Universidade Federal do ABC

Bacharelado em Ciência da Computação

Camila do Amaral Sass

Jéssica Bibiano Lopes da Silva

Wedeueis Braz da Silva

Maquina de Aprendizado Extremo com Aprendizado Incremental

Santo André – SP

2015

Camila do Amaral Sass – 11051013

Jéssica Bibiano Lopes da Silva – 11066513

Wedeueis Braz da Silva – 11004813

Maquina de Aprendizado Extremo com Aprendizado Incremental

Relatório apresentado à Universidade Federal do ABC, como parte dos requisitos para aprovação na disciplina MC3311 – Inteligência Artificial do Curso de Bacharelado em Ciência da Computação.

Professora Doutora Debora Maria Rossi de Medeiros

Santo André – SP

2015

**1. Objetivo**

Este projeto tem como objetivo a comparação de métodos de aprendizagem de maquina a partir de redes neurais artificiais, utilizando um método que foi apresentado em aula e outro, até então desconhecido. O projeto nos deu a oportunidade de consolidar os conceitos teóricos aprendidos em aula, a oportunidade de aprender novos métodos e ter uma experiência pratica do assunto tratado.

**2. Introdução**

Em 1943 o neurofisiologista Warren McCulloch e o matemático Walter Pitts publicaram o artigo “A Logical Calculus of the Ideas Immanent in Nervous Activity”, no qual fizeram uma analogia entre células nervosas e o processo eletrônico, sendo os pioneiros no campo das redes neurais artificiais. No entanto o modelo apresentado por eles era simples, não conseguindo realizar uma de suas principais tarefas: o aprendizado.

Já em 1949, o biólogo e psicólogo Donald Hebb propôs um princípio de aprendizado em sistemas nervosos complexos, baseado na hipótese de que a força das sinapses no cérebro é alterada em resposta à experiência, sendo as conexões entre células ativadas ao mesmo tempo fortalecidas, enquanto as outras possuem tendência ao enfraquecimento. Tal hipótese foi decisiva para o desenvolvimento nessa área de pesquisa.

Em 1957 Frank Rosenblatt introduziu o Perceptron, rede neural de duas camadas de neurônios capaz de aprender através do ajuste de seus pesos, esse modelo representou uma nova abordagem para o reconhecimento de padrões, porém o entusiasmo gerado foi quebrado em 1969 com a publicação do livro “Perceptrons”, escrito por Minsky e Seymour Papert, onde demonstraram que embora o Perceptron fosse capaz de realizar operações booleanas (AND e OR), não era capaz de realizar operações básicas como XOR (ou-exclusivo), fato este que impactou negativamente o cenário da pesquisa na área.

Em 1982, John Hopfield reascendeu as pesquisas nessa área ao propor um modelo de rede com conexões recorrentes, baseada em um aprendizado não supervisionado com a competição dos neurônios, mas o que colocou a área de redes neurais em destaque novamente foi a teoria de redes neurais em multinível (MLP), treinadas com o algoritmo de aprendizado por retropropagação (Backpropagation) que se tornou conhecido após a publicação do livro Parallel Distributed Processing, editado por James L. McClelland e David E. Rumelhart.

