UTILIZAÇÃO DAS REDES NEURAIS ARTIFICIAIS PARA DETECÇÃO DE MENSAGENS DE SPAM

Edson Melo de Souza

Centro Universitário Nove de Julho/Departamento de Ciências Exatas, São Paulo, Brasil, 2006

E-mail: edson.melo@uninove.edu.br

Abstract: In the cyber world, e-mail became an indispensable tool for communication, mainly in the corporative environment, due the fact that is easy to use and it has a low operational cost. However, you receive many messages that are not requested that is called spam, which causes damages and hampers proper flow in the internet. In this regard a light and fast tool becomes necessary to help in the containment of spams, with a small cost of processing. Thus, this work considers the development of a tool for fast detention of spams, aiming to conjugate new attributes, beyond the address of the shipper, such as the subject of the message and the presence of images in the same one.

Keywords: Spam, neural networks, artificial intelligence, electronic mail

Resumo: No mundo digital, o correio eletrônico tornou-se uma ferramenta indispensável para comunicação, principalmente no ambiente corporativo, dada sua praticidade e baixo custo operacional. Entretanto, inúmeras mensagens comercias não solicitadas conhecidas com *spam*, geram prejuízos aos usuários e a própria fluidez da *internet*. Neste cenário uma ferramenta leve e rápida faz-se necessária para ajudar na contenção dos *spam*s com um baixo custo de processamento. Assim, este trabalho propõe o desenvolvimento de uma ferramenta para detecção de *spam*s de forma rápida, visando conjugar novos atributos, além do endereço do remetente, tais como o assunto da mensagem e a presença de imagens na mesma.

***Palavras Chaves*: *Spam*, redes neurais, inteligência artificial, correio eletrônico.**

1 Introdução

O correio eletrônico ou *e-mail* (POSTEL, 2004) é uma das ferramentas mais utilizadas mundialmente para comunicação, principalmente no ambiente corporativo (GÔMARA e BARBOZA, 2005). Entretanto, sua praticidade e seu baixo custo acabaram tornando-se um inconveniente e uma fonte exponencial de problemas dentre os quais podemos destacar as infecções por vírus, códigos maliciosos em busca de senhas e dados no computador, e principalmente, a propaganda em massa não solicitada, denominada *spam*. Esse tipo de mensagem está crescendo a cada dia e o *spam* ou *junk e-mail* prejudicam a funcionalidade das redes de computadores por gerarem um grande tráfego na *internet*, além de prejudicar as corporações tanto no aspecto financeiro como na própria imagem institucional.

Avançadas ferramentas de filtragem de *spams* tais como *SpamAssassin*, *SpamBayes*, *SpamCop Blocking List* e *SpamProb*, são utilizadas para filtrar as mensagens de *e-mail*. Essas ferramentas possuem algoritmos embutidos tais como *Naïve Bayesian*, k-NN, SVMs, entre outros, que utilizam inteligência artificial para obterem melhores resultados na detecção dos *spams*. Cada ferramenta implementa algoritmos diferentes de acordo com as características abordadas para detecção das mensagens, podendo ser através do cabeçalho, do corpo, entre outras (CHHABRA, 205).

Segundo Stuart, Cha e Tappert (2004) os filtros aplicados através de *Naïve Bayesian* são bastante satisfatórios na identificação destas ocorrências, entretanto, necessitam de parâmetros auxiliares, tais como as Listas Negras, Brancas e Cinzas (BLACKLISTIS, 2004) para funcionarem com eficiência. A utilização de SMVs é altamente eficiente quando aplicada sobre as características lingüísticas do autor de uma mensagem (VEL, 2000). Entretanto, esse método funciona para um número baixo de mensagens, além de consumir processamento de máquina muito alto.

Neste cenário uma ferramenta leve e rápida faz-se necessária para auxiliar na contenção dos *spams.* Assim, este trabalho propõe um estudo buscando encontrar a conjugação de novos atributos (WALLACE e NIEVOLA, 2004), além do endereço do remetente, tais como o assunto da mensagem e a presença de imagens na mesma, para o desenvolvimento desta ferramenta utilizando-se das Redes Neurais Artificiais.

2 Definições de *E-mail* e *Spam*

O *E-mail* ou Correio Eletrônico (POSTEL, 2004) é um método que permite compor, enviar e receber mensagens através de sistemas eletrônicos de comunicação. O termo *e-mail* é aplicado tanto aos sistemas que utilizam a *internet* e são baseados no protocolo SMTP (*Simple Mail Transport Protocol*), como aqueles sistemas conhecidos como *intranets*, que permitem a troca de mensagens dentro de uma empresa

ou organização e são, normalmente, baseados em protocolos proprietários.

O envio e recebimento de uma mensagem de *e-mail* são realizados através de um sistema de correio eletrônico. Um sistema de correio eletrônico é composto de programas de computador que suportam a funcionalidade de cliente de *e-mail* e de um ou mais servidores de *e-mail* que, através de um endereço lógico, conseguem transferir uma mensagem de um usuário para outro. Estes sistemas utilizam protocolos que permitem o tráfego de mensagens de um remetente para um ou mais destinatários que possuem computadores conectados à *internet*.

O formato do *e-mail* está definido nas RFCs 2822, e 2045 a 2049 que são conhecidas como MIME.

Mensagens de *e-mail* consistem basicamente de duas seções principais:

- **Cabeçalho (*header*)** – É estruturado em campos que contém os dados do remetente, destinatário, assunto e outras informações sobre a mensagem;
- **Corpo (*body*)** – Contém o texto da mensagem.
- O **delimitador** entre o corpo e o cabeçalho de uma mensagem é uma linha em branco.

