Resumo

O correio eletrônico já se tornou um dos mais importantes meios de comunicação existente. Mas o problema na sua utilização surge ao enviar mensagens indesejadas. A quantidade de mensagens indesejadas enviadas diariamente é enorme, e com isso grandes problemas como desperdício de largura de banda, tempo e produtividade começam a se tornar um grande problema. Muitos algoritmos já foram aplicados com o intuito de analisar essas mensagens, porém os que apresentam maiores eficiências nos dados obtidos são os que utilizam algoritmos inteligentes baseado em redes neurais artificiais. Este trabalho visa mostrar um comparativo usando três algoritmos inteligentes baseado em Redes Neurais Artificiais na classificação de mensagens legítimas e não legítimas. Será mostrado qual classificador melhor se adéqua à base de dados usada. Serão usadas três técnicas de mineração de dados: SOM (Mapas Auto-Organizáveis), Redes MLP (Perceptron Multi Camadas), Redes RBF (Função de Base Radial) e Árvores de Decisão.

PALAVRAS-CHAVE: *E-mail, Spam, Redes Neurais Artificiais, Perceptron Multi Camadas, Função de Base Radial, SOM.*

Abstract

The electronic mail has become one of the most important means of communication exists. But the problem arises in its use to send junk mail. The amount of spam messages sent daily is enormous and with big problems like this waste of bandwidth, time and productivity start to become a big problem. Many algorithms have been applied in order to analyze these messages, but those with greater efficiencies in the data are those using intelligent algorithms based on artificial neural networks. This paper shows a comparative study using three intelligent algorithms based on Artificial Neural Networks in classification of legitimate messages and not legitimate. It will be shown that the classifier would fit best into the database used. Be used three techniques of data mining: SOM (Self-Organizing Maps), networks MLP (Multi Layer Perceptron) networks RBF (Radial Basis Function) and Decision Trees.

KEYWORDS: *E-mail, Spam, Artificial Neural Network, MultiLayer Perceptron, Radial Base Function, SOM.*

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

SOM - Self-Organizing Maps (Mapas Auto-Organizáveis) Kohonen

MLP – MultiLayer Perceptron (Perceptron Multi Camadas)

RBF - Radial Basis Function (Função de Base Radial)

ONG - Organização Não-Governamental

UCE - Unsolicited Comercial E-mail

SPIT - Spam over IP Telephony

ADSL - Asymmetric Digital Subscriber Line

HTML - HyperText Markup Language

OCR - Optical Character Recognition

SVM - Support Vector Machine

k-NN - K-Nearest Neighbour

AdaBoost - Adaptive Boosting

RNA - Redes Neurais Artificiais

SN - Sistema Nervoso

EMQ - Erro Médio Quadrado

WTA - Winner-Takes-All

XOR - Exclusive OR

ADALINE - Adaptive Linear Neuron

UCI-R - Repositório Universidade da Califórnia Irvine

WEKA - Waikato Environment for Knowledge Analysis

Capítulo 1

Introdução

A cada dia a Internet se faz mais presente na vida das pessoas, revolucionando a forma como os dados são obtidos. Diversos serviços são disponibilizados através do uso da internet, mas de fato, o principal deles é o e-mail.

O correio eletrônico existe antes mesmo da Internet, porém somente com a popularização da Internet é que o e-mail pode se expandir e se tornar um dos maiores meio de comunicação. Segundos pesquisas realizadas, mais de 80% dos usuários preferem o e-mail ao telefone. Sua extrema facilidade para enviar a mensagem, sua rapidez ao entrega a mensagem, normalmente poucos minutos, esteja o destinatário ao seu lado ou do outro lado do mundo e principalmente seu custo, normalmente custo zero, o faz ser considerado o maior meio de comunicação.

Todas essas qualidades trazem consigo algumas conseqüências: o mau uso do e-mail. Pessoas se aproveitam dessa facilidade para fazer publicidade gratuita e não autorizada, enviando e-mails em massa para destinatários que não solicitaram nem autorizaram essas mensagens [1].

A quantidade de mensagens não autorizadas (*SPAM*) enviadas diariamente em cada vez maior. Isso pode trazer diversos problemas à pessoa ou à organização. Por exemplo, um alto consumo de banda, prejuízos financeiros causados por fraude, má utilização de servidores devido à necessidade de armazenar e processar os spams, grande perda de tempo selecionando mensagens legitimas e apagando as não legitimas e principalmente podendo provocar atraso no envio e recebimento das mensagens legítimas [1].

Por causa disso, diversos algoritmos são criados e testados todos os dias com o intuito de reduzir ou até mesmo eliminar essas mensagens não legitimas. Porém algoritmos comuns têm se apresentado ineficientes na classificação correta dessa quantidade de spam enviado diariamente.

Com isso é apresentado alguns testes comparativos usando algoritmos inteligentes baseados em Redes Neurais com o intuito de mostrar o resultado final dessas classificações de email legítimo e email não legítimos. Serão usadas três técnicas de mineração de dados: Mapas Auto-Organizáveis (SOM), Redes MLP (Perceptron Multi Camadas) e Redes RBF (Função de Base Radial).

1.1 SPAM

Com a popularização do correio eletrônico, esse serviço tornou-se bastante utilizado para envio de *spam* – termo criado para denominar mensagens eletrônicas que, na maioria dos casos, são de intuito publicitário, visando de alguma forma, a promoção de serviços, produtos e/ou eventos e que são enviadas sem a autorização prévia dos destinatários. O *spam* é definido como *e-mail* não solicitado, emitido de forma indiscriminada, direta ou indiretamente, por um remetente que não tem nenhum relacionamento com o destinatário [1].

O primeiro *spam* surgiu em 1994 e desde então esta prática de enviar *e-mails* não solicitados tem sido cada vez mais comum, em quantidades cada vez maiores e com diversos objetivos, utilizando-se de diferentes aplicativos e meios de propagação na rede. Com a evolução da Internet, dos aplicativos e tecnologias, o spam está crescendo junto também. Hoje em dia, o *spam* vai além de apenas enviar mensagens publicitárias, mas também é uma forma de fazer propagação de vírus e golpes [2].

Algumas das principais categorias de *spams* são [2]:

- **Golpes**: geralmente são usadas longas explicações com o simples intuito de confundir os destinatários, oferecendo uma excelente oportunidade financeira, quando na verdade não passa de uma fraude. Um comum golpe são aqueles que falam para o usuário enviar um determinado valor, normalmente de pequeno valor, ou até mesmo um depósito, e com isso o usuário encaminha para mais destinatários com a promessa de, por exemplo, se 10% dos destinatários que receberem fizerem o mesmo, você

terá um valor muitas vezes ao valor investido. Outro comum golpe são golpes onde usam alguma associação ou ONG[1], informando que a tal associação está com problemas e precisa de ajuda financeira para continuar a com a obra social;

- **Financeiro**: são relacionados com empréstimo facilitado e ofertas de crédito sem consulta aos órgãos de proteção ao crédito;

- **Saúde**: aparecem normalmente como *marketing* de produtos farmacêuticos, diversos tipos de pílulas, drogas, poções e produtos naturais. “emagreça enquanto dorme.”, “Aumente seu desempenho sexual.”, “Aumente seu vigor físico”, “Tenha uma pele mais bonita.” São algumas das promessas feitas nesse tipo de *spam*;

- **Boatos (*hoaxes*)**: Estão associados a histórias falsas, fantasiosas escritas com o intuito de alarmar ou iludir aqueles que estão lendo. Geralmente tratam de pessoas que necessitam urgentemente de algum tipo de ajuda. Um exemplo de boato enviado por spammers brasileiros menciona um livro usado em escolas norte-americana que traz um mapa onde a Amazônia é considerada território internacional:

> *Todos nós já ouvimos falar que os americanos querem transformar a Amazônia num parque mundial com tutela da ONU, e que os livros escolares americanos já citam a Amazônia como floresta mundial.*
>
> *Pois chegaram as mãos de um amigo o livro didático "Introduction to geography" do autor David Norman, livro amplamente difundido nas escolas públicas americanas para o Junior High School (correspondente à nossa sexta série do 1º grau).*
>
> *Olhem o anexo e comprovem o que consta a página 76 deste livro e vejam que os americanos já consideram a Amazônia uma área que não é território brasileiro, uma área que rouba território de oito países da América do Sul e ainda por cima com um texto de caráter essencialmente preconceituoso.*
>
> *Vamos divulgar isso para o maior número de pessoas possível a fim de podermos fazer alguma coisa ante a esse absurdo...*

[1] ONG – Organização Não-Governamental são associações do terceiro setor da sociedade civil, que se declaram com finalidades públicas e sem fins lucrativos, que desenvolvem ações em diferentes áreas e que, geralmente, mobilizam a opinião pública e o apoio da população para modificar determinados aspectos da sociedade.

- **Phishing:** o *phishing* tem sido utilizado por criminosos organizados devido aos seus altos ganhos potenciais ganhos financeiros. A idéia é que o *e-mail* se assemelha bastante com o *e-mail* legítimo de uma organização (geralmente bancos ou instituições financeiras). Neles, os spammers tentam convencer, por isso o e-mail tenta ser o mais fiel a um possível e-mail legítimo da instituição, com logomarca e tudo mais iguais, os usuários a digitar seus dados como senha de acesso, letras de acesso, número da conta, em um formulário ou um *website* idêntico ao da organização, porém ele é controlado pelo golpista.

- **Educação:** apresentam-se sob a forma de publicidade de cursos de capacitação e qualificação em diversos níveis. Oferecem, de uma forma geral, facilidades para se conseguir um diploma de nível superior e/ou especialização podendo também abrangir cursos de especialização como mecânico, eletricista;

- **Propagandas:** conhecidos como o *e-mail* comercial não solicitado (*Unsolicited Comercial E-mail – UCE)*, divulgam seus diversos produtos e serviços como comumente vistos em spams, entre eles: programas de computadores, produtos de beleza. De acordo com o ANTISPAM.BR,

> esse tipo de *spam* é motivo de discussão e polêmica, afinal, é possível fazer *marketing* na internet sem fazer spam. No entanto, aqueles que insistem em divulgar sua imagem ou negócio por meio de mensagens não solicitadas, acabam comprometendo sua credibilidade. A solução é o *marketing* responsável na rede. Por outro lado, alguns *spams* oferecem produtos que não existem e serviços que nunca serão entregues;

- **Códigos Maliciosos – Malware:** são *spams* que contém programas capazes de causar danos e comprometer os sistemas computacionais, entre eles estão: os vírus, *worms*[2] *e trojans*[3]*.* Normalmente é solicitado ao

[2] *Worms* vem do inglês e significa verme, em computação, é um programa auto-replicante, semelhante a um vírus. Enquanto que um vírus infecta um programa e necessita deste

usuário que execute um determinado arquivo *anexado* ao *e-mail* para atualizar seu cadastro, corrigir uma falha, usar um discador para acessar a internet ou ainda para prometer proteger seu computador de vírus quando na verdade ele é o vírus.

- **Pornografia:** é uma modalidade de *spam* bastante antigo e com bastante número de mensagens enviadas diariamente. Duas questões estão relacionadas a este tópico: o recebimento desse tipo de *spam* pelas crianças e a propagação de material de pedofilia. Nesse caso é importante utilizar filtros anti-spam bem aperfeiçoados além de um acompanhamento do *e-mail* da criança.
- **Outros:** alguns *spams* não se enquadram nas categorias anteriores. Trazem tipicamente falsas ofertas de trabalho, conselho ou dicas de como vencer em jogos de azar, como melhorar o desempenho e aumentar a quantidade de visitas no seu *website*, entre outros.

Segundo Teixeira [3], uma nova geração de *spam* surgiu com os novos aplicativos e serviços como o *Spam* sobre Telefone IP (*Spam over* IP *Telephony* – SPIT). Esse tipo de *spam* pode ser um problema grave no futuro, principalmente pela possibilidade de gerar várias chamadas automaticamente com baixo custo.

1.1.1 Técnicas Utilizadas por Spammers

Inicialmente os *spams* eram enviados diretamente aos destinatários, sem qualquer tipo de modificação ou disfarce. Porém com o surgimento e evolução dos filtros anti-*spam*, esse tipo de *e-mail* era facilmente identificado e bloqueado. Com isso, os *spammers* começaram a forjar informações nas mensagens e utilizar endereços falsos.

---

programa hospedeiro para se propagar, já os *worms* é um programa completo e não precisa de outro programa para se propagar.

[3] Trojan Horse ou Cavalo de Troia é um programa que age como a lenda do Cavalo de Troia, entrando no computador e liberando uma porta para um possível invasão e é fácil de ser enviado, é só clicar no ID do computador e enviar para qualquer outro computador.

No fim dos anos 90, servidores de *e-mail* com *relay* aberto[4] eram as principais formas utilizadas pelos remetentes de *spam.* Já a partir do ano 2000, com a popularização das conexões ADSL, cabo, entre outras, os *spammers* começaram a explorar vulnerabilidades nas estações de usuários conectados à internet. Em 2004, foi calculado que a maioria dessas mensagens eram enviadas das maquinas infectadas pertencentes a usuários confiáveis [4].

Essas técnicas de envio de *spam* surgiram devido à necessidade de escapar dos filtros anti-*spam*. Então conforme as empresas melhoram seus filtros, os *spammers* também modificam suas táticas e formas de evitar e escapar desses novos filtros. Ocorrendo assim um padrão circular previsível, com os remetentes de *spam* reinvestindo seus lucros no desenvolvimento de novas técnicas para burlar os novos filtros anti-*spam* [4][2].

[5] Atualmente as principais técnicas usadas pelos remetentes de *spam* para tentar escapar dos filtros são:

1. **Tokenização:** a idéia aqui é que o *spammer* modifica a característica da palavra, adicionando caracteres inválidos ou utilizando alguma acentuação incorreta com o objetivo de enganar o filtro. No entanto essa mudança é de forma que o usuário que está lendo, continue a entender perfeitamente a palavra. Exemplo: s.e.xo; V I A G R A; Gânhè dinheiro fácil;

2. **Texto invisível:** o *spammer* esconde dentro do *e-mail* um texto considerado de mensagem legítima para enganar o filtro. Na hora de

[4] Servidores de *e-mail* que transmitem mensagem de qualquer domínio, para qualquer outro, sem solicitar autenticação e sem restringir a faixa de endereços IP de origem (ANTISPAM.BR, 2009).

visualizar a mensagem, no gerenciador de *e-mail*[5], esse texto não é exibido, sendo visualizada, pelo destinatário, somente o *spam.*

3. **Comentários HTML:** os remetentes escondem as palavras usando comentários HTML ou outras *tags*[6] que serão removidas pelo gerenciador de *e-mail* na exibição da mensagem. Dessa forma, a palavra é exibida corretamente e o filtro é enganado;

4. ***Tags* HTML inválidos:** os remetentes utilizam *tags* inválidas para fazer que os textos considerados legítimos sejam analisados pelo filtro, sendo exibidas no e-mail as mensagens de *spam*.

