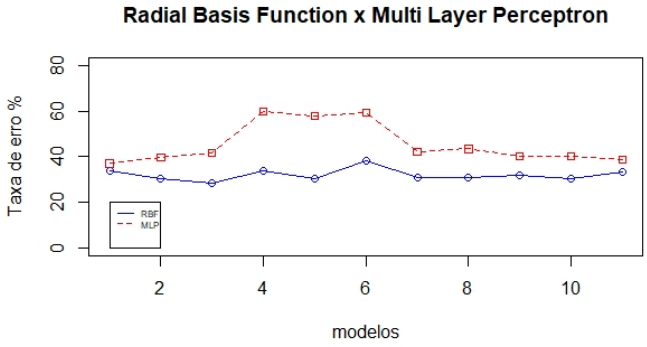

Fig. 6. Taxas de erro FBR x MLP

Na figura 6 são mostradas as taxas de erro dos modelos FBR e MLP, onde reforça ainda mais a discrepância esmagadora entre os dois modelos, as taxas de erro são bem maiores nos modelos MLP como era de se esperar visto as taxas de acerto apresentadas.

Eu fiz o cálculo da média de tempo de processo de um modelo, e por mais incrível que possa parecer, os resultados ainda foram a favor do modelo FBR, já que a média de tempo do processo foi de 1 minuto e 19 segundos, enquanto a média de tempo de processo da MLP foi de 2 minutos aproximadamente, oque deixa ainda mais claro a superioridade da FBR em relação a MLP.

VII. Conclusão

Neste trabalho abordei um assunto de grande importância em sistemas inteligentes, com o objetivo de mostrar o funcionamento dos algoritmo de redes neurais de Função de Base Radial e Perceptron de Multicamadas, analisando os resultados obtidos,comparando-os, analisando suas performances e se eram satisfatoriamente bons ou não.Também foi feita uma visão geral de como o projeto foi realizado,além dos softwares e IDE's que foram utilizados na sua construção.

Felizmente acredito eu, ter conseguido cumprir todos os requisitos do trabalho, apesar das dificuldades encontradas em relação aos algoritmos, foi possível a sua realização graças não somente ao material disponibilizado na plataforma moodle da disciplina, mas também aos fórums e blogs de pesquisa e ajuda em relação ao assunto.

Este trabalho foi muito importante para o meu conhecimento, aprendi várias coisas em relação a FBR e MLP .O

projeto levou bastante tempo para ser construido, mas apesar das dificuldades foi interessante, a àrea de inteligência artificial é muito vasta principalmente quando pensamos em sistemas complexos.

REFERENCES

[1] Gareth James, Daniela Witten, Trevor Hastle, Robert Tibshirani An Introduction to Statistical Learning with applications in R, 2014

[2] Machine Learning, Tom M. Michel, 1997

[3] Neural Networks: a comprehensive foundation, S. Haykin, 1994.

[4] Intelligent Systems, Negnevitsky, 2005