Um *e-mail* típico pode ser visualizado no quadro 1.

From: John Doe <markov@example.com>

Sender: Teddy Markov <mjones@machine.example>

To: Smith Anderson <smitha@example.net>

Subject: Festival

Date: Fri, 12 Nov 2004 04:52:06 -0600

Message-ID: <1234@local.machine.example>

This is a message just to say hello.

Quadro 1: Estrutura típica de um *e-mail*.
Fonte: O autor.

Pela facilidade de manipulação e baixo custo operacional, o *e-mail* facilita a disseminação do *spam* que é o termo usado para referir-se aos *e-mails* não solicitados, que geralmente são enviados para um grande número de pessoas. Quando o conteúdo é exclusivamente comercial, esse tipo de mensagem é chamada de UCE (*Unsolicited Commercial E-mail*). Dentre os problemas relacionados ao *spam* pode-se citar:

- Conteúdo impróprio;
- Prejuízos financeiros;
- Não recebimento de mensagens legítimas;
- Perda de tempo com lixo eletrônico;
- Perda de produtividade.

Existem diversas modalidades de *spam* e entre elas podemos citar:

- **Correntes (*chain letters*):** São os textos solicitando que a mensagem seja repassada para a sua lista de contatos;
- **Boatos (*hoaxes*) e lendas urbanas**: Mensagens com o objetivo de fomentar um determinado boato;
- **Propagandas**: Oferecer, sem autorização do destinatário, um produto ou serviço;
- **Códigos maliciosos**: Mensagens com o objetivo de roubar informações do usuário.

Para que um *e-mail* seja classificado como *spam*, este deve apresentar características que possam diferenciá-lo das mensagens legítimas. O grande problema encontrado nessa identificação está nos atributos que o compõe, pois um *e-mail* legítimo pode ser entendido como *spam*, dependendo do método de classificação.

Entre as mensagens consideradas como *spam* e as consideradas legítimas, encontram-se os falso-positivos e os falsos-Negativos. Os falsos-positivos são as mensagens legítimas classificadas como *spam*, já os falso-negativos são mensagens de *spam* classificadas como legítimas. A literatura apresenta diversas técnicas para detecção de *spam*, entretanto, o maior desafio atualmente é o de reduzir o número de falso-positivos, pois uma filtragem incorreta pode acarretar na perda de uma mensagem legítima.

2.1 Processo de Descoberta de Conhecimento e Mineração de Dados – KDD (*Knowledge Discovery and Data Mining*)

A descoberta de conhecimento KDD é o processo de extração de conhecimento novo, útil e interessante a partir de bases de dados. Este processo tem natureza iterativa e interativa e é composto por uma série de atividades (FAYYAD, 1996). A mineração de dados, por sua vez, pode ser considerada o núcleo da KDD, consistindo na aplicação de algoritmos de extração de padrões a partir de dados brutos.

Três etapas essenciais fazem parte do processo de KDD (FELDENS, 1997):

- **Pré-processamento**: Atividades que visam gerar uma representação conveniente para os algoritmos de mineração, a partir da base de dados. Inclui a seleção automática e/ou manual de atributos relevantes, amostragem, transformações de representação, etc.;
- **Mineração de dados**: Aplicação de algoritmos de aprendizado aos dados pré-processados;
- **Pós-processamento**: Seleção e ordenação das descobertas interessantes, mapeamentos de representação de conhecimento e geração de relatórios.

O processo é dito iterativo por ser composto de uma série de etapas seqüenciais, podendo haver retorno às etapas anteriores, isto é, as descobertas realizadas, ou a

falta delas, podem levar à novas hipóteses de descoberta. Neste caso pode-se decidir pela retomada dos processos de mineração, ou uma nova seleção de atributos, por exemplo, para validar hipóteses que surgiram durante o processo.

O núcleo de um sistema de descoberta de conhecimento é formado pelos algoritmos de extração de padrões que, quando devidamente aplicados sobre as bases de dados, auxiliam nas tarefas de mineração de dados. Os padrões detectados geram associações interessantes entre os elementos contidos na base de dados, e estes, podem ser de diferentes tipos, fornecendo o conhecimento em vários níveis.

3 Redes Neurais Artificiais

Segundo Braga, Carvalho e Ludemir (2000), em "Sistemas Inteligentes", "as Redes Neurais Artificiais (RNAs) são modelos matemáticos que se assemelham às estruturas neurais biológicas e que tem capacidade computacional adquirida por meio de aprendizado e generalização".

O aprendizado em RNAs consiste na fase onde a rede absorve dados e, a partir destes, modifica seus parâmetros de entrada. Esta etapa pode ser considerada como uma adaptação da RNA às características intrínsecas de um problema, onde se procuram cobrir um grande espectro de valores associados as variáveis pertinentes. Isto é feito para que a RNA adquira, através de uma melhora gradativa, uma boa capacidade de resposta para o maior número de situações possíveis.

Por sua vez, a generalização de uma RNA está associada à sua capacidade de dar respostas coerentes para dados não apresentados a ela durante o treinamento. Espera-se que uma RNA treinada tenha uma boa capacidade de generalização independentemente de ter sido controlada durante o treinamento. No entanto, atualmente, segundo Braga, Carvalho e Ludemir (2000), boa parte das pesquisas atuais na área, visam o desenvolvimento de modelos e técnicas de aprendizado que tenham algum controle de generalização.

