Rede Neural PMC como classificadora de padrões: uma aplicação ao problema de introdução de conservantes no processamento de bebidas

Gustavo Miranda de Aguiar
Uberlândia, Brasil
Email: gumiranda@ufu.br
Lara Carolina Luciana e Oliveira
Uberlandia, Brasil
Email: lara.carolinnaa@gmail.com
Marcelo Oliveira Costa Filho
Uberlandia, Brasil
Email: marceloocfilho@gmail.com
Natan Luis Silva Rodovalho
Uberlandia, Brasil
Email: natanlsr@gmail.com

***Abstract*—Redes Neurais Perceptron de múltiplas camadas (PMC) são vistas como um tipo de arquitetura bem flexível com relação a aplicação em vários domínios de conhecimento. A classificação de padrões, que consiste em classificar amostras dentro de um número de classes, é uma das áreas onde o uso redes neurais PMC é bastante comum. Este artigo tem como finalidade apresentar a aplicação de um perceptron de múltiplas camadas como um classificador de padrões. Para isso, será implementado um projeto prático proposto no livro "Redes Neurais Artificiais para Engenharia e Ciências Aplicadas", de Silva, Spatti e Flauzino (2010), o qual diz respeito a adição de conservantes no processamento de bebidas levando-se em consideração a presença de certas variáveis. Além do processo utilizado para o treinamento do PMC proposto, serão apresentados e analisados resultados considerados relevantes, obtidos após treinamento e testes na rede neural construída.**

I. Introdução

Redes Perceptron de multiplas camadas (PMC) constituem um tipo específico de redes neurais perceptron que se caracterizam pela presença de duas ou mais camadas de neurônios processadores. Tais camadas são divididas em camadas intermediárias ou ocultas e uma única camada de saída responsável pelo fornecimento da saída da rede. Há também a presença de uma camada de entrada para receber os sinais de entrada. As camadas se conectam através de links que contém pesos responsáveis pela intensidade dos sinais circulantes e o processo de aprendizagem consiste em reajustar os valores destes pesos até que algum critério seja atingido.

A arquitetura das redes PMC é do tipo feedforward pois o fluxo de informações parte da camada de entrada em direção à camada de saída. O treinamento é supervisionado (as saídas da rede são conhecidas) e realizado segundo o algoritmo backpropagation, o qual será detalhadamente abordado neste trabalho.

Redes PMC são eficientes para operar em diversas áreas de conhecimento, como classificação de padrões, aproximação de funções e previsão de séries temporais. No presente estudo, elas serão aplicadas como classificadoras de padrões a fim de implementar um projeto proposto no livro "Redes Neurais Artificiais para Engenharia e Ciências Aplicadas", de Silva, Spatti e Flauzino (2010). Este artigo tem como objetivo apresentar as metodologias necessárias para o desenvolvimento do referido projeto, bem como a descrição dos resultados obtidos.

II. Apresentação do Problema

Segundo SILVA, SPATTI e FLAUZINO (2010), laboratórios de processamento de bebidas realizam ensaios com o objetivo de identificar qual tipo de conservante deve ser adicionado a uma bebida específica. Este processo é norteado pelos valores de quatro variáveis presentes na bebida: teor de água, grau de acidez, temperatura e tensão interfacial. Dependendo de tais valores, é escolhido um tipo de conservante, A, B ou C, a ser introduzido na bebida.

Diante disso, os autores propõem a implementação de uma rede neural PMC com o objetivo de determinar qual conservante deve ser adicionado a um certo lote de bebidas considerando as quatro variáveis descritas como as entradas da rede.

III. Metodologia

A arquitetura descrita pela figura 1, também apresentada no livro, foi utilizada a fim de implementar uma rede PMC para solucionar o problema proposto.