5. **Mensagem bilíngüe:** a mensagem é enviada em duas línguas distintas, ambas sendo apresentadas na leitura do *e-mail.*

6. **Texto redundante:** uma mensagem legítima é enviada em texto plano e, em HTML, a mensagem *spam.* O gerenciador de *e-mail* por padrão mostra a segunda mensagem. O filtro anti-*spam* analisa as duas e o *e-mail* é considerado legítimo.

7. **Imagens com texto:** Essa técnica envolve enviar o *spam* dentro de uma imagem. Para detectar esse tipo de *spam* é necessário adotar o método de reconhecimento ótico de caracteres (Optical Character Recognition – OCR) que irá escanear as imagens do *e-mail* e identificar as letras e palavras.

---

[5] Gerenciador (ou cliente) de *e-mail* é o programa responsável por carregar as mensagens do servidor e disponibilizá-las ao usuário.

[6] Uma tag é uma palavra-chave (relevante) ou termo associado com uma informação (ex: uma imagem, um artigo, um vídeo) que o descreve e permite uma classificação da informação baseada em palavras-chave.

1.2 Filtros de SPAM

Nessa seção será descritos alguns filtros que normalmente são usados para filtragem de mensagens e classificação das mesmas.

1.2.1 Listas Negras

Alguns provedores de Internet implementam o que se é chamado de listas negras[7]. Estas listas contêm endereços de *e-mail* ou até mesmo domínios relacionados à prática de *spam.* O intuito dessas listas é diminuir o volume de *spams* circulado na rede [6].

O problema com essas listas é que algumas delas[8] bloqueiam completamente um domínio e não apenas o endereço que enviou o *spam.* Assim, usuários deste domínio têm seu e-mail bloqueado, mesmo não sendo *spammers*, o que é bastante prejudicial.

1.2.2 Listas Brancas

Conceito similar porém com a idéia ao contrario das listas negras. Aqui serão colocados os domínios que são autorizados a serem recebidos. E também sofre do problema das listas negras, como são colocados domínios, se algum *spammer* de uma lista branca enviar um *e-mail* com conteúdo de *spam*, o *e-mail* passará normalmente e chegará à caixa postal do usuário.

[7] Exemplos de listas negras podem ser encontradas em http://spamlinks.net/filter-bl.html

[8] A lista MAPS RBL group é conhecida por fazê-lo.

1.2.3 Categorias dos filtros existentes

A idéia dos filtros surgiu através da utilização de palavras chaves, ou seja, se contiver determinado termo apresentado no cabeçalho ou no corpo do *e-mail*, a mensagem é marcada como *spam.* O maior problema desse método é que o mesmo não é escalável, pois para cada nova palavra relacionada a *spam*, a mesma deverá ser incorporada manualmente ao filtro. Os filtros, de acordo com Haykin, ainda podem ser categorizados em [7]:

- *Filtros definidos pelos usuários:* permitem a criação de regras na forma: "IF____ *campos contêm* ___*, mover para a pasta* ___". O problema desse método é que fica inviável está criando uma regra do tipo para cada novo tipo de *spam* criado.

- *Filtros de cabeçalhos:* analisam os cabeçalhos para detectar se são falsos ou não.

- *Filtros de conteúdo:* varrem todo o corpo da mensagem de *e-mail* e utilizam-se da presença, ausência e freqüência de palavras para determinar se determinada mensagem é ou não *spam.* Esse tipo de filtro é encontrado na grande maioria dos filtros de detecção de *spam* encontrados atualmente.

1.2.4 Algoritmos Naive Bayesian

O seu uso é bastante comum em problemas de classificação de textos. A idéia básica é usar a probabilidade na estimativa de uma dada categoria presente em um documento ou texto. No caso deste método, *naive*[9] é usado porque o modelo assume que existe independência entre as palavras, isto é, a probabilidade de ocorrer uma palavra não depende da existência de outra(s). Esta solução já foi mostrada nos trabalhos de Özgür, Güngör e Gürgen, Androutsopoulos et al e Zhang

[9] *Naive* em inglês quer dizer ingênuo, sem conhecimento.

et al [8][9][10], e já é utilizada em alguns programas, como por exemplo, o gerenciador de *e-mail* Mozilla Thunderbird[10].

1.2.5 *Support Vector Machine (SVM)*

São conjuntos de treinamento supervisionados utilizados para classificação e regressão. Pertencem a uma família de classificadores lineares generalizados. Sua propriedade é minimizar simultaneamente o erro na classificação empírica e maximizar a divisão geométrica entre as classes. O SVM já foi utilizado por Zhang et al, Drucker, Wu e Vapnik e Hidalgo [10][11][12].

1.2.6 K-Nearest Neighbour (k-NN)

O k-NN, ou k-vizinhos mais próximos, é um método usado para reconhecimento de padrões. A idéia básica é que, dado um documento qualquer na entrada, o sistema dá uma pontuação para seus vizinhos mais próximos entre os documentos de treinamento, e usa as categorias dos *k* documentos com maior pontuação para adivinhar a categoria do documento de entrada. Os pesos às quais os vizinhos pertencem são então ajustados, usando pontuações próximas, e a soma total da pontuação dos k-vizinhos mais próximos é usada para classificar os documentos [13]. O k-NN é usado por Yang e Pedersen.

1.2.7 Árvores Boosting

A idéia aqui destes algoritmos é encontrar uma regra de classificação muito precisa e combiná-la com hipóteses básicas, ou, digamos, pouco precisas. Um algoritmo particular é chamado de AdaBoost. A idéia dele é manter um conjunto de pesos relevantes dos padrões de treinamento, utilizando-os para forçar o aprendizado a

[10] Mais detalhes do filtro podem ser obtidos em http://wiki.mozilla.org/thunderbird2:junkmail

concentrar-se nos exemplos de mais difícil classificação [10][14]. O AdaBoost foi utilizado por Zhang, Zhu e Yao e Carreras e Marquez.

1.2.8 Aprendizado baseado em memória

É um método que armazena dados de treinamento em uma estrutura de memória. O método assume que o ato de raciocinar é baseado diretamente na reutilização de experiências armazenadas em vez do conhecimento adquirido [10][15][16]. Esse método foi empregado por Zhang, Zhu e Yao e Sakkis et al.

1.2.9 Sistemas Imunológicos Artificiais

Utilizam heurísticas do aprendizado de máquina baseado no sistema imuno-biológico. Segundo De-Castro, é possível relacionar as características de imunovigilância e resposta imune aos procedimentos de segurança computacional [17][16], que podem incluir detecção de *spam*, eliminação de vírus e intrusos de rede. Esse método foi utilizado por Guzella et al.

1.2.10 Self-Organizing Maps (SOM)

O Mapa Auto-Organizável de Kohonen é uma RNA com duas camadas [18]: a camada de entrada *I* e a de saída *U.* O SOM foi idealizado a partir da analogia com a região do córtex cerebral humano. Destacam-se as potencialidades de visualização de dados multivariados, análise de agrupamentos, mineração de dados, descoberta de conhecimento e compressão de dados.

1.3 Problema do Falso Positivo

Um dos principais problemas enfrentados na detecção de *e-mail* é quanto às classificações incorretas, ou seja, *e-mails* legítimos classificados como *spam* ou vice-versa. Os erros de classificação podem ser separados em dois tipos:

- **Falso Negativo:** acontece quando o filtro anti-*spam* detecta um *spam* como se fosse uma mensagem legitima, de modo que o usuário recebe em sua caixa de mensagens um *spam* que foi considerado como mensagem legítima.

- **Falso Positivo:** ocorre quando uma mensagem legítima é classificada como *spam.*

Apesar de trazer alguns aborrecimentos e inconvenientes, um falso negativo não é um problema grave pois basta o usuário apagar a mensagem recebida incorretamente pelo filtro anti-*spam* de sua caixa de entrada do *e-mail*. Já o caso contrário, ou seja, o falso positivo pode se tornar um grave problema, já que uma mensagem legítima pode ser filtrada pelo anti-*spam* podendo trazer sérios prejuízos ao usuário que não receberá o *e-mail* legítimo. Alguns filtros permitem que o *e-mail* classificado como *spam* seja enviado a uma pasta especifica para posterior análise do usuário, o que evita sérios transtornos ao usuário.

1.4 Caracterização do Problema

1.4.1 O Problema Abordado

O problema central dessa pesquisa pode ser resumido na classificação de *e-mails* em legítimos ou *spams.* Características relevantes das duas classes de *e-mail*s (*spam* e legítima) devem ser identificadas no processo de classificação.

Devido a sua capacidade de generalização, as redes neurais artificiais mostram-se bastante eficazes para abordar o problema, empregando treinamento por meio de exemplos. A topologia da rede é determinada pela organização em camadas dos elementos de processamento das RNA, os neurônios [6]. Já a arquitetura das RNA define os modelos de organização com características especificas como, por exemplo, as do tipo MultiLayer Perceptron (MLP);

1.4.2 Relevância do Tema

Algumas estatísticas realizadas mostram que mais de 60% dos usuários de *e-mail* dizem ter perdido a confiança no *e-mail* por causa de *spam*, como a pesquisa realizada pela IronPort (2008). A pesquisa informa que o volume diário de *spams* é superior a 120 bilhões, o que significa aproximadamente 20 mensagens de *spam* por dia para cada pessoa no planeta. Não bastasse isso, o *spam* fica a cada dia mais perigoso, ou seja, o que antes era apenas uma propaganda de um produto, hoje contém URL com executável para o usuário baixar, instalar e disseminar o vírus.

O crescimento do número de *spams* pode ser visto na Figura 1 que mostra a quantidade de *spams* recebida por um único usuário com múltiplos endereços de *e-mail.*

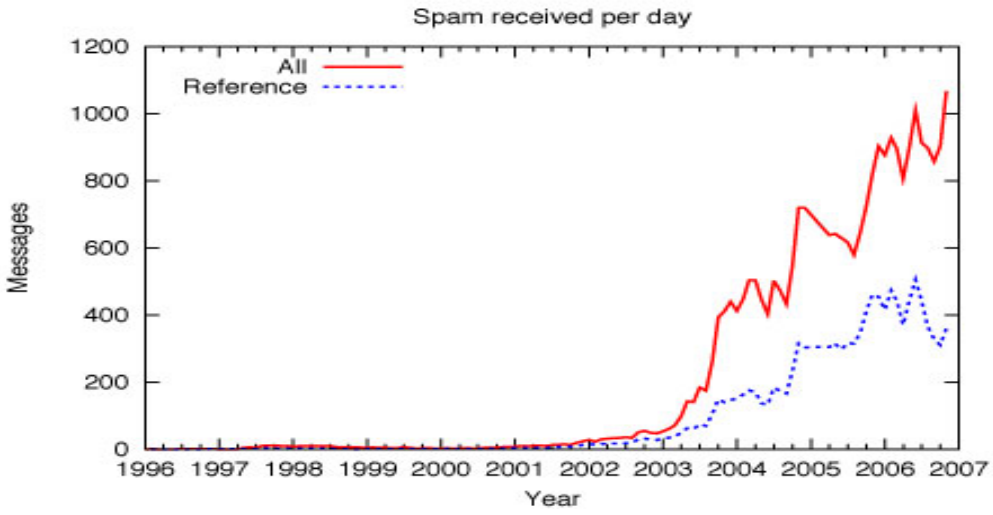

Figura 1.1: Estatística de spams recebidos por dia ate 2007
FONTE – Spamnation – http://spamnation.info/stats/ acessado em 2009

Já a estatística de *spams* recebidos e reportados ao SpamCop (2009) de novembro de 2008 a outubro de 2009 é apresentado na Figura 2. Já a publicada pelo CertBR (2009) relativa ao ano de 2003 a setembro de 2009 é mostrada na Figura 3 e também evidencia essa estatística. Vale lembrar que esses dados não se referem ao numero total de *spams* que circulam na rede e sim aos reportados ao SpamCop e Cert.BR. Porém eles são bastante representativos considerando o total de mensagens.