O conceito atual é que aprendizado e generalização caminham juntos em vez de a generalização surgir naturalmente como conseqüência do aprendizado. O processamento da informação em RNAs é feito por meio de estruturas neurais artificiais em que o armazenamento e o processamento da informação são realizados de maneira paralela e distribuída, por elementos processadores relativamente simples. Cada elemento processador corresponde a um neurônio artificial, também conhecido como modelo de McCulloch-Pitts, ou simplesmente modelo MCP.

Ainda de acordo com Braga, Carvalho e Ludemir (2000), "*uma das características mais importantes das RNAs é que as mesmas são aproximadoras universais de funções multivariáveis contínuas. Em outras palavras, qualquer problema de aproximação de funções contínuas pode ser resolvido por meio de RNAs, independente do número de variáveis envolvidas*."

Uma RNA é composta por várias unidades de processamento, cujo funcionamento é bastante simples conforme a figura 1. Essas unidades, geralmente são conectadas por canais de comunicação que estão associados a determinado peso. As unidades fazem operações apenas sobre seus dados locais, as quais são as entradas recebidas pelas suas conexões. O comportamento inteligente de uma RNA vem das interações entre as unidades de processamento da rede. A operação de uma unidade de processamento, proposta por McCulloch-Pitts (HAYKIN, 2001) em 1943, pode ser resumida da seguinte maneira:

- Sinais são apresentados à entrada;
- Cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade;
- É efetuada a soma ponderada dos sinais que produz um nível de atividade;
- Se este nível de atividade exceder certo limite (*threshold*) a unidade produz uma determinada resposta de saída.

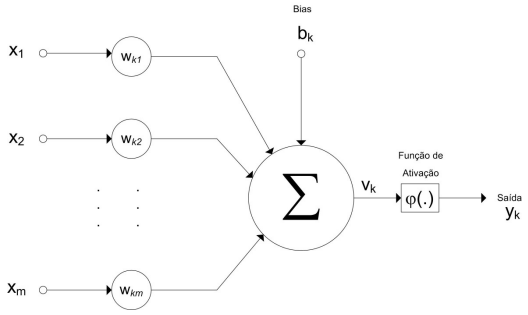

Figura 1: Modelo de um Neurônio Artificial.
Fonte: HAYKIN, 2001.

Após o modelo de McCulloch-Pitts, surgiram outras abordagens, dando flexibilidade aos pesos e maior capacidade à Rede Neural; através dos neurônios com funções de ativação não-lineares, das arquiteturas com mais de uma camada e algoritmos apropriados para alterar os pesos sinápticos.

De forma geral, nos neurônios artificiais, os seguintes elementos estão envolvidos:

- Estímulos de entrada (x): Valores apresentados à rede;
- Conjunto de sinapses (w): Ligações entre neurônios. Cada ligação possui um valor (peso), que representa a sua força: os estímulos de entrada são multiplicados pelos respectivos pesos de cada ligação, podendo gerar um sinal tanto positivo (excitatório) quanto negativo (inibitório);
- Combinador Linear $(\sum)$: Executa o somatório dos sinais produzidos pelo produto entre os pesos sinápticos e as entradas fornecidas ao neurônio. Em outras palavras, é o integrador dos sinais que chegam ao neurônio. A saída do

neurônio é definida pelo seu valor de ativação calculado mediante a equação 1;

- Saída (Y): É o valor de resposta da rede.

$$v_k = \sum_{j=1}^{m} W_{kj} x_j + b_k \quad (1)$$

Onde:

v é o campo local induzido do neurônio ***k***;

w são os pesos das conexões do neurônio ***k***;

x é o valor de cada um dos ***m*** estímulos que chegam ao neurônio ***k***;

b é o valor do bias que tem a função de aumentar ou diminuir a entrada líquida da função de ativação.

As Funções de Ativação fornecem o valor da saída de um neurônio em termos de campo local induzido.

3.1 Funções de Ativação

A Função Limiar ou *Heaviside* é referenciada na literatura para o modelo de McCulloch-Pitts. Possui característica de "tudo-ou-nada", ou seja, assume o valor 1 se o campo local induzido de um neurônio é positivo, e 0 caso seja negativo. A Função Limiar é expressa por meio da equação 2:

$$y_k = \begin{cases} 1, se\ v_k \geq 0; \\ 0, se\ v_k < 0; \end{cases} \quad (2)$$

Nos neurônios construídos com essa função, a saída ***y*** será igual a 0, caso o valor de ativação ***v*** seja negativo e 1 nos casos em que o valor de ativação seja positivo.

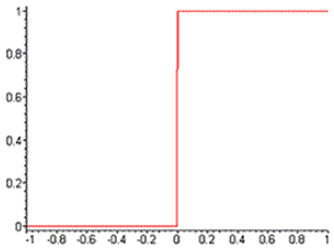

Figura 2: Função Limiar de Ativação.
Fonte: LNCC.

A Função Sigmóide que, ao contrário da função limiar, pode assumir todos os valores entre 0 e 1 também pode ser utilizada como função de ativação. A sua representação mais utilizada é a função logística, definida pela equação 3:

$$y_k = \frac{1}{1 + e(-av)} \quad (3)$$

Onde:

- ***a*** é o parâmetro de inclinação da função sigmóide;
- ***v*** é o valor de ativação do neurônio.

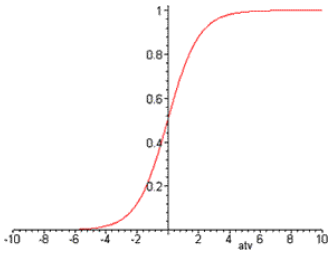

Figura 3: Função Sigmóide.
Fonte: LNCC.