Nota-se que a rede neural implementada possui 4 sinais na camada de entrada, todas assumindo valores reais, que correspondem as 4 variáveis: teor de água (x1), grau de acidez (x2), temperatura (x3) e tensão interfacial (x4). A arquitetura

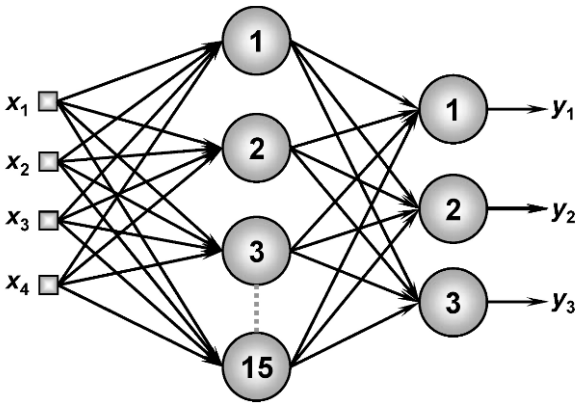

Fig. 1: Arquitetura da Rede

descreve duas camadas processadoras, sendo uma delas uma camada oculta contendo 15 neurônios e a outra uma camada de saída com 3 neurônios. As saídas, representadas por y1, y2 e y3, determinam o tipo de conservante obtido pela rede como resposta. A relação entre as saídas dos neurônios da camada de saída e o tipo de conservante é apresentada pela tabela 1.

Conservante	y1	y2	y3
Tipo A	1	0	0
Tipo B	0	1	0
Tipo A	0	0	1

TABLE I: Padronização das saídas da rede

É importante ressaltar que a arquitetura utilizada é flexível diante de variações na definição do problema. Caso o número de conservantes considerados seja maior, basta adicionar o número equivalente de neurônios de saída e seguir a mesma estratégia de padronização das saídas.

IV. Backpropagation

Para que a rede neural proposta possa fornecer as saídas corretas é necessário submetê-la a um processo de treinamento que consiste em sucessivos reajustes nos valores dos pesos dos links da rede. Em redes neurais PMC tal processo é norteado pela estratégia *backpropagation*.

De maneira resumida, as saídas da rede são calculadas utilizando um vetor de pesos iniciado aleatoriamente, produzindo os erros dos neurônios de saída (obtidos através da diferença entre as saidas esperadas e as saídas produzidas). Tais erros serão usados para reajustar os pesos dos links que ligam a camada oculta à camada de saída. Na sequência, são ajustados os pesos dos links que ligam a camada de entrada à camada oculta. Esse processo é realizado para cada amostra de treinamento (x1, x2, x3, x4) e suas respectivas saídas desejadas representadas no arquivo de treinamento por (d1, d2, d3).

Quando a última amostra é processada finaliza-se uma época. O conceito de época diz respeito ao processamento do conjunto total de amostras de treinamento. Assim que uma época termina é realizado o cálculo do erro quadrático médio, importante para a definição do critério de parada do algoritmo. Calculado o erro quadrático médio de uma época, o critério de parada do algoritmo é verificado.

Detalhes da implementação de todos esses passos, incluindo os processos de reajuste de pesos, obtenção das saídas da rede e cálculo do erro quadratico médio e outros aspectos relacionados ao treinamento da rede, são apresentados nas próximas seções.

V. Implementação

A rede neural PMC em estudo foi implementada em linguagem Java, utilizando os arquivos de treinamento e de teste disponibilizados em SILVA, SPATTI e FLAUZINO (2010) . O código fonte relativo a esta implementação, bem como os arquivos de dados utilizados, não estão incluídos neste artigo por motivos práticos, porém podem ser acessados em : https://github.com/laracarolina/PerceptronMultiplasCamadas. O objetivo das próximas seções é fornecer informações de alto nível relativas ao processo adotado para resolver o problema proposto.

Foram utilizadas matrizes para representar os vetores de pesos das camadas processadoras. Há matrizes separadas para armazenar os pesos dos neurônios da camada oculta e os pesos dos neurônios da camada de saída. Os valores atribuídos à taxa de aprendizagem (utilizada na fórmula de reajuste de pesos) e a precisão (relacionada ao critério de parada do processo de treinamento) são, respectivamente, 0.1 e 0.000001. Foi incluído um viés de entrada com valor igual a -1.

A rede foi treinada de duas formas: com a inclusão do parâmetro momento na fórmula de reajuste de pesos e sem a inclusão deste parâmetro. Com isso, foi possível verificar o impacto que este parâmetro pôde gerar no processo de treinamento. Para o parâmetro momento foi atribuído o valor 0.9.

A função de ativação utilizada para a implementação foi a função logística, cuja aplicação resulta em valores entre 0 e 1. Tal função é definida na figura 2.

$$g(u) = \frac{1}{1 + e^{-\beta u}}$$

Fig. 2: Função de Ativação Logística

Para que as saídas fossem 0 ou 1, como padronizado, foi utilizado o seguinte critério:

- Caso g(u) seja igual ou maior que 0.5, a saída produzida é 1.
- Caso g(u) seja menor que 0.5, a saída produzida é 0.