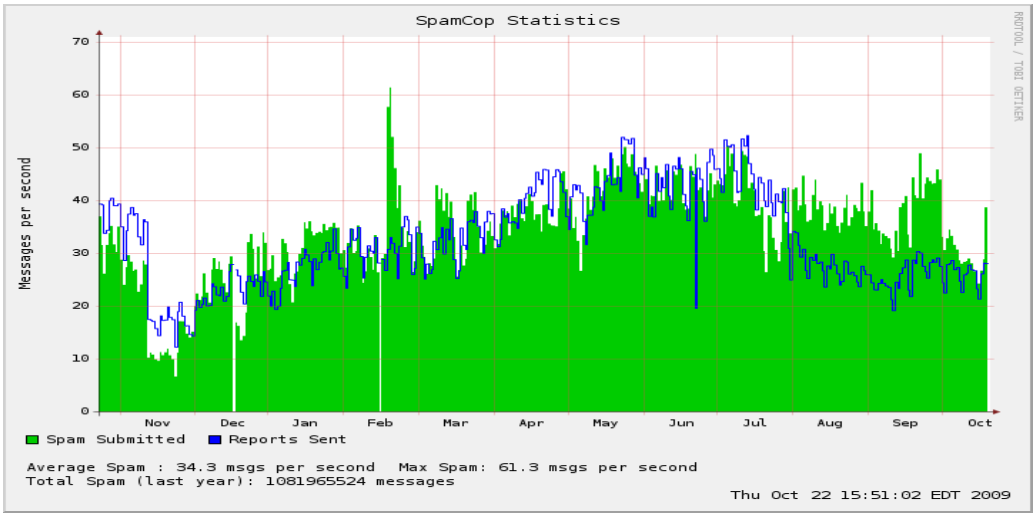

Figura 1.2: Spams reportados ao SpamCop por segundo – Nov/2008 a out/2009
FONTE – SpamCop (2009) – http://spamcop.net/spamgraph.shtml?spamyear

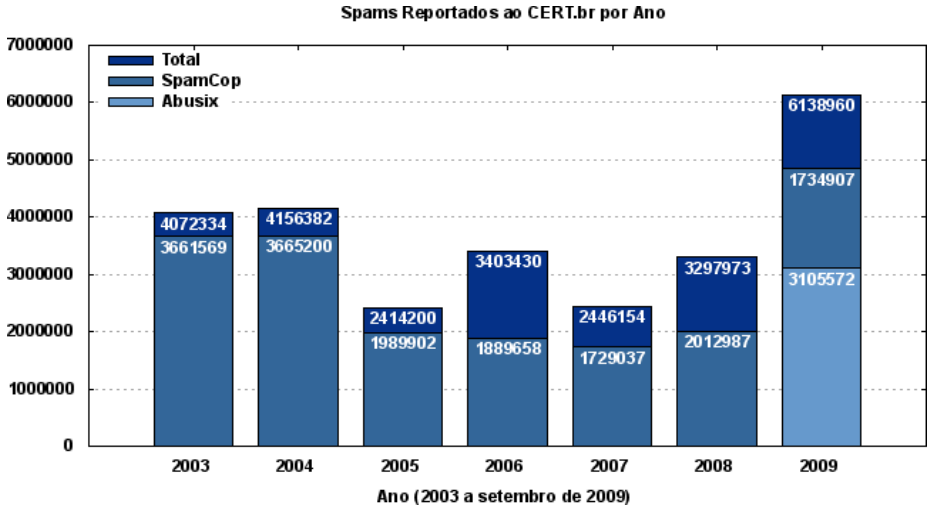

Figura 1.3: Spams reportados ao Cert.br por ano
FONTE – CertBR (2009) – http://www.cert.br/stats/spam

Conforme já visto, os *spams* podem causar muitos prejuízos às pessoas e às empresas. Em 2002, o custo para usuários não corporativos foi de aproximadamente U$ 230 milhões e para usuários corporativos foi de aproximadamente U$ 9 bilhões. Outro dado interessante é que na pesquisa, aproximadamente 16% dos usuários trocam de *e-mail* devido ao *spam* e 8% compram por meio do *spam.*

Grande parte dos *spams* são enviados por *botnets*, redes de computadores infectados por bot[11] e utilizadas para aumentar o potencial de envio de *spam*, sendo que um pequeno número de *botnets* é responsável por um alto volume de *spams*. A Figura 4 mostra os principais *botnets* e o percentual de envio de *spams* a cada um deles.

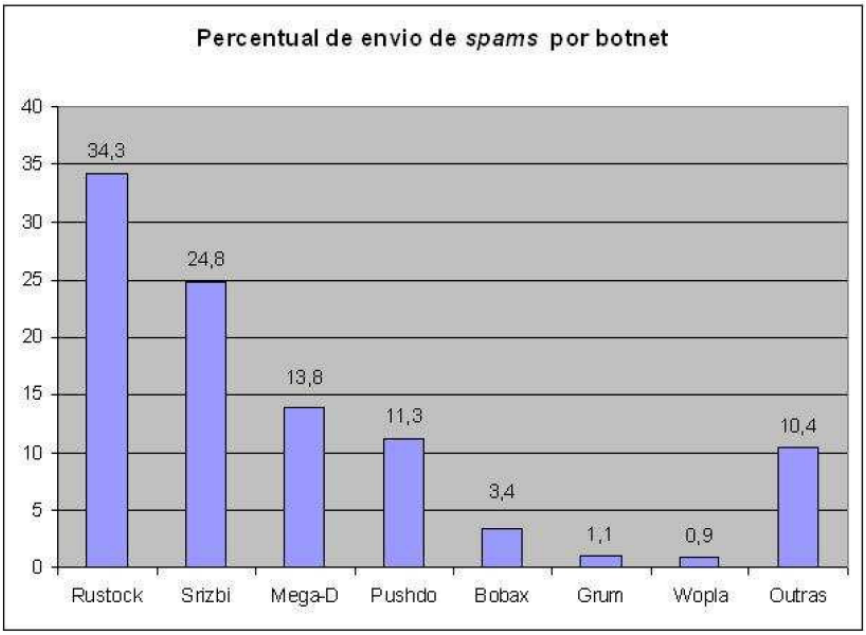

Figura 1.4: Percentual de envio de *spams* por *botnet.* Adaptado de Marshal (2008) – http://www.m86security.com/trace/spam_statistics.asp

Na figura 1.5 é mostrado um gráfico representando o percentual de *spams* enviados de acordo com sua categoria. Conforme visto, o maior número de *spams* são enviados contendo assunto relacionado à saúde com quase 74%. Enquanto a

[11] Bot é um programa de computador capaz de se propagar automaticamente pela rede, explorando vulnerabilidades ou falhas em aplicativos.

Figura 1.6 mostra por país e vemos que o Brasil representa um alto índice de produção de *e-mails* com conteúdo *spam*.

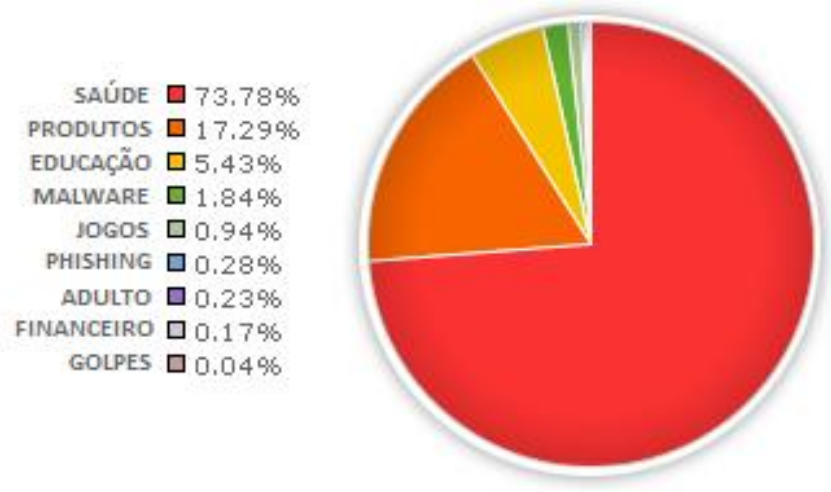

Figura 1.5: Percentual de *spams* por categoria. Adaptado de Marshal (2008).
Fonte: http://www.m86security.com/trace/spam_statistics.asp - Acesso em 11/2009

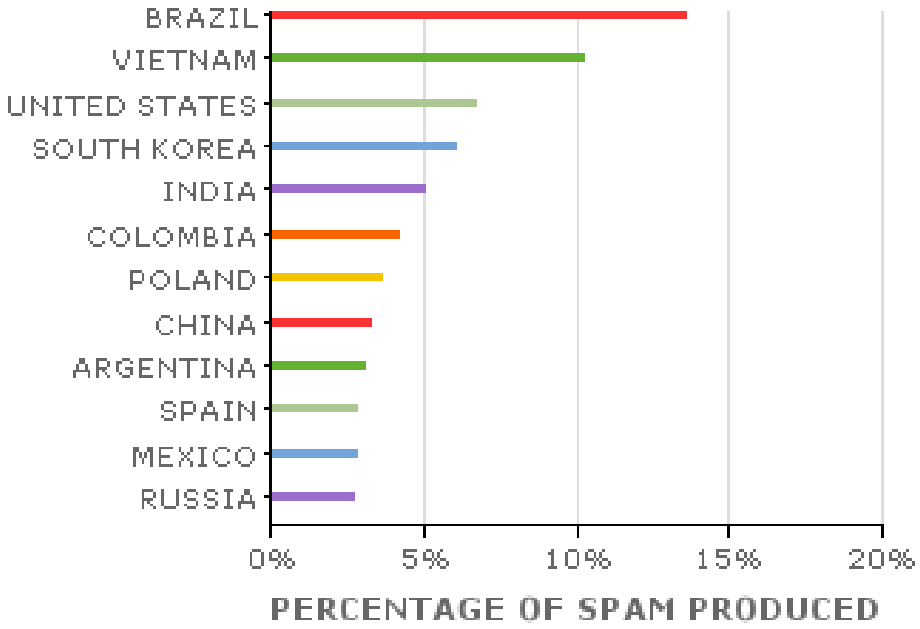

Figura 1.6: Percentual de *spams* por país.
Fonte: http://www.m86security.com/trace/spam_statistics.asp - acesso em 11/2009

Objetivos e Estrutura do Trabalho

O objetivo desse Trabalho de Conclusão de Curso (TCC) visa mostrar como se comporta a classificação de um filtro anti-*spam* utilizando três algoritmos distintos para treinamento: o SOM, o MLP e o RBF.

A idéia é fazer testes e verificar, dado uma base de dados, qual dos algoritmos apresentados vai se adequar melhor ao problema proposto, ou seja, qual o percentual de acertos. Falso positivo e falso negativo apresentado por cada algoritmo.

Capítulo 2

Redes Neurais Artificiais

Redes Neurais Artificiais é a tentativa do homem de criar computadores que imitem ou se comportem como o homem. Porém a ciência ainda não conseguiu reproduzir perfeitamente o comportamento desejado, principalmente no que se trata da visão, comportamento e fala.

As RNA têm sido empregadas com sucesso em diferentes processos de classificação. Neste trabalho, são utilizadas na classificação de *e-mails* como legitima ou *spams*. Será apresentado e explicado como é o funcionamento dos algoritmos que será usado no trabalho e um pouco das aplicações de redes neurais em reconhecimento de padrões.

As RNA são modelos matemáticos que se assemelham às estruturas neurais biológicas e têm a capacidade computacional adquirida por meio de aprendizagem e generalização. A idéia é que esses modelos se assemelhem com o sistema nervoso dos seres vivos e com a sua capacidade de processar informações. De acordo com Haykin, a rede neural pode ser vista como [7]:

> um processador maciçamente paralelamente distribuído construído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso. Ela se assemelha ao cérebro em dois aspectos:
>
> 1. O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem.
> 2. Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

Uma rede neural é definida como um conjunto de neurônios artificiais organizados em camadas, na qual podemos identificar a entrada, os neurônios internos e os de saída. A sua arquitetura é definida pela forma na qual esses neurônios estão organizados e interconectados, ou seja, o número de camadas, o número de

neurônios por camada, tipos de conexão entre os neurônios e a topologia da rede. Nas próximas seções serão explicados os algoritmos usados nesse trabalho.

2.1 Neurônio Biológico

O paradigma das Redes Neurais Artificiais surgiu com o crescente interesse no conhecimento do cérebro e mente humana e com isso o aumento no interesse em pesquisar o papel do funcionamento do Sistema Nervoso (SN), o qual motivou a construção de modelos matemático-computacionais que pudessem auxiliar nos aspectos neurobiológicos envolvidos em diversas atividades cognitivas.

Figura 2.1: Imagem de parte do SN humano
Fonte: http://fu2re.wordpress.com/2009/06/16/neuronio/ acesso em 11/2009.

Os neurônios são as células básicas do SN. Existem dois tipos principais de neurônios, os que realizam processamento e os que interconectam diferentes partes do cérebro entre si, ou conectam o cérebro a outras partes do corpo.

Individualmente, os neurônios são simples e não são capazes de fazer muita coisa, são simples, porém a conexão entre eles proporciona uma diversidade enorme de tarefas realizadas pelo Sistema Nervoso.

De forma geral, os axônios são responsáveis pela transmissão do impulso nervoso a outros neurônios, os dendritos relacionam-se com a captação desses estímulos, enquanto que no corpo celular encontram-se o núcleo e organelas responsáveis pelas demais atividades da célula.

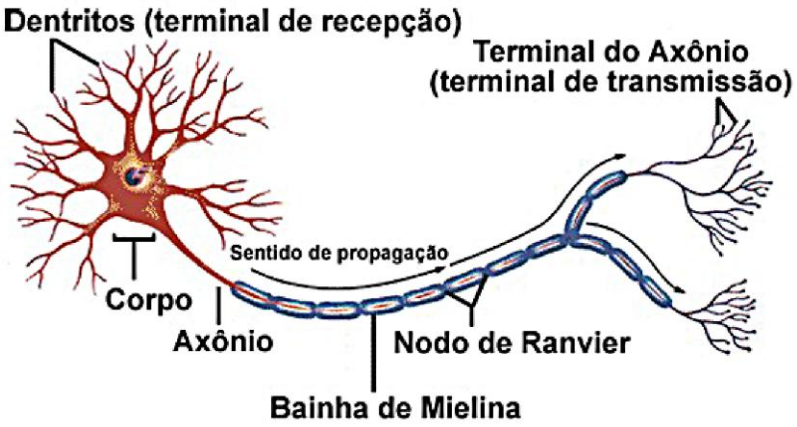

Figura 2.2: Estrutura básica de um neurônio
Fonte: http://www.brasilescola.com/biologia/sistema-nervoso.htm - acesso em 11/2009

2.2 Redes MultiLayer Perceptron - MLP

As redes neurais MultiLayer Perceptron (MLP), ou Perceptron de Múltiplas Camadas, são redes que apresentam uma ou mais camadas intermediárias, comumente chamadas de camada oculta ou escondida.

Cada uma das camadas da MLP possui sua função específica. A camada da entrada é composta por unidades sensoriais e é responsável pela recepção e propagação das informações de entrada para a camada seguinte. Já as camadas intermediárias são compostas por unidades que transmitem as informações por meio das conexões entre as unidades de entrada e saída. A camada de saída é formada por neurônios que recebem os dados da camada intermediária, fornecendo a resposta à rede.

A arquitetura de uma rede MLP pode ser caracterizada assim:

- Fluxo de sinal através da rede é unidirecional, da esquerda para a direita, camada a camada;
- A rede é do tipo totalmente conectada, qualquer camada da rede está conectada a todas as outras unidades na camada anterior;
- O modelo de cada neurônio da rede inclui uma função de ativação não linear e diferencial em qualquer ponto;
- O processamento realizado por cada neurônio é definido pela combinação dos processamentos realizados pelos neurônios da camada anterior que estão conectados a ele.