Quando o valor do parâmetro é aumentado, tendendo ao infinito, esta função comporta-se como uma função de limiar, observada na figura 3:

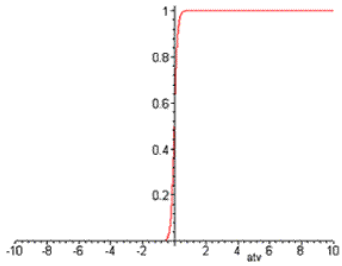

Figura 4: Função Sigmóide - a tendendo ao ∞.
Fonte: LNCC.

A Tangente Hiperbólica também pode ser utilizada como função de ativação. Como função logística, também possui forma de “s”, assumindo valores entre 1 e -1, sendo representada pela equação 4:

$$\delta(v) = \tanh(v) \quad (4)$$

Onde:

- ***v*** é o valor de ativação da unidade.

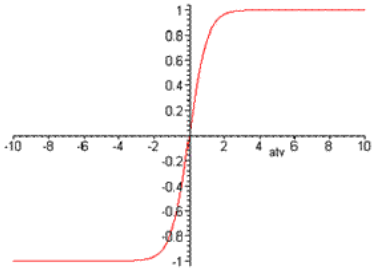

Figura 5: Função Tangente Hiperbólica.
Fonte: LNCC.

3.2 Processo de Aprendizagem

Para que uma RNA possa fornecer resultados convenientes é necessário que a mesma passe por uma fase de treinamento, onde seus pesos são ajustados de forma que ela se adapte aos diferentes estímulos de entrada. Durante a fase de treinamento, ocorre o seu aprendizado.

Há vários processos de aprendizado, os quais, de forma geral, podem ser classificados em:

- **Aprendizado Supervisionado**: É fornecida uma referência do objetivo a ser alcançado. Um exemplo de processo de Aprendizado Supervisionado é o Aprendizado por Correção de Erro.
- **Aprendizado Não-Supervisionado**: Neste caso não é fornecida nenhuma referência externa. Podemos citar como exemplo o processo de Aprendizado Competitivo e o Aprendizado *Hebbiano*.

Entre os diversos modelos de RNAs, o do tipo *Perceptron* foi o pioneiro nesta área: simples e de fácil implementação, utiliza neurônios do tipo McCulloch-Pitts. Limitada à classe de problemas linearmente separáveis, pode, no entanto, ser utilizada em tarefas de classificação simples. O *Perceptron*, proposto por Rosenblatt, é composto pelo neurônio de McCulloch-Pitts, com Função de Limiar e Aprendizado Supervisionado. Sua arquitetura consiste em uma camada de entrada e uma camada de saída. A limitação desta RNA se encontra na reduzida gama de problemas que consegue tratar, ou seja, na classificação de conjuntos linearmente separáveis.

3.3 Modelo *Perceptron* de Multi-Camadas

O *Perceptron* de Multi-Camadas é uma extensão do *Perceptron* simples, capaz de trabalhar com problemas não-linearmente separáveis, figuras 7 e 8. Este avanço foi possível através da utilização de, pelo menos, uma camada entre a camada de entrada e a camada de saída. Estas camadas intermediárias, conhecidas como camadas ocultas, trabalham como um reconhecedor de características, que ficam armazenadas nos pesos sinápticos. O algoritmo de treinamento mais utilizado para esse modelo é o "*Backpropagation*", um tipo de Aprendizado Supervisionado por Correção de Erro.

A modelagem da arquitetura de uma rede *Perceptron* Multi-Camadas, figura 6, envolve a escolha da quantidade de camadas e o número de unidades em cada camada. Destacam-se para esse modelo alguns aspectos importantes:

- Escolha do número de unidades de entrada;
- Definição da função de ativação que irá direcionar o comportamento da rede;
- Codificação da camada de saída e a formatação da resposta da rede.

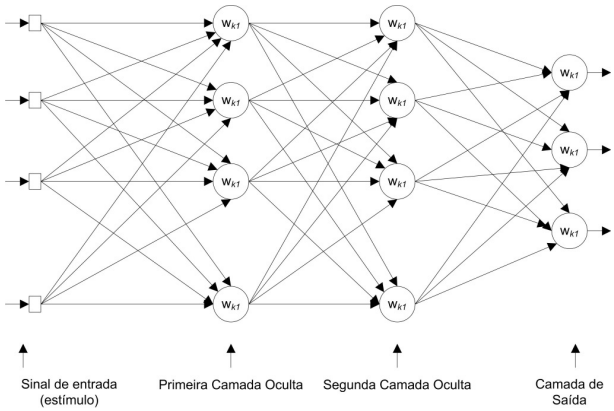

Figura 6: Grafo arquitetural de um perceptron de múltiplas camadas.
Fonte: HAYKIN, 2001.

Outros parâmetros devem ser escolhidos, referentes ao treinamento tais como taxa de aprendizado e o conjunto de treinamento. A respeito do conjunto de treinamento devem ser estudados os dados relevantes, os quais destaquem as características que devem realmente ser "aprendidas" pela rede. O processamento de cada unidade é influenciado pelo processamento efetuado pelas unidades das camadas anteriores. Cada camada desempenha um papel específico, como segue:

- **Camada de Entrada**: Receptora de estímulos;
- **Primeira Camada Oculta**: Cada unidade desta camada define uma reta no espaço de decisão, refletindo características dos padrões apresentados;
- **Segunda Camada Oculta**: Combina as retas definidas pela camada anterior, formando regiões convexas onde o número de lados é definido pelo número de unidades da camada anterior, conectados a unidade desta camada;
- **Camada de Saída**: Combina as regiões formadas pela camada anterior, definindo o espaço de saída da rede.