VI. Treinamento da Rede

Após iniciar os vetores de pesos da camada oculta e da camada de saída com valores aleatórios entre 0 e 1, o processo de treinamento da rede é iniciado. Os passos utilizados para a implementação do treinamento são descritos a seguir. Em seguida, será apresentado todo processo adotado para a execução de tais passos.

1) Iniciar os vetores de pesos com valores aleatórios;
2) Calcular o erro quadrático médio;
3) Para cada amostra (x1, x2, x3, x4) associada as saidas desejadas para os neurônios de saída (d1, d2, d3), repetir as instruções:
   - Obter os potenciais de ativação dos neurônios da camada oculta;
   - Obter as saídas produzidas pelos neurônios da camada oculta;
   - Obter os potenciais de ativação dos neurônios da camada de saída;
   - Obter as saídas produzidas pelos neurônios da camada de saída;
   - Calcular os gradientes dos neurônios da camada de saída;
   - Ajustar os pesos dos neurônios da camada de saída;
   - Calcular os gradientes dos neurônios da camada oculta;
   - Ajustar os pesos dos neurônios da camada oculta.
4) Calcular o erro quadrático médio;
5) Verificar critério de parada. Caso não seja atendido, voltar para 3. Caso seja atendido, encerrar o treinamento.

A. *Cálculo do erro quadrático médio*

O primeiro passo do algoritmo de treinamento implementado neste trabalho consiste em calcular o erro quadrático médio considerando os vetores de pesos iniciados aleatoriamente. Ele é obtido efetuando um somatório de parcelas, que correspondem ao erro de cada amostra.

O valor obtido é dividido pelo número total de amostras. No caso deste trabalho, o erro de uma amostra é o somatório das diferenças (elevadas ao quadrado) entre as saídas esperadas e as saídas desejadas para cada neurônio de saída. Para isso, anteriormente devem ser calculadas as saídas da rede para tal amostra, cujos cálculos são detalhados adiante.

B. *Processamento de uma época*

O próximo passo do treinamento é iniciar uma época. O contador utilizado para guardar o número de épocas durante o treinamento é incrementado e a primeira amostra a ser processada é obtida através da leitura da primeira linha do arquivo de treinamento. Tal linha contém, além dos valores de x1, x2, x3 e x4, as saídas desejadas para os neurônios de saída: d1, d2, d3.

Obtida a amostra e seus respectivos valores de saída desejados, as saídas da rede são calculadas e os pesos são ajustados (tais passos serão detalhados a seguir). A próxima amostra é obtida, repetindo-se o ciclo, até que todas as amostras tenham sido processadas. Ao final da época, o critério de parada é verificado.

C. *Cálculo das saídas da rede*

A cada amostra apresentada à rede, são calculadas as saídas dos neurônios da camada de saída, necessárias ao processo de reajuste de pesos. Este processo corresponde à etapa *feedforward* do algoritmo backpropagation: primeiramente são calculadas as saídas dos neurônios da camada oculta e, posteriormente, as saídas dos neurônios da camada de saída. Para calcular as saídas dos neurônios ocultos deve-se obter o potencial de ativação destes neurônios.

No caso deste trabalho, o potencial de ativação de um neurônio é o somatório das multiplicações de cada valor do vetor de entradas pelo peso correspondente no vetor de pesos do neurônio (o vetor de pesos de um neurônio é uma linha da matriz de pesos de uma camada).

Os potenciais de ativação são armazenados em um vetor. A função de ativação logística é aplicada a cada valor deste vetor, gerando o vetor de saídas da camada oculta. O cálculo das saídas da camada de saída é realizado de maneira análoga, calculando primeiro os potenciais de ativação de cada neurônio e aplicando a função logística para a obtenção das saídas.

D. *Reajuste de Pesos*

A cada amostra apresentada à rede, são calculadas as saídas da rede e em seguida realizam-se os reajustes dos pesos, que constituem a etapa *backward* do algoritmo backpropagation. Este processo é baseado na Regra Delta Generalizada, a qual define as fórmulas de reajuste apresentadas adiante. Primeiramente são calculados os reajustes de pesos para os pesos da camada de saída. Tais reajustes são guiados pelos erros atribuídos a cada neurônio de saída, calculados através da diferença entre a saída esperada para o neurônio e a saída obtida para o mesmo.