As redes MLP apresentam um poder computacional muito maior que as redes sem camadas intermediárias [19][18]. O numero de camadas intermediárias é determinado pelo tipo de função a ser aproximada. Um uso elevado no número de camadas não é recomendado, pois, a cada vez que o erro no processo de treinamento é retropropagado, ele se torna menos útil e preciso. Segundo ainda Braga, a única camada que tem uma noção precisa do erro é a camada de saída [19].

Quanto ao número de neurônios, o uso de muitas unidades pode levar a rede a memorizar os padrões de treinamento em vez de extrair as características gerais que permitirão a generalização ou o reconhecimento de padrões não vistos durante o treinamento.

A função de ativação representa o efeito que a entrada interna e o estado atual de ativação exercem na definição do próximo estado de ativação da unidade. Existem várias funções de ativação, como algumas exemplificadas a seguir:

- **Função Linear:** repete o sinal de entrada do neurônio na sua saída. É bastante utilizada nos neurônios da camada de saída da rede. A sua expressão é dada pela Equação 2.1:

$$y = f(x) = x \tag{2.1}$$

- **Logarítmica Sigmoidal:** é um tipo de função logística não linear. Aqui os pesos sinápticos são modificados mais intensamente para aqueles neurônios da rede onde os sinais funcionais estão no meio do seu intervalo [7][18][19]. Essa função possui um intervalo entre zero e um, já que a entrada do neurônio vai de negativo até infinito. A sua expressão é dada pela Equação 2.2:

$$y = f(n) = \frac{1}{1 + e^{-n}}. \tag{2.2}$$

- **Tangente Sigmoidal:** é a função logística reescalada e modificada por um bias [20]. Esta gera as saídas entre -1 e 1, enquanto a entrada do neurônio vai de negativo até infinito. A expressão é dada pela Equação 2.3

$$y = f(n) = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}} \tag{2.3}$$

Esse algoritmo de treinamento é do tipo supervisionado. Nesse processo, é apresentado à rede um conjunto de padrões de treinamento, composto de valores de entrada e as respectivas saídas desejadas. O objetivo aqui é ajustar os pesos sinápticos de forma a minimizar o erro entre a resposta desejada e a saída real da rede. Essa minimização do erro é incremental, pois os ajustes nos pesos são feitos a cada etapa de treinamento. A forma genérica para alteração dos pesos por correção de erros é calculada pela Equação 2.4:

$$w_i(t+1) = w_i(t) + \eta\epsilon(t)x_i(t). \tag{2.4}$$

2.2.1 O Backpropagation

O Backpropagation é uma generalização da Regra Delta proposta por Widrow-Hoff. O algoritmo utiliza o principio da minimização de uma função custo, no caso, a soma

dos erros médios quadráticos sobre um conjunto de treinamento, utilizando a técnica do gradiente descendente. A principal mudança em relação à Regra Delta é a utilização de funções contínuas como função de saída dos neurônios ao invés da função de limiar lógico. O uso de funções contínuas permite a utilização de busca do gradiente descendente para os elementos das camadas escondidas por ser deriváveis.

Os sinais são propagados pela rede, camada por camada, até a apresentação do resultado na camada de saída [19]. O resultado obtido é comparado com o desejado. Se a saída obtida não estiver correta, o erro é calculado e então o erro é retropropagado da camada de saída para a camada de entrada.

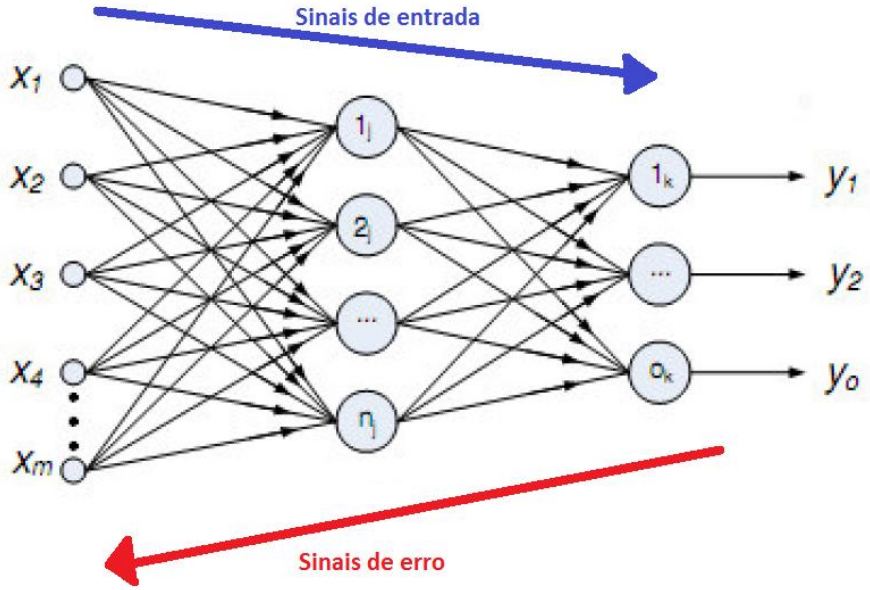

Figura 2.3: Sinais de entrada e retropropagação dos erros.
Fonte: Adaptação de Almeida - Programa de Mestrado em Modelagem Matemática e Computacional,cursada em 2006.

2.2.2 Validação Cruzada

Um dos grandes problemas que podem acontecer durante o treinamento de uma rede neural é o chamado *overfitting*, ou sobre-treinamento. Esse fenômeno acontece quando o erro do conjunto de dados de treinamento continua diminuindo e, em algum ponto desse processo, a rede começa a diminuir sua taxa de acertos. De

acordo com Braga, existem duas formas ou alternativas que podem ser utilizadas para evitar esse sobre-treinamento [19]: poda ou *prunning*, que elimina pesos ou neurônios irrelevantes; e validação cruzada, que encerra o treinamento quando o erro de validação começa a subir, indicando que a rede esta aprendendo com ruído nos dados.

O conjunto de dados é dividido aleatoriamente em três subconjuntos: treinamento, validação e teste. O conjunto de treinamento deve ser utilizado para aprendizagem da rede, buscando otimizar os pesos. O conjunto de teste é utilizado para interromper o processo de treinamento quando for observado que o erro médio quadrado (EMQ), calculado pela Equação 2.5, para os dados de teste começa a aumentar de forma continua, assim temos:

$$EMQ = \frac{\sum_{i=1}^{k}(di - yi)^2}{2}, \tag{2.5}$$

Onde *k* é o numero de neurônios na camada de Said, y*i* a saída atual da rede e *di* é a saída desejada para o neurônio i. Finalmente o terceiro conjunto é utilizado para testar a capacidade de generalização da rede.

2.3 Redes Self-Organizing Maps - SOM

Na década de 80, Teuvo Kohonen desenvolveu uma rede que é chamada de redes SOM (Self-Organizing Maps – Mapa Auto-Organizável). Esse tipo de rede possui uma forte inspiração neurofisiológica, sendo baseada no mapa topológico presente no córtex cerebral.

Haykin [7] caracteriza um mapa auto-organizável

> pela formação de um mapa topográfico dos padrões de entrada, no qual as localizações espaciais (i.e. coordenadas) dos neurônios na grade são indicativas das características estatísticas intrínsecas contidas nos padrões de entrada, daí o nome “mapa auto-organizável”.

Esse tipo de rede usa o treinamento não-supervisionado. Aqui, o treinamento é somente os padrões de entrada que estão disponíveis para a rede. Com esses padrões, a rede busca estabelece uma harmonia com as regularidades estatísticas dos dados de entrada. Com isso, desenvolve a habilidade de formar representações internas e criar novas classes ou grupos automaticamente.

Analisando os dados de entrada, a rede determina algumas propriedades dos dados e aprende, refletindo as propriedades na saída da rede para que seja possível encontrar padrões e características nos dados de entrada. As redes SOM utilizam o método do aprendizado competitivo, baseado na competição entre as unidades de saída para serem ativadas. Assim, apenas um neurônio de saída ou grupo de neurônios vizinhos fornece uma resposta ativa. Isso é chamado de *Winner-Takes-All (WTA)* – o vencedor leva tudo e tem como base uma função onde o nível de ativação indica a similaridade entre o vetor de pesos do neurônio e o vetor de entrada [18].

O grau de similaridade entre o vetor de pesos do neurônio e o vetor de entrada da rede é medido calculando a distância euclidiana ou distância de *Manhattan.* Uma vez que a distância entre o vetor de pesos de um determinado neurônio e o vetor de entrada é mínima para todos os neurônios da rede, esse neurônio terá seu peso atualizado. A Equação 2.6 mostra a forma como os pesos do neurônio vencedor e de seus vizinhos são atualizados.

$$w_{ji}(t+1) = \{ \frac{w_{ji}(t) + \eta(t)(x_i(t) - w_{ji}(t0),\ se\ j \epsilon \Lambda(t)}{w_{ji}(t),\ caso\ contrario} \tag{2.6}$$

Onde w_{ji} *(t)* é o peso da conexão entre o elemento de entrada *x(t)* e o neurônio *j, n(t)* é a taxa de aprendizado e Λ é a vizinhança do neurônio vencedor em um instante de tempo *t.*

Podemos resumir o algoritmo de treinamento para as redes SOM da seguinte maneira: inicializar pesos e parâmetros; para cada padrão de treinamento é definido o neurônio vencedor, seus pesos e de seus vizinhos são atualizados, se o numero de ciclo for múltiplo de N, a taxa de aprendizado é reduzida e também a área de vizinhança. Esse processo ocorre até que o ajuste dos pesos não seja mais significativo e o mapa de características pare de mudar [19].

O treinamento da rede SOM ocorre em duas fases: na primeira ocorre o treinamento da rede para organizar os dados, de forma que os mais parecidos fiquem próximos de si. Para isso, quando um padrão de entrada *p* é apresentado, a rede procura a unidade mais parecida com *p* e assim, constrói um mapa topológico, em que os nós topologicamente próximos respondem de forma semelhante a padrões de entrada semelhantes. A segunda fase é a de classificação, em que a rede SOM utiliza o mapa organizado para identificar a classe mais próxima à entrada. No final desse processo, cada neurônio ou conjunto de neurônios vizinhos representa um padrão distinto dentro do universo de padrões de entrada da rede.

2.4 Função de Base Radial (RBF)

A Função de Base Radial surgiu em 1988 como uma possível alternativa às redes MLP. Na sua formulação geral, a rede neural RBF possui uma camada de entrada(chamado de nós sensoriais), uma camada de nós escondidos e uma camada de saída. Neste tipo de rede, a ativação de um nó é uma função da distância entre os vetores de entrada e seus pesos, por isso seu nome Funções de Base Radial [20]. Na saída dessa rede, os parâmetros ajustáveis são os pesos de uma combinação linear.

Assim como as MLP, as Redes Neurais de Função de Base Radial (RBF) são redes neurais com neurônios ocultos. Portanto, as redes de Função de Base Radial(RBF) podem também resolver problemas do tipo não - linearmente separáveis, como por exemplo o XOR.

As principais diferenças entre as Redes RBF e as Redes MLP são [18]:

- Os neurônios da camada intermediária têm apenas funções de base radial como função de ativação, que são funções localizadas de tal maneira que apenas algumas unidades escondidas ficarão ativadas ao receberem um dado conjunto de exemplos de entrada;

- Nas redes MLP, as funções de ativação têm como entrada líquida, uma média ponderada entre os exemplos de entrada e o conjunto de pesos.

Por outro lado, nas redes RBF a utilização destas funções de ativação de base radial fazem com que a sua ativação seja obtida a partir de uma norma ponderada da diferença entre o vetor de entrada e o centro da função de base radial;

- A camada de saída é composta por unidades de processamento lineares.

As redes RBF constroem aproximações locais de entrada-saída, resultando em sensibilidade reduzida à ordem de apresentação dos dados do treinamento [11][14].

As funções de ativação (f_{oi}) mais utilizadas são [18]:

Tabela 1: funções de base radial normalmente utilizadas.

Lâmina *spline* fina	$\phi(\zeta)=\frac{\zeta}{\sigma^2}\log\left(\frac{\zeta}{\sigma}\right)$
Multi-Quadrática	$\phi(\zeta)=\sqrt{\zeta^2+\sigma^2}$
Multi-quadrática inversa	$\phi(\zeta)=\frac{1}{\sqrt{\zeta^2+\sigma^2}}$
Gaussiana	$\phi(\zeta)=\exp\left(-\frac{\zeta^2}{2\sigma^2}\right)$

A função de base radial do tipo Gaussiana é a mais utilizada na prática. O parâmetro σ é o desvio-padrão da função gaussiana. Dessa forma, σ define a distância Euclidiana média que mede o espalhamento dos dados representados pela função de base radial em torno de seu centro.

O raio de cada uma das funções de base radial de uma rede RBF pode assumir diferentes valores, no entanto, para redes RBF reais, o mesmo raio utilizado para cada neurônio não-linear já permite que a rede uniformemente aproxime qualquer função contínua, desde que haja número suficiente de funções de base radial. Na prática, o valor do raio das funções de base radial afeta as propriedades numéricas dos algoritmos de aprendizado, mas não afeta a capacidade geral de aproximação das redes RBF [21][22].

O treinamento das redes RBF pode ser realizado de diversas formas. Entre as várias propostas, o treinamento mais utilizado é o treinamento híbrido, por utilizar uma aprendizagem não supervisionada, para determinar os parâmetros das funções de base radial da camada escondida e um aprendizado supervisionado para ajustar os pesos que ligam a camada escondida a de saída. Portanto, desde que as funções de base radial, após determinação de seus parâmetros, sejam considerados fixas, o ajuste dos pesos que ligam a camada escondida para a camada de saída para a rede RBF fica equivalente a uma rede ADALINE. Neste caso, ao se utilizar neurônios lineares na camada de saída e uma função objetivo como o erro médio quadrático, o treinamento destas redes é bastante rápido, uma vez que dispomos de um sistema de equações lineares para ser resolvido, o que nos permite utilizar técnicas lineares de inversão de matriz [18].