As camadas intermediárias da rede são como detectores de características, as quais serão representadas, internamente, através dos pesos sinápticos.

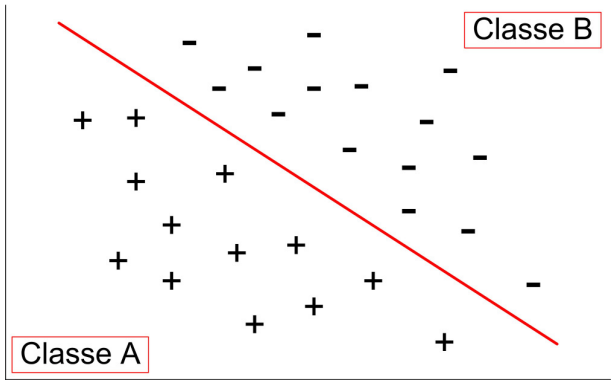

Figura 7: Classes Linearmente Separáveis.
Fonte: O autor.

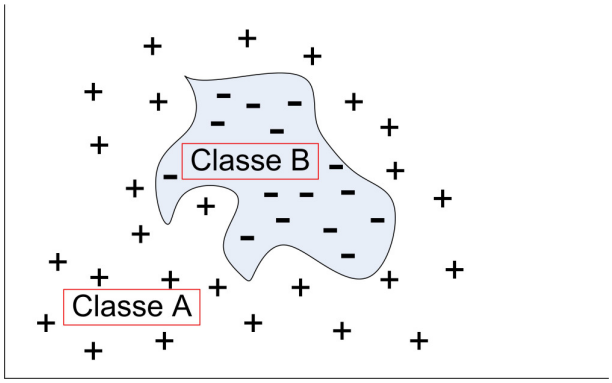

Figura 8: Classes Não-Linearmente Separáveis.
Fonte: O autor.

3.4 Algoritmo *Backpropagation*

Backpropagation é o algoritmo para treinamento de redes Multi-Camadas mais difundido. É constituído de duas fases: propagação e retropropagação.

Na fase de propagação, após ser apresentado o padrão de entrada, a resposta de uma unidade é propagada como entrada para as unidades na camada seguinte, até a camada de saída, onde é obtida a resposta da rede e o erro é calculado.

Na retropropagação os pesos são recalculados desde a camada de saída até a camada de entrada, ou seja, no sentido contrário da apresentação. Durante a fase treinamento apresenta-se um conjunto formado pelo par "entrada para a rede" e "valor desejado para resposta" a entrada. A saída será comparada ao valor desejado e será computado o erro global da rede, que influenciará na correção dos pesos no passo de retropropagação, apesar de não haver garantias que a rede forneça uma solução ótima para o problema.

Os passos do algoritmo *backpropagation* são descritos a seguir:

1- **Inicialização**: Inicializa os pesos sinápticos e o bias (*b*) aleatoriamente, com valores no intervalo [-1, 1];

2- **Apresentação dos padrões de treinamento***:* Treinamento "*on-line*": Para cada exemplo do conjunto de treinamento, efetue os passos 3 e 4. Treinamento "em lote": Para cada "época" do conjunto de treinamento, efetue os passos 3 e 4.

3- **Computação adiante (Propagação)***:* Depois de apresentado o exemplo do conjunto de treinamento $T = \{x(n), d(n)\}$,sendo $x(n)$ a entrada apresentada à rede e $d(n)$ a saída desejada, calcule o valor da ativação v_j por meio da equação 5:

$$y_k = \frac{1}{1+e(-av)} \quad (5)$$

Para o cálculo da saída y da unidade k , utilizando a função sigmóide, como no exemplo. Utilize a saída das unidades de uma camada como entradas para a seguinte, até a última camada. A saída das unidades da última camada será a resposta da rede.

4- **Calcule o Sinal de Erro:** Fazendo a saída $y_j = O_j(n)$, será $O_j(n)$ a resposta da rede. Calcule o sinal de erro de acordo com a equação 6:

$$e_j(n) - d_j(n) - O_j(n) \quad (6)$$

Onde:

$d_j(n)$ é a saída desejada com resposta para cada unidade na interação (n). Este sinal de erro será utilizado para computar os valores dos erros das camadas anteriores e fazer as correções necessárias nos pesos sinápticos.

5- **Computação para trás (Retropropagação):** Calcule os erros locais, d , para cada unidade, desde a camada de saída até a de entrada. O gradiente local é definido pela equação 7 e 8:

$$\delta_j(n) = e_j(n) O_j(n)(1 - O_j(n)) \quad (7)$$

Para a unidade da camada de saída ou

$$\delta_j(n) = O_j(n)(1 - O_j(n)) \sum \delta_k W_{kj} \quad (8)$$

Para as unidades das demais camadas.

Onde:

- $O_j(i - O_j)$ – é a função de ativação diferenciada em função do argumento. Valor de ativação;
- d_k – é o erro das unidades da camada anterior, conectadas a unidade j ;
- w_{kj} – são os pesos das conexões com a camada anterior.

Após o cálculo dos erros de cada unidade, calcule o ajuste dos pesos de cada conexão segundo a regra delta generalizada e atualize os pesos segundo a equação 9:

$$\Delta w_{kj}(n+1) = \alpha w_{kj}(n) + \eta \delta_j y_j \quad (9)$$

Para o cálculo dos ajustes dos pesos faça 10:

$$w(n+1) = w(n) + \Delta w_{kj}(n) \quad (10)$$

Para atualizar os pesos sinápticos onde:

- α - é a constante de momento, quando α=0, esta função funciona como a regra delta comum;
- η - é a taxa de aprendizado;
- y_j - é a saída produzida pela unidade j.