O processo de reajuste de pesos requer a obtenção dos gradientes dos neurônios. O gradiente de um neurônio diz respeito ao impacto produzido no erro global da rede em função de uma variação no potencial de ativação deste neurônio. As figuras 3 e 4 apresentam as fórmulas definidas para o cálculo dos gradientes dos neurônios da camada de saída e da camada oculta, respectivamente.

$$\delta_j(n) = \frac{\partial E(n)}{\partial x_j(n)} = -e_j(n)\varphi'_j(x_j(n))$$

Fig. 3: Gradiente de um neurônio da camada de saída. (Erro do neurônio multiplicado pela derivada da função de ativação aplicada ao potencial de ativação do neurônio).

$$\delta_l(n) = \varphi'_l(x_l(n)).\sum_{j=1}^{J}(\delta_j(n).\omega_{lj}(n))$$

Fig. 4: Gradiente de um neurônio da camada oculta. (Derivada da função de ativação aplicada ao potencial de ativação do neurônio multiplicada pelo somatório cujas parcelas envolvem a multiplicação do gradiente de cada neurônio da próxima camada pelo peso do link que chega a este neurônio).

Após o cálculo dos gradientes dos neurônios de saída seus pesos são reajustados confome figura 5 e após o cálculo dos gradientes dos neurônios ocultos seus pesos são reajustados conforme figura 6.

$$\Delta\omega_{lj}(n) = -\gamma.\delta_j(n).u_l(n)$$

Fig. 5: Reajuste de um peso de um neurônio de saída. (Multiplicação entre taxa de aprendizagem, gradiente do neurônio e sinal circulante no link).

$$\Delta\omega_{kl}(n) = -\gamma\delta_l(n).O_k(n)$$

Fig. 6: Reajuste de um peso de um neurônio oculto. (Multiplicação entre taxa de aprendizagem, gradiente do neurônio e sinal circulante no link).

Todas as fórmulas foram utilizadas no código de implementação da rede, disponível em : *https://github.com/laracarolina/PerceptronMultiplasCamadas*, utilizando as matrizes de pesos e os vetores contendo as entradas, as saídas desejadas e produzidas.

Foi incorporado ao processo de reajuste de pesos o parâmetro momento, o qual permite o aumento da taxa de aprendizagem sem que a estabilidade da rede seja comprometida. Ao ajustar um peso no processamento de uma amostra, tal parâmetro considera o valor do reajuste do peso no processamento da amostra anterior. O valor de reajuste desse peso será reforçado caso este valor tenha o mesmo sinal do valor anterior de reajuste.

E. Critério de Parada

O critério de parada adotado para o algoritmo de treinamento relaciona o erro quadrático médio de duas épocas sucessivas. Caso a diferença entre o erro quadrático de uma época e o erro quadrático médio da época anterior seja menor que um valor pequeno, denominado precisão, o treinamento da rede é encerrado. Caso tal critério não seja atendido uma nova época é iniciada. Como mencionado anteriormente, a precisão adotada foi de 0.00001.

VII. Resultados

Nesta seção serão apresentados todos os resultados obtidos após a execução de testes na rede, incluindo tempo de treinamento, número de épocas e desempenho em função do número de amostras classificadas corretamente. A fim de comprovar o impacto positivo provocado pela inclusão do parâmetro momento nos reajustes de pesos, seão apresentados separadamente os resultados obtidos com e sem a presença deste parâmetro.

A. Teste da Rede

Para comprovar o sucesso do processo de treinamento empregado, o arquivo com as amostras de treinamento foi utilizado para verificar se, ao processar uma amostra, as saídas produzidas pela rede após o treinamento coincidem com as saídas desejadas.

1) Parâmetro momento ausente: A tabela II mostra as saídas desejadas para os 3 neurônios de saída (representadas por d1, d2 e d3) para 20 amostras e as saídas obtidas (representadas por y1, y2 e y3).