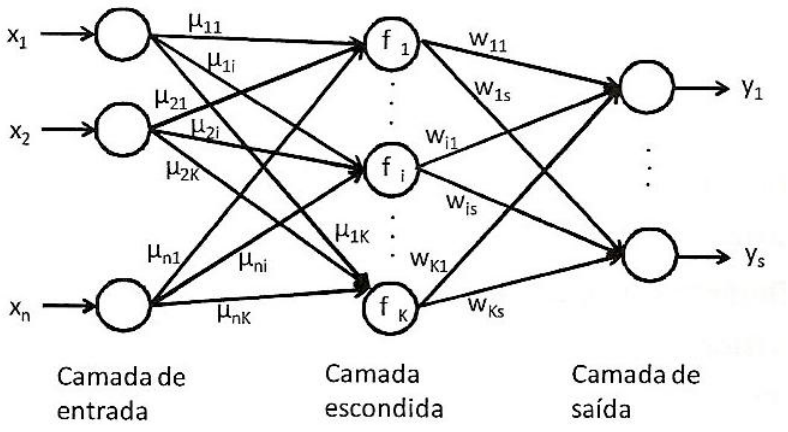

Figura 2.4: Função de Base Radial – RBF [18].

A rede RBF pode fazer uma aproximação de qualquer função contínua através da combinação linear de funções gaussianas com centros em diferentes posições do espaço de entrada, como pode ser visto na Figura 2.5.

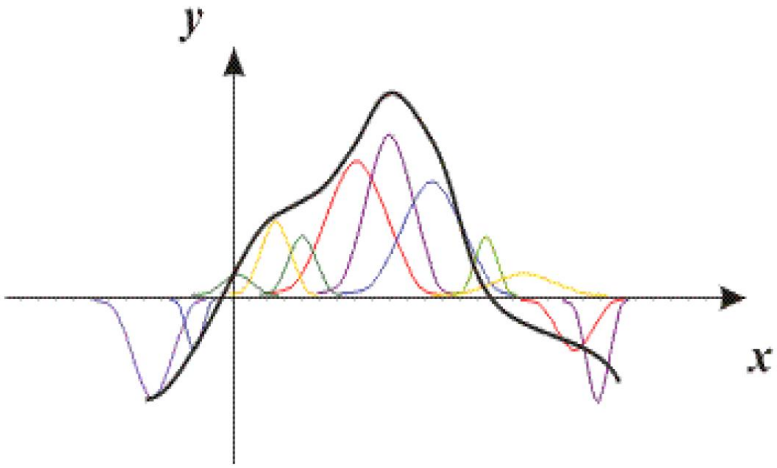

Figura 2.5: Aproximação de Funções 1-D usando a rede RBF.
Fonte: Guilherme, 2008 – http:// www.deti.ufc.br/~guilherme

Na figura 2.6 vemos como se comporta o gráfico para o caso 2-D.

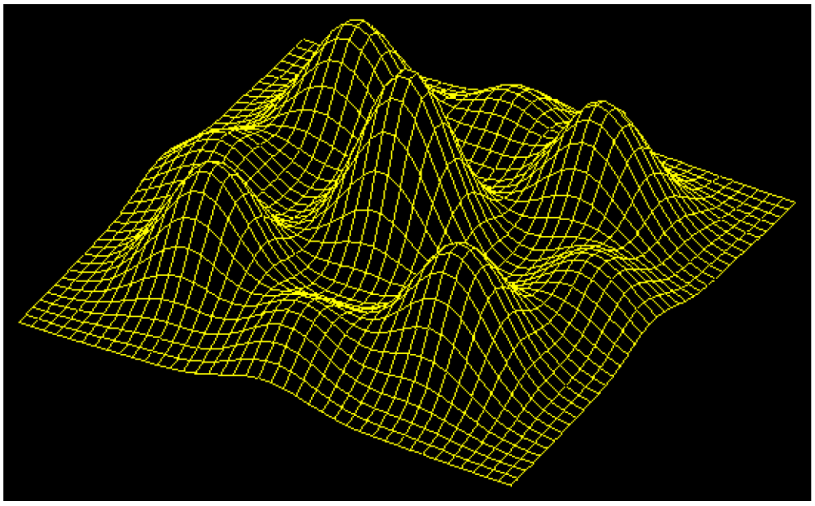

Figura 2.6: Aproximação de Funções 2-D usando a rede RBF.

2.5 Formas de aprendizado

As RNAs precisam de um conjunto de exemplos, ou base de dados, fornecidos à rede para o treinamento. O algoritmo de aprendizado é o principal responsável pela aquisição de conhecimento da rede. O algoritmo de aprendizado é um conjunto de regras que ajusta os parâmetros do aprendizado da rede. Existem basicamente três formas de o conhecimento ser adquirido: aprendizagem supervisionada, não-supervisionada e reforço [19].

1. Aprendizagem supervisionada: aqui a entrada e a saída desejada é fornecida para a rede através daquilo que chamamos de supervisor. Um padrão de entrada é fornecido à rede e a sua saída atual é comparada com a saída desejada e a diferença é aplicada para o ajuste dos pesos da rede. O processo de minimização do erro é que garante a eficiência da RNA. Porém, a presença do supervisor tem relação com o sucesso do aprendizado. Na ausência do mesmo, a rede é incapaz de progredir em apresentar respostas para o problema.

2. Aprendizado não-supervisionado (auto-organização): como o nome sugere, não precisa do supervisor, ou seja, um agente externo que indica a resposta desejada para um determinado padrão de entrada. Aqui utilizasse padrões redundantes para que a rede responda de maneira semelhante.

3. Aprendizado por reforço: difere do algoritmo supervisionado por não fornecer à rede a resposta desejada, a informação apresentada a ela é se a saída está correta ou não. Como exemplo, uma criança comete algum erro e os pais oferecem indicações para mostrar que o comportamento apresentado pela criança está errado. Neste caso, o aprendizado por reforço considera o ambiente em que se estar inserido para reforçar o aprendizado.

Capítulo 3

Experimentos

Este capítulo detalhará as técnicas e elementos utilizados no desenvolvimento deste trabalho, como: a Base de Dados, os Algoritmos de IA.

3.1 A Base de Dados

A base de dados utilizada neste trabalho foi obtida atrás do Repositório de Aprendizagem de Máquina da Universidade da Califórnia Irvine – UCI-R [21].

Esta base específica foi escolhida pela disponibilidade de acesso e por já ter sido utilizada em diversos trabalhos sobre detecção de *spam* [23], inclusive em trabalhos onde é feita uma análise comparativa de diferentes algoritmos. Construir uma base de dados própria demandaria um tempo considerável e exigiria tarefas adicionais de representação e do pré-processamento dos atributos de entrada. Como a idéia deste trabalho visa fazer uma análise comparativa das diversas técnicas de detecção, a escolha de uma base pronta, que apresenta o necessário e suficiente grau de confiabilidade no cenário de *spam* facilita o desenvolvimento deste trabalho.

3.1.1 Estrutura da Base de Dados

A base de dados utilizada foi obtida através de uma coleta de *e-mails* pessoais e de trabalho, onde os *spams* são originados de caixas postais individuais e a palavra “George” é uma indicação de não-*spam*. A base foi gerada e pré-processada no período compreendido entre Junho e Julho de 1999, por Jaap Suermondt, Erik Reeber, Mark Hopkins e George Forman e sendo muito utilizado para a construção de filtros de *spam* personalizados [21].

Esta base de dados possui a representação de 4601 mensagens de *e-mail*, onde 2788 mensagens (60,6%) representam os *e-mails* não-*spam* e 1813 mensagens (39,4%) representam *spam.* O arquivo de dados que representa esta base de dados (*spambase.data)* possui 57 atributos de entrada, onde a maior parte destes atributos é indicação de frequência de ocorrência de palavra ou caractere em particular no corpo do *e-mail.* Esta base possui apenas um atributo que serve para a classificação do *e-mail*, indicando "*1"* para *spam* e "*0"* para não-*spam*.

A seguir a descrição dos atributos:

- 48 atributos [zero a cem] são do tipo freqüência de palavra, indicando a porcentagem de palavras no *e-mail* referentes às palavras associadas ao *spam:* (número de vezes que uma palavra aparece no *e-mail*) / (total do número de palavras do *e-mail*). Uma *palavra*, neste caso, é uma cadeia de caracteres alfanuméricos ou uma marca de final da cadeia.

- seis atributos representando a freqüência de caracteres no *e-mail* relacionando a porcentagem de caracteres no *e-mail* que estão associados a um caractere em específico: (número de ocorrência dos caracteres) / (total de caracteres no *e-mail).*

- um atributo com a maior sequência ininterrupta de letras maiúsculas.

- um atributo relacionado à média de caracteres maiúsculos

- um atributo com o total de letras maiúsculas no *e-mail.*

- um atributo representando a classe de *e-mails* considerados como *spam* (1) ou não (0).

Entre as palavras utilizadas pelos atributos têm-se algumas como: *free, credit, business* (estes geralmente são encontrados em *e-mails* que são *spams*) e *Project, meeting, George, hp* (normalmente encontrados em *e-mails* pessoais que são legítimos). No Anexo I é encontrado a relação completa de todos os atributos desta base de dados. Esta base contém aproximadamente 8% de erro na classificação do *spam.*

3.1.2 Representação das entradas para o processo de treinamento

Foram mostradas várias técnicas existentes no processo de filtragem de *e-mail*, porém foram analisadas e testadas apenas: MultiLayer Perceptron de uma saída, Mapas Auto-Organizáveis (SOM) e Função de Base Radial (RBF).

Para esta representação das entradas dos algoritmos, os atributos da base de dados foram representados de acordo com o pseudocódigo do Algoritmo 1.

A base de dados "*spambase.data"* possui a representação de uma matriz de números inteiros: 4601 padrões x 58 atributos, onde os padrões são distribuídos em linhas e os atributos em colunas. Serão usados 3450 (75%) padrões para treinamento e 1151 (25%) para a validação.

Tabela 2: Algoritmo de representação da base de dados

Algoritmo 1:

1. Os padrões foram misturados aleatoriamente pelo WEKA®.
2. Separação dos atributos de entrada e saída – 57 atributos de entrada (freqüência de palavras e caracteres e 1 atributo de saída representando a classificação do e-mail: (1) *spam* e (0) não-*spam*.
3. Separação dos padrões para processamento dos algoritmos – 3450 padrões de treinamento e 1151 padrões de validação.

A Figura 3.1 representa o fluxograma da execução dos algoritmos utilizados através da ferramenta WEKA.

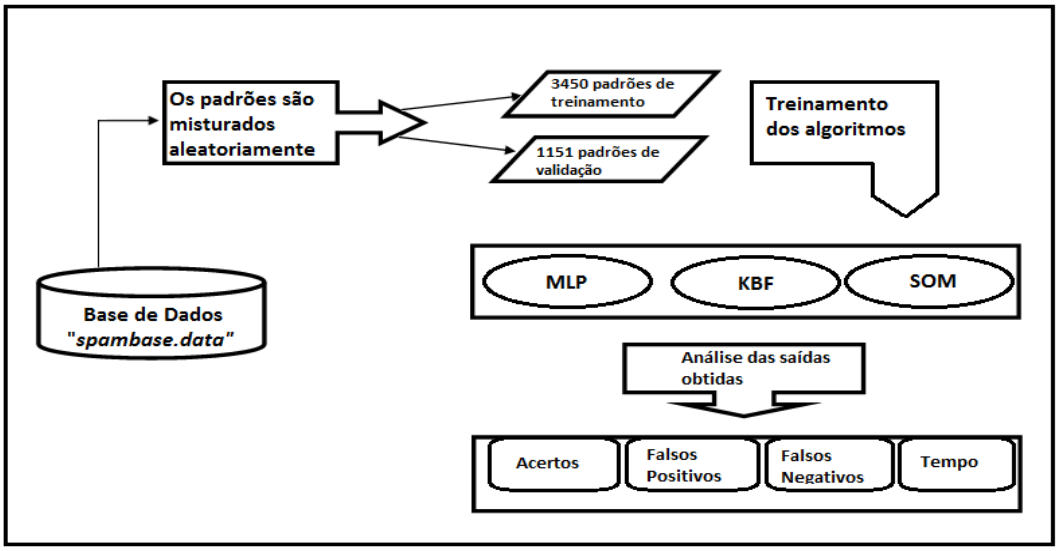

Figura 3.1: Fluxograma da execução dos algoritmos utilizados

3.1.3 A ferramenta WEKA®

Para realizar nossa análise, treinamento, validação e teste da rede, foi utilizado o módulo *Experimenter* da ferramenta WEKA®, versão 3.7.0 que é uma coleção de softwares com algoritmos de Aprendizagem de Máquina na linguagem de programação JAVA, muito utilizado para resolução de problemas de associação, classificação, regressão e clusterização. Foi desenvolvido pela universidade de Waikato e é uma ferramenta gratuita de código aberto, com uma interface bastante simples (Figura 3.2). O WEKA®, além de suportar os algoritmos proposto neste trabalho: MLP, SOM e RBF, é um *software* muito utilizado pela comunidade em aplicação e testes de RNAs.

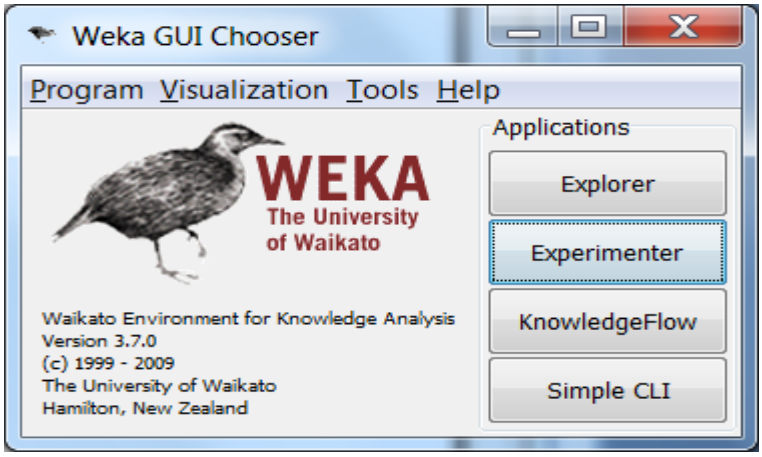

Figura 3.2: Interface gráfica do WEKA®

Conforme se observa, o WEKA® oferece quarto opções de uso: *Explorer, Experimenter, KnowledgeFlow e Simple CLI.* A opção *Explorer* permite o pré-processamento e aplicação de técnicas de aprendizagem de máquina, a segunda opção é mais usada para comparação entre técnicas variadas de inteligência computacional, a terceira opção é uma interface nova e reformulada para o WEKA® e finalmente a opção *Simple CLI* permite a realização de experimentos por meio de comandos (*prompt)*. Neste trabalho será usado apenas a primeira opção: *Explorer.*

Logo na aba principal (*preprocess*) (Figura 3.3) podemos escolher a base de dados que será usada assim como fazer um pré-processamento dos dados, como usar uma função de normalização por exemplo.