6- **Interação:** Refaça os itens 3, 4 e 5 referentes à propagação, cálculo do erro e retropropagação, apresentando outros estímulos de entrada, até que sejam satisfeitas as condições de treinamento; as quais podem ser: o erro da rede está baixo, sendo pouco alterado durante o treinamento; o número máximo de ciclos de treinamento foi alcançado.

3.5 Correção de Erro

O erro de uma Rede Neural pode ser calculado como a diferença entre a saída real gerada pela rede e a saída desejada, fornecida em um ensino supervisionado a partir da equação 11:

$$e_k = d_k - y_k \quad (11)$$

Onde para um estímulo k:

- e_k – sinal de erro;
- d_k – saída desejada apresentada durante o treinamento;
- y_k – saída real da rede após a apresentação do estímulo de entrada.

Durante o aprendizado supervisionado, portanto, os erros vão sendo calculados sucessivamente, até que cheguem a um valor satisfatório, definido a *priori*. Sendo assim, surge uma curva de erros, a qual está diretamente relacionada à natureza do modelo de neurônio utilizado.

Se a rede é formada por unidades lineares, como no modelo de McCulloch-Pitts, na superfície de erro será encontrada um único valor mínimo. Por outro lado, se a rede é constituída por unidades não-lineares, podem ser encontrados diversos valores mínimos chamados de mínimos locais, além do mínimo global.

O processo de Aprendizado por Correção de Erros utiliza algoritmos para caminhar sobre a curva de erros, com o intuito de alcançar o menor valor de erro possível, o mínimo global. Muitas vezes, o algoritmo não alcança este mínimo global, atingindo o que chamamos de mínimo local. Caso este erro alcançado seja desfavorável, é necessário recomeçar o processo de aprendizado.

Para a correção do erro, os pesos da rede devem ser ajustados, de forma a aproximar a saída real à desejada. De acordo com a Regra Delta de Aprendizado, apresentada a seguir, tal ajuste dependerá do próprio erro calculado; do valor do estímulo de entrada que é "transmitido" pelo peso a ser ajustado; e também da taxa de aprendizado, a qual se relaciona à cautela com que a curva de erros é percorrida. Para um dado estímulo k, no passo de treinamento n, aplica-se a equação 12:

$$\Delta w_{ji} = \eta e_k(n) x_j(n) \quad (12)$$

Onde:

- $\Delta w_{ji}(n)$ – Valor de ajuste a ser acrescido ao peso w_{ji};
- η – Taxa de aprendizado;
- $e_k(n)$ – Valor do erro;
- $x_j(n)$ – Valor do estímulo.

O valor atualizado do peso será 13:

$$w(n+1) = w(n) + \Delta w_{kj}(n) \quad (13)$$

Portanto, pode-se utilizar a Regra Delta para corrigir os valores dos pesos, minimizando a função de erro e, também conhecida como Função de Custo 14:

$$\varepsilon(n) = \frac{1}{2} e^2(n) \quad (14)$$

Onde:

- $\varepsilon(n)$ - Erro da rede no passo n do treinamento;
- $e^2(n)$ - Valor da função de custo no passo n do treinamento.

4 Materiais e Métodos

Para o desenvolvimento do trabalho foram selecionadas 5.437 mensagens de correio eletrônico no período compreendido entre fevereiro e junho de 2006 em uma empresa de médio porte com a colaboração de 9 funcionários, onde, após uma filtragem manual por relevância e exclusão de referências cruzadas, obteve-se um total de 3.349 mensagens sendo 1.645 consideradas legítimas e 1.704 como *spam*.

Os algoritmos para extração e tratamento dos dados foram desenvolvidos em linguagem ANSI C em um ambiente *Linux/Debian-Sarge* 3.3.5-13 em um *Atlhon* 1.8 GHz com 256MB de memória RAM e *Hard Disk* de 40GB e compilados com o GNU/GCC 3.3.5 com a biblioteca REGEX.

As mensagens foram alocadas em diretórios para distinguir os tipos estudados a fim de facilitar os métodos de busca dentro dos arquivos. Um algoritmo foi desenvolvido para efetuar a varredura em tais diretórios e extrair informações para a composição dos atributos.

Através do KDD, foram extraídas as palavras do campo “Assunto” para compor uma lista de restrições.

Tais palavras foram elencadas de acordo com o grau de aparição nas mensagens apontadas como *spam*. Na segunda etapa foram extraídos os endereços de *e-mail* com base na RFC 822 [*Request for Comments*: 822, 1982], seguidos da verificação da presença de imagens contidas no *header* da mensagem.

4.1 Pré-processamento dos Atributos

Após a análise dos dados brutos obtidos, estes foram submetidos a um algoritmo que efetuou o tratamento e a separação dos caracteres alfa-numéricos por meio de uma lista definida pela relevância dos mesmos. Os caracteres selecionados foram:

{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u, v,x,z,y,z,-,_}

Para obter-se uma boa performance e homogeneidade dos dados, todos os caracteres foram convertidos para caixa baixa, objetivando um menor esforço do algoritmo no processamento.

Após esta fase, foram efetuadas análises estatísticas a fim de consolidar os padrões a serem utilizados nos estudos com a RNA. Determinou-se que a utilização de percentuais da ocorrência de aparição dos caracteres sobre o total de caracteres que compõe o endereço do *e-mail,* assim como o campo "Assunto", era de grande relevância para o estudo. Desta forma, foram efetuados os cálculos dos percentuais através de um algoritmo aplicado sobre os dados brutos conhecidos. Tais informações foram obtidas mediante a aplicação da equação 15:

$$f(c_i) = \frac{\sum_{i=1}^{t} c_i}{T} \quad (15)$$

Onde:

f = percentual do *i*-ésimo caractere sendo analisado;

c_i = *i*-ésimo caractere analisado;

$T = t$ = total de caracteres do endereço de *e-mail* ou do assunto.