Amostra	d1	y1	d2	y2	d3	y3
1	1	1	0	0	0	0
2	1	1	0	0	0	0
3	0	0	1	1	0	0
4	1	1	0	0	0	0
5	0	0	1	1	0	0
6	0	0	0	0	1	1
7	0	0	1	1	0	0
8	0	0	0	0	1	1
9	1	1	0	0	0	0
10	0	0	1	1	0	0
11	1	1	0	0	0	0
12	1	1	0	0	0	0
13	1	1	0	0	0	0
14	0	0	1	1	0	0
15	0	0	1	1	0	0
16	0	0	0	0	1	1
17	0	0	0	0	1	1
18	1	1	0	0	0	0
19	1	1	0	0	0	0
20	0	0	1	1	0	0

TABLE II: Saídas desejadas e saídas obtidas após o treinamento com parâmetro momento ausente

2) Parâmetro momento presente: A tabela III mostra as saídas desejadas para os 3 neurônios de saída (representadas por d1, d2 e d3) para 20 amostras e as saídas obtidas (representadas por y1, y2 e y3).

Amostra	d1	y1	d2	y2	d3	y3
1	1	1	0	0	0	0
2	1	1	0	0	0	0
3	0	0	1	1	0	0
4	1	1	0	0	0	0
5	0	0	1	1	0	0
6	0	0	0	0	1	1
7	0	0	1	1	0	0
8	0	0	0	0	1	1
9	1	1	0	0	0	0
10	0	0	1	1	0	0
11	1	1	0	0	0	0
12	1	1	0	0	0	0
13	1	1	0	0	0	0
14	0	0	1	1	0	0
15	0	0	1	1	0	0
16	0	0	0	0	1	1
17	0	0	0	0	1	1
18	1	1	0	0	0	0
19	1	1	0	0	0	0
20	0	0	1	1	0	0

TABLE III: Saídas desejadas e saídas obtidas após o treinamento com a presença do parâmetro momento

B. Dados de desempenho

Foi implementado um algoritmo para calcular a porcentagem de acertos da rede considerando amostras cujas saídas

desejadas são conhecidas. Caso a saída da rede fosse diferente da saída esperada para uma amostra, um contador para o número de erros era incrementado. Assim, ao processar todas as amostras foi possível obter o número de respostas corretas produzidas pela rede após o treinamento da mesma. Estes e outros resultados obtidos pelo treinamento são apresentados a seguir, considerando o processamento de 130 amostras.

1) Parâmetro momento ausente: Atribuindo o valor 0 ao parâmetro momento, foram obtidos os resultados de desempenho apresentados na tabela IV.

Resultado	Valor
Tempo de treinamento	4.846 segundos
Amostras classificadas corretamente	125
Erro Quadrático Médio	0.02608341
Número de épocas	1000
Porcentagem de acertos	96.1538%

TABLE IV: Dados de desempenho com parâmetro momento ausente

2) Parâmetro Momento presente: Utilizando o parâmetro momento nos resjustes de peso, foram obtidos os resultados de desempenho apresentados na tabela V.

Resultado	Valor
Tempo de treinamento	1.209 segundos
Amostras classificadas corretamente	126
Erro Quadrático Médio	0.02682567
Número de épocas	294
Porcentagem de acertos	96.9230%

TABLE V: Dados de desempenho com parâmetro momento presente

VIII. Análise dos Resultados

A partir dos resultados apresentados é possível concluir que o treinamento da rede neral PMC implementada foi bem sucedido. Nos dois cenários utilizados para testes (parâmetro momento incluído e ausente), a maioria das amostras foram classificadas corretamente, ou seja, o perceptron foi capaz de determinar o conservante adequado para cada bebida na maioria dos casos.

Nota-se que, com inclusão do parâmetro momento, a rede classificou corretamente uma amostra a mais se comparada a mesma rede sem a utilização do momento. Porém, não houve diferença significativa quanto a porcentagem de acertos nos dois cenários, comprovando que em ambos o treinamento do perceptron obteve sucesso.

É importante salientar que, por mais que a porcentagem de acertos obtida é bastante próxima nos dois casos, há uma grande diferença com relação ao tempo de treinamento e número de épocas necessárias para a conclusão do treinamento. Com o parâmetro momento incluído nos reajustes de pesos, o treinamento da rede foi concluído com cerca de 700 épocas a menos se comparado ao treinamento sem o parâmetro momento. Com isso, o momento reduziu significavelmente o tempo de treinamento, proporcionando desempenho computacional mais eficiente.