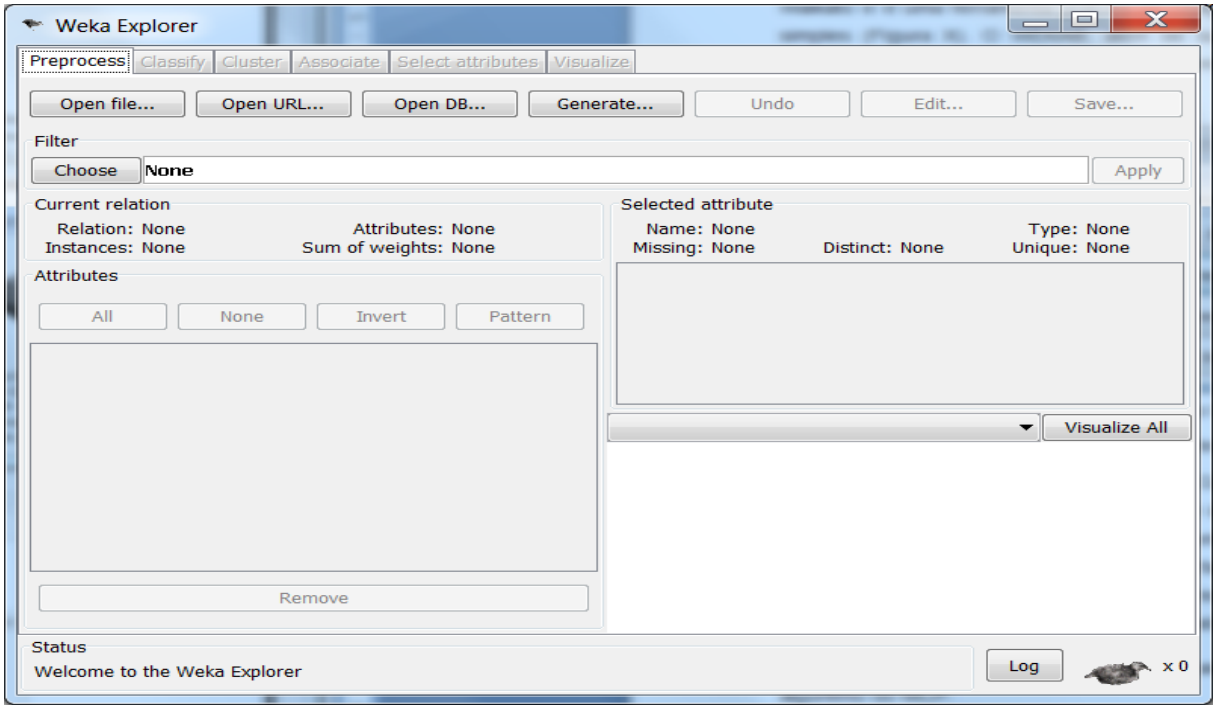

Figura 3.3: Interface gráfica da opção *Explorer* do WEKA®

A aba *Classify* é que será o nosso foco pois nela será escolhida os algoritmos usados neste trabalho (figura 3.4).

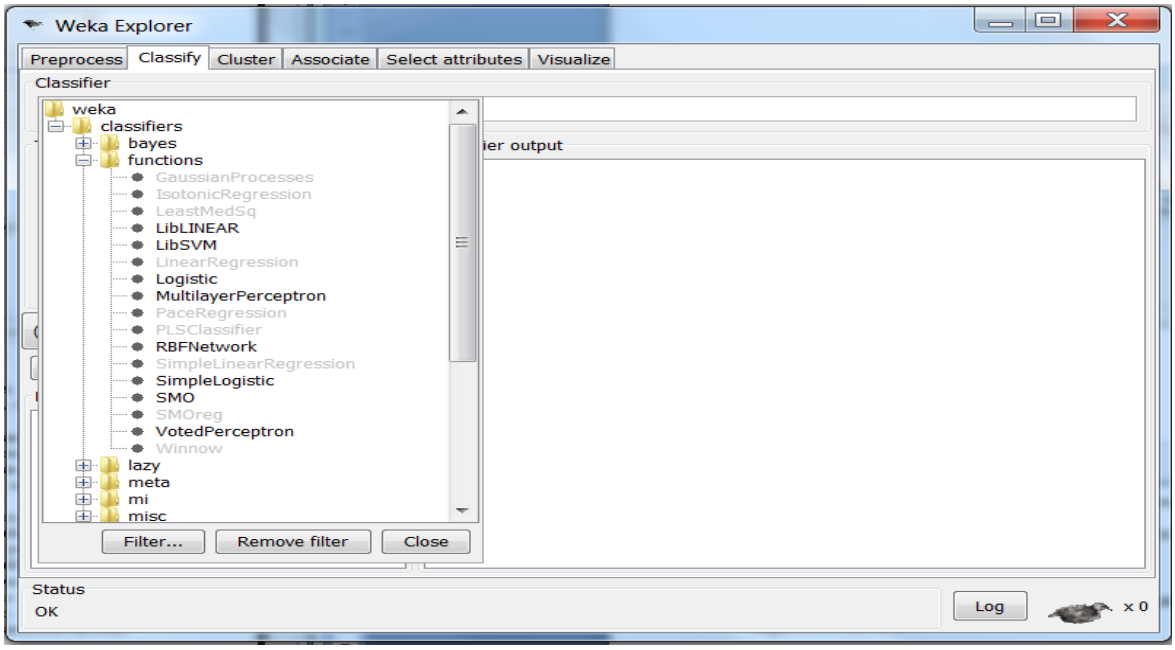

Figura 3.4: Interface gráfica do WEKA na aba *Classify*

Foram configurados os seguintes parâmetros de RNA no WEKA® para o algoritmo do MLP:

- Porcentagem de dados da base de treinamento, validação e teste;
- Taxa de aprendizado;
- Momento;
- Número de neurônios da camada escondida;
- *Threshold* de validação
- Número máximo de épocas de treinamento

Os valores usados serão apresentados posteriormente nos testes assim como os resultados obtidos

Para o algoritmo do RBF foram usados os seguintes parâmetros do WEKA®:

- Semente de Clusterização
- Número de Clusters

Finalmente para o algoritmo SOM, foram usados os seguintes parâmetros do WEKA®:

- Complexidade do parâmetro C
- Número de *folds* para a validação cruzada

3.2 Técnicas Utilizadas

Aqui será descrito como foi utilizado os algoritmos dentro do WEKA®.

3.2.1 MLP – Perceptron Multi Camadas

O algoritmo MLP foi implementado selecionando na aba *Classify* no WEKA® e escolhido o classificador “weka/classifiers/functions/MultilayerPerceptron”. O número de neurônios da camada escondida foi variado entre 10 a 100. O número de épocas também foi fixado em 500.

A taxa de aprendizagem e o momento, durante o treinamento, foram variados durante os testes.

Tabela 3: Algoritmo MLP

Algoritmo 2:
Definição dos parâmetros de treinamento. Taxa de aprendizado variado, Momento variado, Número de neurônios na camada escondida variado entre 10 e 100, 75% para treinamento, 25% para validação Os pesos e saídas anteriores são passados como parâmetros, sendo o aprendizado reconstituído a cada iteração. Tempo de processamento é somado a cada iteração. As saídas são arredondadas e apenas o valor absoluto é armazenado.

3.2.2 RBF – Função Base Radial

O algoritmo RBF foi implementado selecionando na aba *Classify* no WEKA® e escolhido o classificador “weka/classifiers/functions/RBFNetwork”. O número de cluster processado será definido através do uso de um classificador de *clusterização*, tal como o algoritmo de mapas auto-organizáveis de *Kohonen*. Após o término do algoritmo de *clusterização* usaremos o valor de cluster como um padrão de entrada para o parâmetro do algoritmo do KBF.

Tabela 4: Algoritmo KBF

Algoritmo 3:
Execução previamente de algum algoritmo de *clusterização*, tal como mapas auto-organizáveis de *Kohonen.* Usar o valor dos *clusters* como parâmetro para a definição dos dados de entrada para o algoritmo KBF. Selecionar uma Semente de Clusterização 75% para treinamento, 25% para validação Os pesos e saídas anteriores são passados como parâmetros, sendo o aprendizado reconstituído a cada iteração. Tempo de processamento é somado a cada iteração. As saídas são arredondadas e apenas o valor absoluto é armazenado.

3.2.3 SOM – Mapas Auto-Organizáveis

O algoritmo SOM foi implementado selecionando na aba *Classify* no WEKA® e escolhido o classificador “weka/classifiers/functions/SMO”. O número de *Folds* será usado o padrão do WEKA® para a validação cruzada. Será variado durantes os testes a variável c que é a complexidade do parâmetro c do algoritmo.

Tabela 5: Algoritmo SOM

Algoritmo 4:
Definição dos parâmetros de treinamento. Variar a variável c que representa a complexidade do parâmetro c do algoritmo. Escolher o número de *folds* para a validação cruzada. 75% para treinamento, 25% para validação Os pesos e saídas anteriores são passados como parâmetros, sendo o aprendizado reconstituído a cada iteração. Tempo de processamento é somado a cada iteração. As saídas são arredondadas e apenas o valor absoluto é armazenado.

Capítulo 4

Resultados Obtidos

Neste trabalho foram implementados três algoritmos de computação inteligente que são: Perceptron Multi Camadas – MLP, Mapas Auto-Organizável de Kohonen – SOM e as Funções de Base Radial – RBF, todos eles validados pelo programa WEKA®. Para todos os algoritmos foi utilizado a mesma base de dados obtida através do repositório UCI na universidade de Irvine e também foi utilizado a mesma quantidade de padrões de treinamento (3450 padrões – 75%) e de validação (1151 padrões – 25%). Foi utilizado a base de dados completa, do jeito que já veio pré-processada e o próprio WEKA® tratou de fazer as normalizações necessárias. Cada algoritmo foi testado várias vezes, com mudanças de parâmetros do algoritmo porém sempre permanecendo a mesma taxa de treinamento (75%) e de validação (25%).

Os resultados obtidos foram levados em consideração e analisados de acordo com os valores obtidos referentes a taxas de acertos, falsos positivos, falsos negativos e o tempo de processamento do algoritmo. A taxa de acerto se refere ao número de acertos obtidos na classificação dos padrões de validação, dividido pelo numero total de padrões de validação. Como já descrito no capitulo 1, falsos positivos se referem ao número de padrões de validação que representam *e-mails* legítimos do usuário, mas que foram classificados incorretamente como *spam*, dividido pelo número total de padrões de validação. Falsos negativos se referem ao número de padrões de validação que representam *spam* mas foram classificados incorretamente como *e-mail* legítimo, dividido pelo numero total dos padrões de validação. Tempo de processamento foi o tempo que cada algoritmo levou para treinamento e validação. Todos os valores, com exceção do tempo de processamento, são multiplicados por cem (valores em percentual).

4.1 Treinamento dos algoritmos no WEKA® utilizando a Base de Dados completa

Será descrito aqui os resultados obtidos após a utilizados dos algoritmos.

4.1.1 MultiLayer Perceptron – MLP

O algoritmo MLP foi implementado através de uma função já pronta no WEKA®. O programa foi executado várias vezes, onde foi coletado os melhores valores a cada execução. Em cada execução foi permanecido a taxa de treinamento (75%) e de validação (25%), a taxa de aprendizado foi fixada em 0.01 e o momento em 0.1, o número de neurônios na camada escondida foi variado entre 10 e 100 e o número de épocas foi fixado em 500. A Figura 4.1 e a Figura 4.2 mostra como se comportou a taxa de acerto e os falsos positivos e negativos de acordo com a variação do número de neurônios na camada escondida.

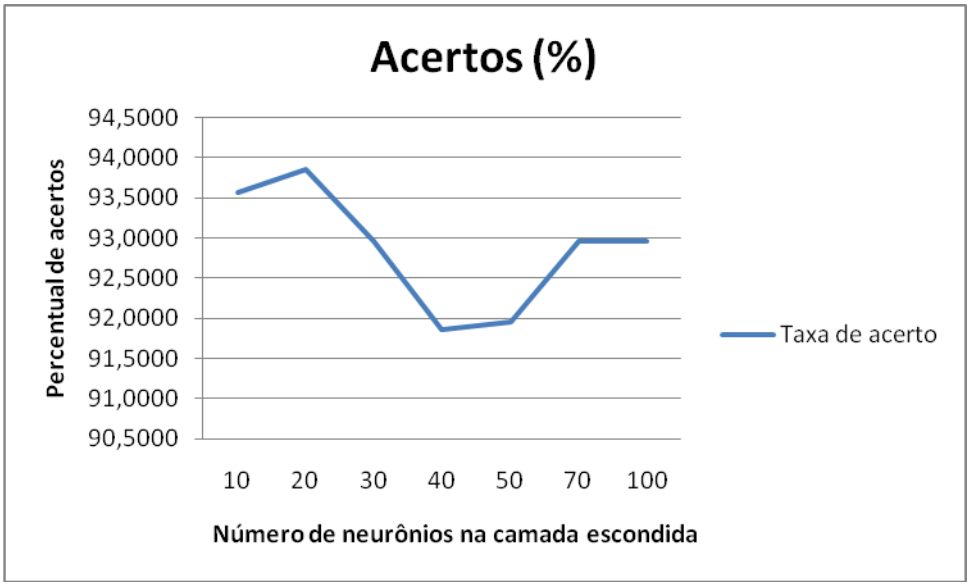

Figura 4.1: Taxa de acerto variada com o número de neurônios.