A partir dos percentuais obtidos pela equação 15, foi possível qualificar os atributos para compor a base de dados. São eles:

Atributos sobre o endereço do *e-mail*

1 - Percentual de números;

2 - Percentual de letras estrangeiras;

3 - Percentual de palavras restritas;

Atributos sobre o *header* do *e-mail*

4 - Presença de Imagem (atribui-se o valor 0.12, caso encontrada);

Atributos sobre o assunto do *e-mail*

5 - Percentual de palavras estrangeiras;

Atributo de Qualificação

6 - Classificações da mensagem (**0** para normal e **1** para *spam*).

O próximo passo foi efetuar os testes através de uma RNA supervisionada MLP (*Multi Layer Perceptron*) treinada com o algoritmo *Backpropagation*.

4.2 Arquitetura MLP Utilizada

Para o desenvolvimento do trabalho, foi adotado o modelo MLP. Este modelo destaca-se entre os outros reconhecidos pela literatura por sua implementação ser relativamente fácil e seus resultados geralmente demonstrarem ser satisfatórios.

Os treinamentos foram executados utilizando-se uma RNA, figura 9, com a seguinte configuração: 6 neurônios na camada de entrada, 20 neurônios na camada oculta e 1 neurônio na camada de saída.

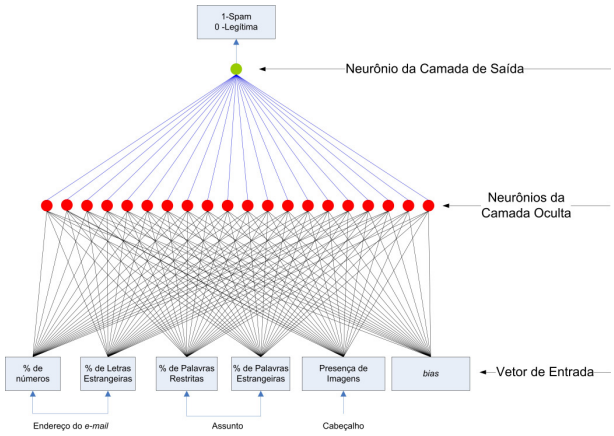

Figura 9: Arquitetura MLP utilizada.
Fonte: O autor.

Conforme pode ser visto na figura 9, o vetor de entrada é composto por 6 elementos. Esses elementos são os 5 atributos extraídos dos *e-mails* os quais podem qualificar um *e-mail* como *spam* ou legítimo, além do bias.

4.3 Treinamento da Rede MLP

O conjunto de dados obtido através do pré-processamento foi submetido à RNA para treinamento e aprendizagem da rede em 3 situações diferentes, ou seja, os valores estipulados para o erro nos treinamentos foram: 0.001, 0.002 e 0.003. Os resultados dos treinamentos são apresentados na tabela 1.

Para o treinamento da rede foram utilizadas 1.704 mensagens sem duplicidades internas e ajustadas para submissão. O tempo de treinamento para o valor 0.001 foi de 10 minutos. Para os outros dois, 6 e 8 segundos, respectivamente. Para o erro 0.001, o tempo gasto não

apresentou relevância, pois o processo foi efetuado apenas 1 vez, como é de característica das RNAs.

Erro Inicial	Épocas	Erro Final
0.001	4.589	0.000769
0.002	113	0.001953
0.003	7	0.002917

Tabela 1: Convergência em épocas.
Fonte: O autor.

5 Resultados e Discussões

Após a fase de treinamento, um novo conjunto de dados constituído por 1.645 mensagens foi submetido aos mesmos processos de tratamento e posteriormente submetido à RNA para verificação.

Os resultados obtidos foram satisfatórios apresentando um alto índice de acertos. O erro médio quadrado manteve o comportamento estável, apesar de não convergir espontaneamente. Na análise de um *e-mail*, o tempo de resposta oscilou entre 0.1 e 0.2 segundos.

Na tabela 2 pode-se verificar a relação entre as classificações corretas e o número dos falso-positivos e falso-negativos, os quais são comparados com alguns resultados encontrados na literatura.

Taxa de aprendizagem 0.002	Algoritmos Utilizados			
Classificação das Instâncias	Naïve Bayes	Ada Boost	C4.5	**MLP**
Corretas	89,74%	82,85%	96,11%	**91,13%**
Incorretas	10,26%	17,15%	3,89%	**8,87%**
Falso Positivos	-	-	-	***4,10%***
Falso Negativos	-	-	-	***4,77%***

Tabela 2 – Resultados Obtidos MLP X Literatura.
Fonte: WALLACE e NIEVOLA, 2004.

6 Conclusão

O algoritmo *backpropagation* forneceu respostas satisfatórias quanto à classificação correta das mensagens com os atributos selecionados. A taxa média de erros para classificação de mensagens incorretas manteve-se abaixo das taxas encontradas na literatura para alguns métodos.

O MLP demonstrou ser um método rápido e eficiente para detecção das mensagens de *spam* com um baixo custo de processamento.

Para trabalhos futuros, novos atributos podem ser considerados para aumentar a eficiência do MLP assim como a utilização de outras técnicas de inteligência artificial podem ser agregadas ao estudo, principalmente para uma seleção com maior refinamento dos atributos.