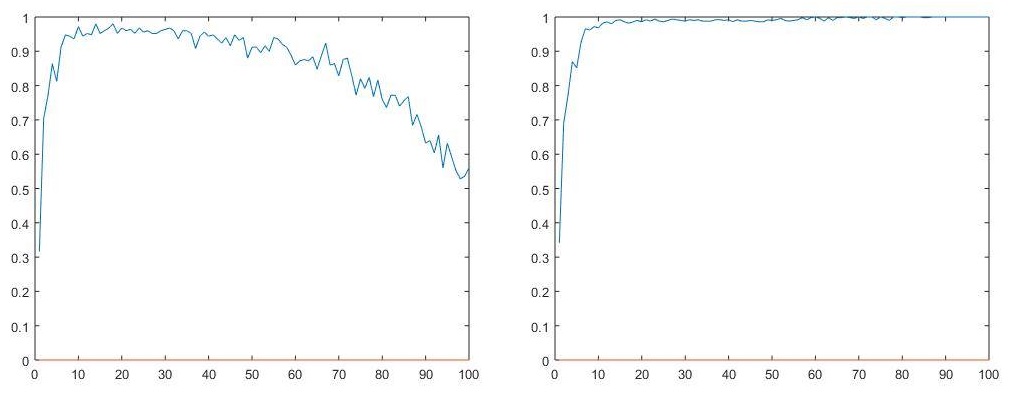
E são as redes em multinível o enfoque do nosso projeto, que abordará o método tradicional de MPL baseado na retropropagação, e também uma nova abordagem descrita no artigo “Universal Approximation Using Incremental Constructive Feedforward Networks With Random Hidden Nodes”, escrito por Guang-Bin Huang, Lei Chen, and Chee-Kheong Siew*.*

**3. Metodologia**

\*Os programas foram implementados em matlab.

Para testar a eficiência da rede implementada utilizamos três problemas de benchmark propostos e disponíveis no site <http://archive.ics.uci.edu/ml/>, que constituem três bases de dados, sendo duas utilizadas para classificação (Iris e Wine) e uma para regressão (Wine Quality). Tendo sido seus resultados comparados com o desempenho do Multilayer Perceptron, disponível nas ferramentas de redes neurais do Matlab, nesses mesmos problemas.

Antes de aplicar o algoritmo implementado foi necessário determinar a quantidade ótima de neurônios para cada rede e para cada problema, para isso utilizamos os programas *\_neuronios\_ELM.m*, *\_neuronios\_MLP.m* e *testeNumeroDeNeuronios.m*, procurando definir a quantidade de neurônios que maximizasse a acurácia nos problemas de classificação e minimizasse o erro quadrático médio no caso da regressão.

Figura

Para o problema de regressão os atributos da base de dados foram normalizados entre 0 – 1, para tornar os resultados mais intuitivos.

O programa principal encontra-se dividido em três subprogramas, cada um correspondente a uma base de dados: *iris\_mean.m*, *wine\_mean.m* e *wineq\_mean.m*. Cada um deles primeiro gera a base de dados para treinamento e teste utilizando os programas disponíveis nomeados com o sufixo \_data.m, os quais carregam os dados disponíveis para cada problema, geram uma permutação aleatória e os dividem em dois arquivos, que possuem as entradas (atributos) e saídas alvo, ficando 2/3 dos dados para treinamento e 1/3 para teste. Em seguida o programa chama a rede implementada em *elm.m* e o algoritmo *mlp.m,* que usa o Backpropagation disponível no Matlab, passando como parâmetro as bases de dados para treinamento e teste, o tipo de problema (regressão ou classificação) e as quantidades de neurônios da camada oculta. Após 10 repetições o programa calcula a média e o desvio padrão dos tempos de treinamento e teste, a média e desvio padrão da acurácia ou o erro quadrático médio e os imprime na tela.

**3.1. Descrição do algoritmo da rede *elm.m*:**

Primeiro o programa carrega os dados de entrada e saída alvo, disponíveis nas bases de dados passadas como parâmetro, e os armazena em variáveis específicas para treinamento e teste. Então, caso o parâmetro *tipo* não seja igual a 0 (correspondente a regressão), ele realiza um pré-processamento nos dados para verificar quantas classes diferentes existem e para codificar corretamente as saídas alvo. Em seguida é realizado o aprendizado com pesos aleatórios na camada oculta utilizando-se os dados da base de treinamento e medindo-se o tempo gasto. Caso o *tipo* seja regressão o programa já calcula o erro quadrático médio do treinamento. Em seguida os pesos são aplicados nos dados da base de teste, e novamente se *tipo* for igual à regressão calcula-se o erro quadrático médio do teste. Por fim, caso o *tipo* seja classificador, a rede calcula as taxas de acerto para o treino e o teste.