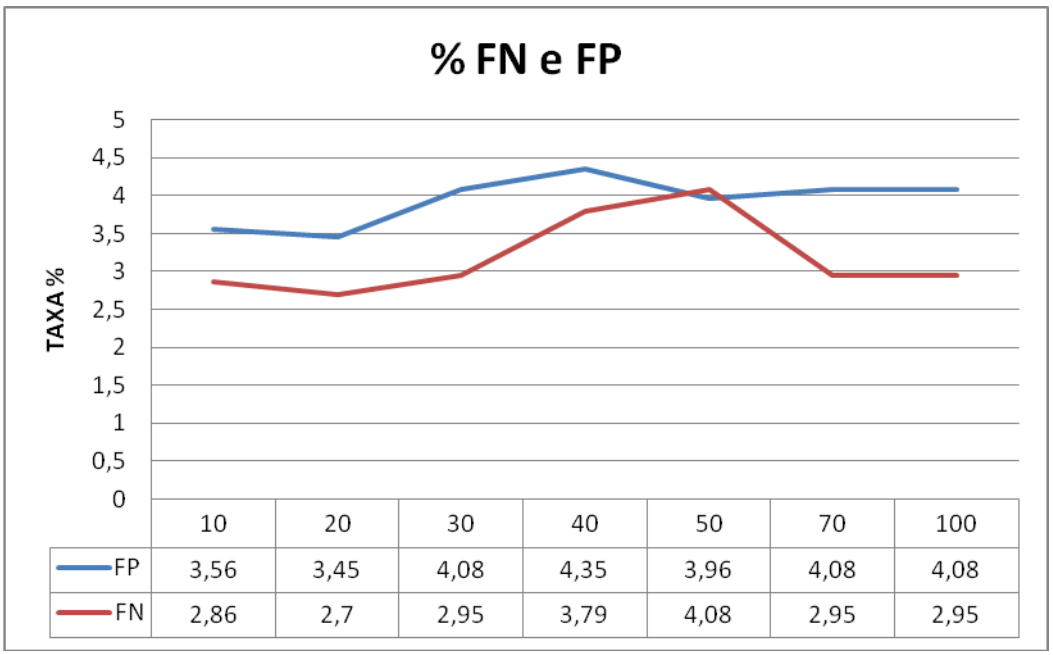

Figura 4.2: Percentual de FP e FN

Conforme pode ser visto pelos gráficos utilizando o número de neurônios igual a 20, apresenta a maior taxa de acerto, a menor taxa de falsos negativos e a menor taxa de falsos positivos. O tempo de processamento, embora maior que o obtido com 10 neurônios, ainda apresenta uma vantagem na escolha de 20 neurônios para o nosso teste apresentado pois levou aproximadamente 1 minuto e meio para o processamento completo enquanto com 100 neurônios levou quase 7 minutos para completar o processamento.

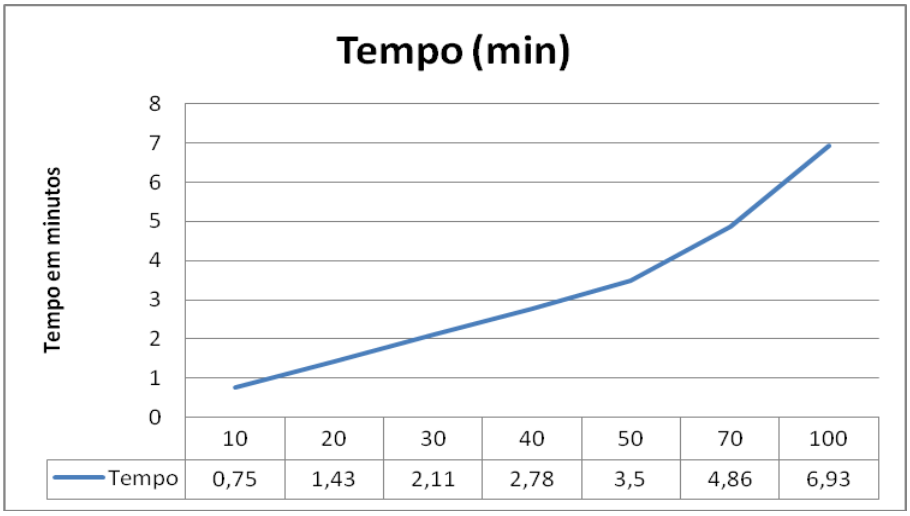

Figura 4.3: Tempo de processamento com o número de neurônios

A Tabela 6 resume melhor os resultados obtidos após o processamento no WEKA®.

Tabela 6: Resumo dos resultados usando MLP.

MLP – MultiLayer Perceptron				
Número de neurônios	**Acertos (%)**	**FP (%)**	**FN (%)**	**Tempo (min)**
10	93,5652	3,56	2,86	0,75
20	93,853	3,45	2,70	1,43
30	92,9565	4,08	2,95	2,11
40	91,8565	4,35	3,79	2,78
50	91,9595	3,96	4,08	3,50
70	92,9565	4,08	2,95	4,86
100	92,9565	4,08	2,95	6,93

Logo concluimos que um bom resultado é o obtido através do número de neurônios igual a 20 pois apresenta a melhor taxa de acertos, com a melhor taxa de falsos positivos e a melhor taxa de falsos negativos aliado com um razoável tempo de processamento quando comparado aos outros testes.

4.1.2 Mapas Auto-Organizável de Kohonen – SOM

O algoritmo SOM foi implementado também através de uma função já pronta no WEKA®. Também foi executado três testes onde foi permanecida a taxa de treinamento (75%) e a taxa de validação (25%). Nesse algoritmo foi variado apenas a complexidade do parâmetro C do algoritmo SOM. A Figura 4.4 e 4.5 mostra como se comportou a taxa de acertos, a taxa de falsos positivos e de falsos negativos.

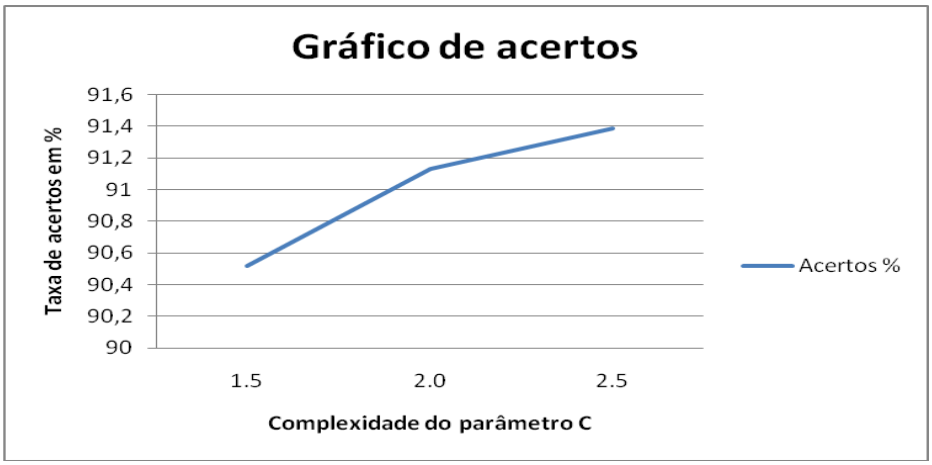

Figura 4.4: Gráfico do número de acertos relacionado com o parâmetro C.

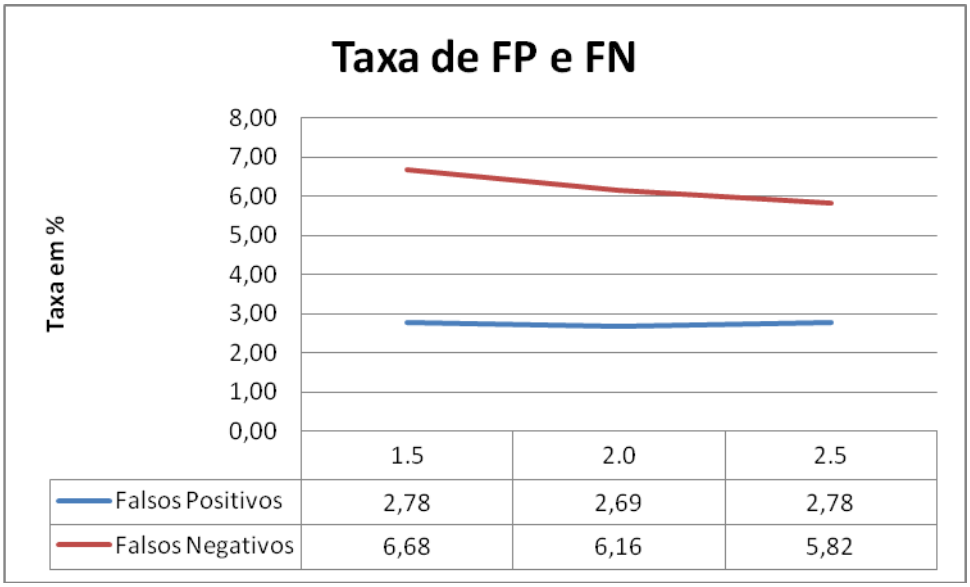

Figura 4.5: Gráfico da taxa de FP e FN com a variação do parâmetro C.

Conforme visto nos gráficos, o melhor resultado é apresentado ao rodar o algoritmo com a complexidade do parâmetro C fixado em 2.5 pois apresenta uma taxa de acerto de 91,39% junto com a menor taxa de falso negativo e uma taxa de falso positivo bastante baixa.

A Figura 4.6 mostra como se comportou o tempo de processamento desse algoritmo.

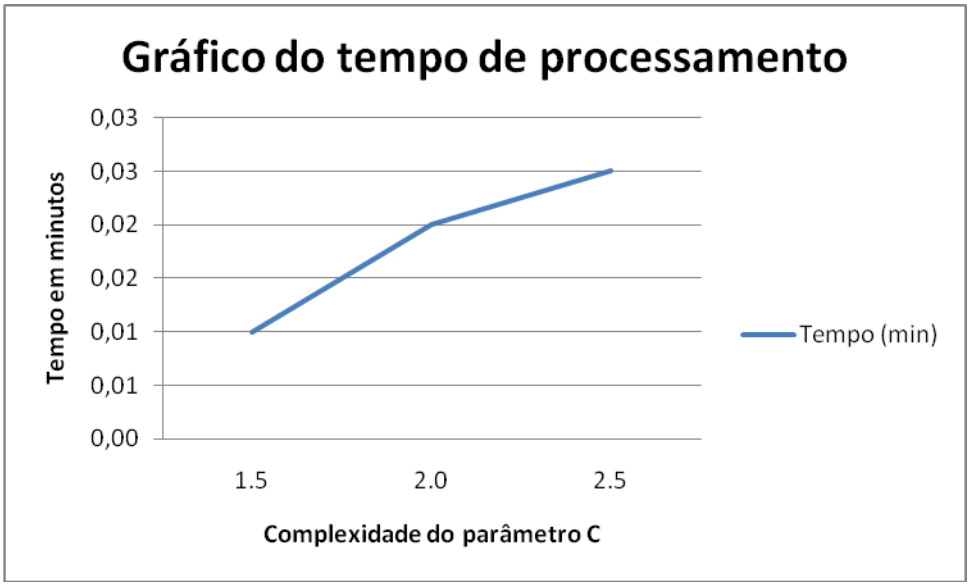

Figura 4.6: Gráfico do tempo de processamento usando SOM.

A Tabela 7 resume melhor os resultados obtidos.

Tabela 7: Resumo dos resultados obtidos usando SOM

SOM – Mapa Auto-Organizável de Kohonen				
Complexidade parâmetro C	**Acertos (%)**	**FP (%)**	**FN (%)**	**Tempo (min)**
1.5	90,52	2,78	6,68	0,01
2.0	91,13	2,69	6,16	0,02
2.5	91,39	2,78	5,82	0,025

O interessante do resultado encontrado é que este algoritmo mostrou-se levemente inferior ao algoritmo MLP pois apresentou taxas de acerto inferior, porém muito próximas das encontradas usando o algoritmo de MLP. O destaque fica no tempo de processamento extremamente rápido ao se comparar aos tempos obtidos usando a MLP, onde o modo mais rápido levou mais de 2 minutos e no algoritmo SOM todos levaram menos de 1 segundo.

4.1.3 Função de Base Radial – RBF

O algoritmo RBF, assim como os demais deste trabalho, também foi implementado usando uma função pronta do WEKA® e assim como já mencionado, o WEKA® trata de normalizar toda a base de dados. Assim como os demais, foi executado três vezes o programa, onde também foi permanecido a taxa de treinamento em 75% e a taxa de validação em 25%. Foi variado nesse algoritmo o número de cluster, que tem uma idéia similar ao numero de neurônios na camada escondida no MLP. O número de cluster é processado através de algum algoritmo de clusterização. O WEKA® usa o KMeans como algoritmo de clusterização. O algoritmo foi executado com número de cluster de 25, 30 e 40. A Figura 4.7 e 4.8 mostra como se comportou a taxa de acertos, a taxa de falsos positivos e a taxa de falsos negativos.