7 Bibliografia

BRAGA, Antônio de P., CARVALHO, André P. de L, LUDERMIR, Teresa B. / Fundamentos de Redes Neurais Artificiais; 11° Escola de Computação - Rio de Janeiro. DCC/IM, COOPE/Sistemas, NCE/UFRJ, 1999.
CHHABRA, Shalendra. Fighting Spam, Phishing and Email Fraud. Riverside, 2005. 244p. Master of Science - Graduate Program in Computer Science, University of California, Riverside, December, 2005. P. 244.

CUNNINGHAM, Pádraig; NOWLAN, Niamh; DELANY, Sarah Jane & HAAHR, Mads. A Case-Based Approach to Spam Filtering that Can Track Concept Drift. Department of Computer Science, Trinity College Dublin 2School of Computing, Dublin Institute of Technology, 2003.

FAYYAD, M. et al. (1996) Advances in Knowledge discovery and data mining. Menlo Park, CA: AAAI Press.

FELDENS, M. (1997) Descoberta de Conhecimento em Bases de Dados e Mineração de Dados. Curso de Pós-Graduação em Ciência da Computação – UFRGS. Artigo publicado na I Oficina de Inteligência Artificial – UCPel.

GHOSH, Madhujit; GUERNSEY, Jonathan J. & KAUR, Devinde. Exploration of Neuro-Fuzzy Spam Filtering based on Naive Bayesian Filters. In: International Conference on Machine Learning: Models, Technologies and Applications. Las Vegas, Nevada, USA. June, 2003. P. 1-6.

GÔMARA, Marcelo Pereira; BARBOZA, Patrícia Medeiros. O monitoramento do *e-mail* corporativo. Valor Econômico, 2005. Disponível em: http://clipping.planejamento.gov.br/Noticias.asp?NOTCod=233284. Acessado em 2 abr. 2006.

GRAY, Alan & HAAHR, Mads. Personalised, Collaborative Spam Filtering. Distributed Systems Group, Department of Computer Science, Trinity College Dublin, Ireland, 2004. CFTD/03/219.

HAYKIN, Simon. Redes Neurais Artificias: Princípios e Práticas. 2ª ed. Porto Alegre, Bookman, 2001. P. 900.

LEES, Jennifer. A Discriminative Approach to Junk Mail Classification. A dissertation submitted to the University of Cambridge towards the M.Phil in Computer Speech, Text and Internet Technology. Computer laboratory University of Cambridge, 2005. P. 91.

LNCC – Laboratório Nacional de Computação Científica. Ministério da Ciência e Tecnologia. Acesso em: 14 ago. 2006.

MEYER, T.A. & WHATELEY , B. SpamBayes: Effective open-source, Bayesian based, email classification system. In in proceedings of Conference on Email and Anti-Spam 2004, July 2004.

MILLER, Chris. Neural Network-based Anti*spam* Heuristics. Group Product Manager Enterprise Email Security. Symantec NEURAL NETWORK-BASED ANTISPAM HEURISTICS, 2003. P. 1-8.

POSTEL, Jonathan B. Simple Mail Transfer Protocol. RFC 821 - Simple Mail Transfer Protocol. Internet FAQ Archives, 2004. Disponível em: http://www.faqs.org/rfcs/rfc821.html. Acesso em: 28 mar. 2006.

RCF: The Request for Comments (RFCs). http://www.rfc-editor.org. Acesso em: 28 mar. 2006.

RENNIE, Jason D. M. An Application of Machine learning to Email Filtering. Artificial Intelligence. Proceedings of the KDD-2000 Workshop on Text Mining, Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Lab Massachusetts Institute of Technology. KDD2000 Text Mining Workshop Boston, MA USA, 2000. P. 1-6.

ROCHA, Luis Fernando. O Spam está matando o e-mail?. Módulo Securit, 2003. Disponível em: http://www.modulo.com.br/pt/page_i.jsp?page=3&catid=7&objid=2474&pagecounter=0&idiom=0. Acesso em: 1º abr. 2006.

RUSSELL, Stuart & NORVIG, Peter. Inteligência Artificial. 2ª ed. Rio de Janeiro, Elsevier Editora Ltda, 2004. P. 1021.

SPAMBAYES. http://spambayes.sourceforge.net. Acesso em: 10 mar. 2006.

SPAMCOP BLOCKING LIST (SCBL). http://www.spamcop.net/bl.shtml. Acesso em: 10 mar. 2006.

SPAMPROBE. http://spamprobe.sourceforge.net/ Acesso em: 10 mar. 2006.

STUART, Ian; CHA, Sung-Hyuk & TAPPERT, Charles. A Neural Network Classifier for Junk E-Mail. Proceedings of Student/Faculty Research Day, CSIS, Pace University, May 7th, 2004. P. 5.1-5.6.

TEIXEIRA, Renata Cicilini. Combatendo o SPAM. Aprenda como Evitar e Bloquear E-mails Não-solicitados. 1a edição, São Paulo. 2004.

TRETYAKOV, Konstantin. Machine Learning Techniques in Spam Filtering. Institute of Computer Science, University of Tartu. Data Mining Problem-oriented Seminar, MTAT.03.177, May 2004, pp. 60-79.

VEL, Oliver de. Mining E-mail Authorship. In: The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August 20 - 23, 2000. P 1-7.

WALLACE, A.B.S. de Macedo & NIEVOLA, Júlio Cesar. Avaliação de Classificadores Anti-*spam* Aplicada no Campo de Cabeçalho de E-mail "From:". Centro de Ciências Exatas e de Tecnologia – Pontifícia Universidade Católica do Paraná (PUC-PR). Curitiba, PR. Brasil. Abril, 2004. P. 7.