O gráfico a seguir evidencia que a diferença nos resultados obtidos nos dois cenários se deu principalmente no número de épocas e tempo de treinamento e os demais resultados foram bem próximos, mostrando que a exclusão do momento não alterou a capacidade avaliativa da rede porém tornou o treinamento mais lento.

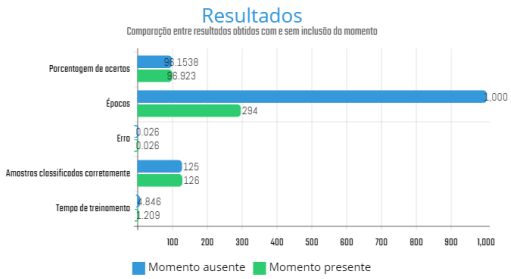

Fig. 7: Comparação entre os resultados obtidos nos dois cenários de treinamento

Os graficos a seguir evidenciam que a diferença entre os erros quadraticos médios das épocas convergiu mais rapidamente para o valor desejado quando o treinamento da rede incluiu o parametro momento, já que o número de épocas necessárias para este caso foi bem menor.

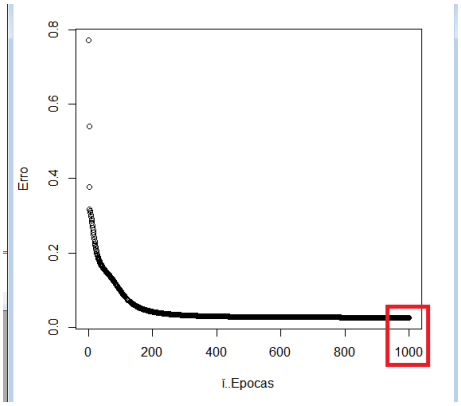

Fig. 8: Convergência ao longo das épocas para o erro desejado com parâmetro momento ausente

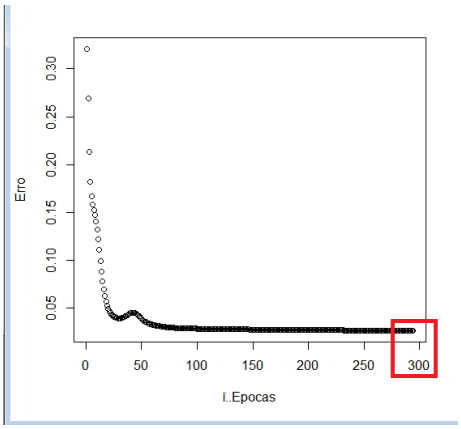

Fig. 9: Convergência ao longo das épocas para o erro desejado com parâmetro momento presente

IX. Considerações Finais

A partir do desenvolvimento deste trabalho foi possível comprovar a eficiência das redes neurais Perceptron de múltiplas camadas como classificadoras de padrões. A implementação de todo o processo necessário para a aprendizagem foi capaz de trazer uma visão mais prática e menos abstrata sobre o funcionamento das redes PMC.

A partir de testes realizados na rede implementada observou-se uma porcentagem de acertos satisfatória, próxima a 97%. Com isso, pode-se dizer que tal rede é capaz de auxiliar, com êxito, o processo de determinação de conservantes no processamento de bebidas. Para que o treinamento da rede possa atingir um desempenho melhor que o apresentado, pode-se considerar uma precisão inferior a que foi utilizada, aumentando assim o número de épocas e, consequentemente, atingindo um erro menor. Porém é importante salientar que um número maior de épocas aumenta o tempo de treinamento, tornado o algoritmo mais lento.

Além disso, foi possível obter conclusões acerca da inclusão do parâmetro momento nos reajustes de pesos. Na ausência ou presença de tal parâmetro a rede apresentou resultados satisfatórios, com a porcentagem de acertos praticamente inalterada nos dois casos. A diferença pôde ser observada no número de épocas e, consequentemente, no tempo de treinamento. O uso do parâmetro momento reduziu em quase 4 vezes o tempo de treinamento da rede, considerando os mesmos valores iniciais para os vetores de pesos.

X. Referências Bibliográficas

1) SILVA, I. D.; SPATTI, D.; FLAUZINO, *Redes Neurais Artificiais para Engenharia e Ciências Aplicadas - Curso Prático. ARTLIBER,2010.*
2) JULIA, R. M. da S. A, *Demonstração da Regra Delta Generalizada para Ajustes de Pesos nos Perceptrons Múltiplas Camadas*