**3.2. Aprendizado na ELM:**

A rede inicialmente não possui nenhum neurônio na camada oculta, então iterarativamente adiciona-se neurônios na camada oculta definindo seus pesos e limiares aleatórios então os pesos de cada novo neurônio na camada oculta para a camada de saída são definidos utilizando-se a seguinte equação wi = (E\*H’)/(H\*H’) onde E é o vetor de erros em relação as saídas desejadas e H é o vetor de saídas da camada oculta que chegam na camada de saída. O processo é repetido até que o número de neurônios na camada oculta atinja um valor prédefinido ou até que o erro fique menor que o esperado.

**4. Resultados**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classificação** | **ELM** | | **MLP** | |
| **Iris** | **Wine** | **Iris** | **Wine** |
| média do tempo de treinamento | 0,398 | 0,356 | 0,516 | 0,697 |
| desvpad do tempo de treinamento | 0,031 | 0,187 | 0,188 | 0,323 |
| média do tempo de teste | 0,000 | 0,000 | 0,022 | 0,019 |
| desvpad do tempo de teste | 0,000 | 0,000 | 0,013 | 0,016 |
| média da acurácia no treinamento | 0,924 | 0,635 | 0,984 | 0,986 |
| desvpad da acurácia no treinamento | 0,071 | 0,120 | 0,010 | 0,010 |
| média da acurácia no teste | 0,906 | 0,605 | 0,966 | 0,938 |
| desvpad da acurácia no teste | 0,095 | 0,159 | 0,014 | 0,025 |

Para o problema de classificação dos dados da base Iris ambos os métodos tiveram resultados muito bons com o MLP tendo uma leve vantagem, com médias de acurácia de (92,4 ± 0,071)% no treinamento e (90,6 ± 0,095)% no teste para a ELM e (98,4 ± 0,010)% no treinamento e (96,6 ± 0,014)% no teste para o MLP, já para a base Wine o MLP teve uma grande vantagem tendo (98,6 ± 0,010)% no treinamento e (93,8 ± 0,025)% no teste contra (63,5 ± 0,120)% no treinamento e (60,5 ± 0,159)% no teste da ELM.

|  |  |  |
| --- | --- | --- |
| **Regressão** | **ELM** | **MLP** |
| **Wine Quality** | **Wine Quality** |
| média do tempo de treinamento | 0,494 | 1,144 |
| desvpad do tempo de treinamento | 0,046 | 0,159 |
| média do tempo de teste | 0,000 | 0,022 |
| desvpad do tempo de teste | 0,000 | 0,011 |
| erro quadrático médio do treinamento | 5,380 | 0,619 |
| desvpad do erro quadrático do treinamento | 0,916 | 0,010 |
| erro quadrático médio do teste | 5,445 | 0,682 |
| desvpad do erro quadrático do teste | 0,877 | 0,041 |

Para o problema de regressão utilizando a base de dados Wine Quality o MLP também teve uma grande vantagem, ficando 5,380 ± 0,916 no treinamento e 5,445 ± 0,877 no teste para a ELM e 0,619 ± 0,010 no treinamento e 0,682 ± 0,041 no teste para o MLP.

**5. Conclusão**

Levando em consideração que utilizamos uma rede MLP sofisticada, disponível nas ferramentas para redes neurais do Matlab, com mecanismos de regulação para evitar ou reduzir problemas clássicos do aprendizado como o overfiting, e também com encapsulamento na definição de parâmetros de aprendizado, a rede implementada ELM apresentou um desempenho razoável e muita velocidade no aprendizado, tornando-se uma alternativa mais simples e interessante para o aprendizado de máquina. Porém é importante ressaltar que o MLP ainda apresenta uma implementação muito mais robusta, que a da ELM analisada.

**6. Referências Bibliográficas**

[1] Guang-Bin Huang, Lei Chen, e Chee-Kheong Siew. *Universal Approximation Using Incremental Constructive Feedforward Networks With Random Hidden Nodes -* IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 17, NO. 4, JULY 2006