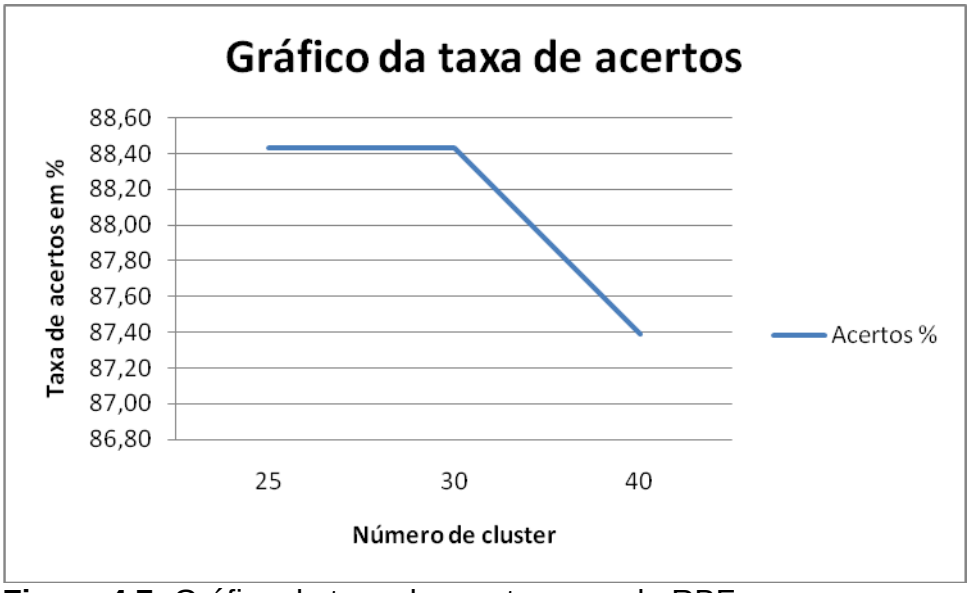

Figura 4.7: Gráfico da taxa de acertos usando RBF

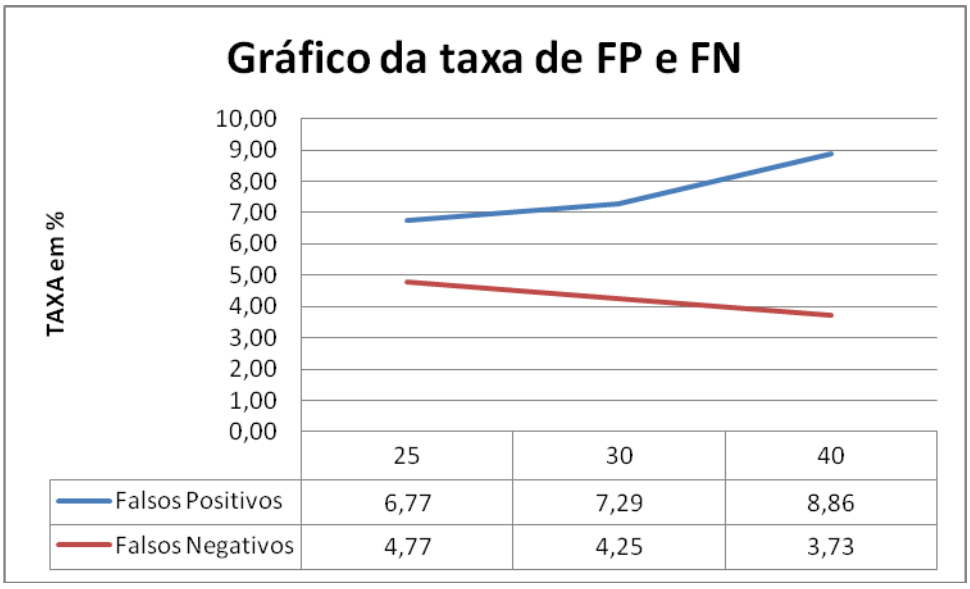

Figura 4.8: Gráfico da taxa de FP e FN usando RBF.

Pelo gráfico, observamos que usando o número de cluster igual a 30 obtemos uma taxa de acerto de 88.40%, junto com uma média de taxa de falsos positivos e falsos negativos. O Tempo de processamento do algoritmo pode ser visto na Figura 4.9.

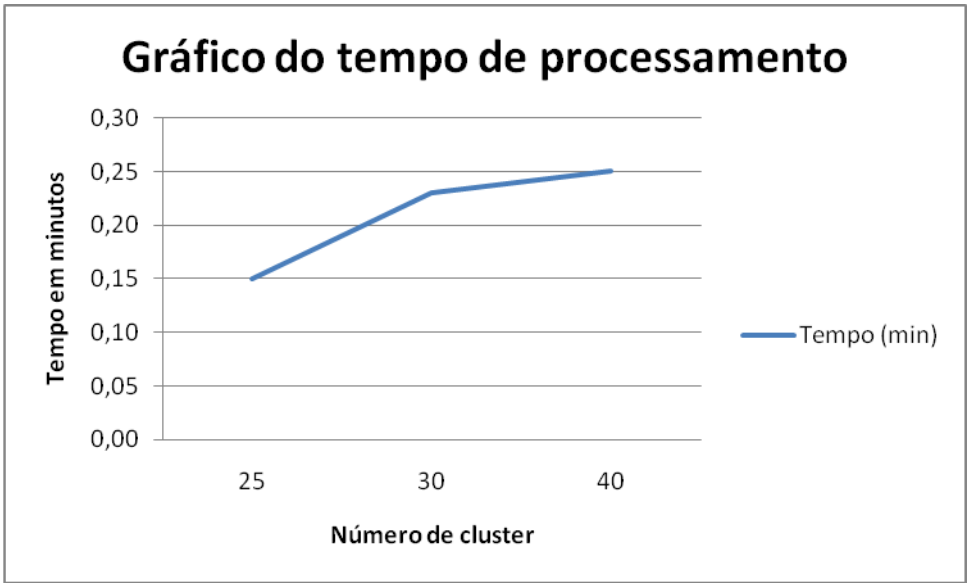

Figura 4.9: Gráfico do tempo de processamento usando RBF.

Conforme visto, o tempo de processamento é maior que o tempo de processamento levado pelo algoritmo de mapas auto-organizável de Kohonen, mas ainda assim é bastante menor que o tempo levado pelo algoritmo de MLP.

A Tabela 8 resume os resultados obtidos deste algoritmo.

Tabela 8: Resumo dos resultados obtidos usando RBF

RBF – Função de Base Radial				
Número de Cluster	**Acertos (%)**	**FP (%)**	**FN (%)**	**Tempo (min)**
25	88,43	6,77	4,77	0,15
30	88,43	7,29	4,25	0,23
40	87,39	8,86	3,73	0,25

Capítulo 5

Conclusão e Trabalhos Futuros

Neste trabalho foi proposto uma análise comparativa entre um conjunto de técnicas que se utilizam de técnicas de aprendizado de máquina para a detecção de *spam.* Conforme visto, os algoritmos escolhidos para a análise foram: MultiLayer Perceptron (MLP), Mapas Auto-Organizável (SOM) e Função de Base Radial (RBF). A abordagem adotada, como já mencionado, foi a de filtros de conteúdo por apresentar um melhor resultado prático. A base de dados utilizada foi a base disponível no Repositório de Aprendizagem de Máquina da Universidade da Califórnia, Irvine – UCI-R, com 4601 mensagens reais de *e-mail*, produzida no ano de 1999. Esta base conta com 58 atributos, onde um atributo representa a classificação do *e-mail* como *spam* ou não-*spam* enquanto os outros 57 atributos representam a freqüência de palavras ou freqüência de caracteres no corpo da mensagem.

Diversas dificuldades são encontradas na detecção de *spam* em filtros por conteúdo, porém a principal está relacionada com o próprio conteúdo das mensagens recebidas pelo usuário, pois estas podem ser facilmente confundidas com *spam* ou *e-mail* legítimo dependendo de como se refiram no corpo da mensagem. Por exemplo, caso algum destinatário receba vários *e-mails* relacionado com a palavra *credit,* podem ser confundidos com *spam* devido à presença dessa palavra e ao fato da base tratar uma freqüência grande da palavra *credit* como *spam* mesmo que em uma determinada situação, a mesma não seja *spam.* O filtro de conteúdo requer uma atualização constante de sua base de dados para apresentar uma correta classificação diariamente, visto que a cada dia, novas palavras são usadas com o intuito de enganar tais filtros.

A forma como a palavra é escrita também é fundamental para a detecção do *spam* pelo filtro de conteúdo. Por isso *spammers* têm o costume de criar artifícios nas palavras, de forma que engane o filtro, deixando ele passar como legítimo, mesmo sendo um *spam* porém sem prejudicar totalmente a leitura do usuário, de

forma que o usuário receberá a mensagem modificada porém continuará entendendo perfeitamente o conteúdo da mesma. Por exemplo a palavra *credit* poderia ser escrita e enviada no corpo do email como *cr3dit* ou ainda *cred1t*, isso jê é suficiente para o algoritmo falhar na detecção. Outras formas, como por exemplo mensagens não textual (txt ou HTML) também pode prejudicar a detecção do *spam.* E ainda tem sido bastante comum o uso de *spams* onde no corpo da mensagem são enviados em formato .gif, .jpg, .bmp, .pdf onde o conteúdo de *spam* é apresentado normalmente no cliente de *e-mail* como uma figura porém a análise irá falhar devido ao fato do algoritmo não ler mensagens no formato de imagens. Para isso seria preciso um refinamento no algoritmo, fazendo o *e-mail* recebido passar uma pré filtragem através do uso de reconhecimento de imagens para transformar a imagem em texto e assim o filtro analisar corretamente a mensagem e classificá-la como *spam* ou não-*spam*.

A Tabela 9 mostra uma comparação com os resultados que foram obtidos neste trabalho após o uso dos algoritmos analisados. São mostrados os resultados para a taxa de acertos, a taxa de falsos positivos, a taxa de falsos negativos e o tempo total de processamento de cada caso. Para o caso de filtros por conteúdo, é altamente interessante que se produza uma alta taxa de acertos, uma baixa taxa de falsos positivos, isto porque garante que mensagens legítimas do destinatário seja erroneamente classificado como *spam* e com isso podendo prejudicar substancialmente o usuário que não receberá a mensagem, aliado com um baixo tempo de processamento. O falso negativo também é desejado manter uma taxa baixa porém ela não prejudica tanto o usuário visto que um *spam* classificado como legítimo basta que o usuário vá pessoalmente até o *e-mail* e exclua-o. Foi mostrado o resultado do MLP com 20 neurônios na camada escondida, a complexidade do parâmetro C do mapa auto-organizável de Kohonen em 2.5 e o numero de cluster do RBF em 30.

Tabela 9: Comparação dos resultados obtidos.

Algoritmos	Acertos (%)	FN (%)	FP (%)	Tempo (min)
MLP	93,8530	3,45	2,70	1,43
SOM	91,13	2,69	6,16	0,02
KBF	88,43	7,29	4,25	0,23

Na tabela é mostrada os melhores valores obtidos em cada algoritmo. Apesar do tempo de processamento bastante elevado da MLP (mais de 1 minuto), ela se mostra o melhor algoritmo entre os testados por apresentar a maior taxa de acerto e a menor taxa de falso positivo. Apesar da taxa de falso negativo ser relativamente pouco maior que a obtida ao usar o algoritmo SOM, o falso positivo não é um problema tão grande como o falso negativo para o usuário.

Uma grande dificuldade encontrada neste trabalho foi a dificuldade de entender como os parâmetros de cada algoritmo se comporta nessa base de dados em especifico. Foi necessário vários testes variando cada parâmetro até conseguirmos chegar num resultado aceitável para o trabalho e de forma a obter um resultado também aceitável do ponto de vista real.

Um dos pontos que facilitou bastante este trabalho foi uso da ferramenta WEKA® pois o mesmo já continha os algoritmos usados implementados, bastando para isso apenas alterar os parâmetros necessários de cada algoritmo.

Para trabalhos futuros, propõe-se o uso de um pré-processamento para análise de mensagens com conteúdo de imagens, como as que contém mensagem com formato do tipo .gif, .bmp, .pdf. Também é sugerido utilizar filtros de cabeçalhos já que este trabalho apresentou uma base de dados com filtros por conteúdo.

Bibliografia

[1] SCULLEY, D.; CORMACK, G. Filtering e-mail spam in the presence of noisy user feedback. In: Proceedings of the Fifth Conference on Email and Anti-Spam. Mountain View, CA, USA: CEAS, 2008. Disponível em: <http://www.ceas.cc/2008/papers/ceas2008-paper-34.pdf>. Acesso em: 11/2009

[2] ANTISPAM.BR. AntiSpam.br - Comite Gestor da Internet no Brasil - CGI.br. 2008. Site: Núcleo de Informação e Coordenação do Ponto br - NIC.br. Disponível em: <http://www.antispam.br/>. Acesso em: 12/09/2009.

[3] TEIXEIRA, R. C. Combatendo o Spam: Aprenda como evitar e bloquear e-mails não-solicitados. 1a. ed. São Paulo, SP, Brasil: Novatec, 2004. pag: 171

[4] KASPERSKY, L. Introduction to spam. 2008. Site: Home page. Disponível em: <http://www.kaspersky.com/spam/>. Acesso em: 14/11/2009.

[5] WITTEL, G. L.; WU, S. F. On attacking statistical spam filters. In: Proceedings of the First Conference on Email and Anti-Spam. Mountain View, CA, USA: CEAS, 2004. Disponível em: <http://www.ceas.cc/papers-2004/170.pdf>. Acesso em:16/11/2009.

[6] FILHO, J. A. S. Utilização de Redes Neurais Artificiais em Classificação Autônoma de Peças Metálicas Empregando Imagens Radiográficas Aplicáveis a Sistemas IVA. Dissertação (Mestrado) — CEFET-MG - Programa de Pós-Graduação em Modelagem Matemática e Computacional, 2006.

[7] Haykin, S., "Neural Networks: A Comprehensive Foundation", 2 edn, Prentice-Hall, Inc (1999).

[8] Özgür, L., Güngör, T., Gürgen, F., “Adaptive anti-spam filtering for agglutinative languages: a special case for Turkish”, Pattern Recognition Letters 25 (2004) 1819-1831.

[9] Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Paliouras, G., Spyropolos, C.D., “An evaluation of naïve Bayesian anti-spam Filtering”, Proccedings of the Workshop on Machine Learning in the New Information Age (2000) 9-17.

[10] Zhang, L., Zhu, J., Yao, T., “An Evaluation of Statistical Spam Filtering Techniques”. ACM Transactions on Asian Language Information Processing 3 (2004) 243-269.

[11] Drucker, H., Wu, D., Vapnik, V.N., “Support vector machines for categorization”, IEEE Transactions on Neural networks 10 (1999) 1048-1054.

[12] Hidalgo, J.G., “Evaluating cost-sensitive unsolicited bilk e-mail categorization”, Proccedings of SAC-02, 17th ACM Symposium on Applied Computing, (2002) 615-620.

[13] Yang, Y., Pedersen, J.O., “A Comparative Study on Feature Selection in Text Categorization”, Proccedings of the International Conference on Machine Learning (1997).

[14] Carreras, X., Marquez, L. "Boosting Trees for Clause Splitting", Proceedings of CoNLL-2001, Toulouse, France (2001) 73-75

[15] SAKKIS, G. et al. A memory-based approach to anti-spam filtering for mailing lists. Information Retrieval, Kluwer Academic Publishers, Hingham, MA, USA, v. 6, n. 1, p. 49-73, January 2003. ISSN 1386-4564.

[16] Guzella, Thiago S.; Uchô, Joaquim Q.; Santos, Tomaz A. Mota; Caminhas, Walmir M. “Proposta de um modelo de classificação de padrões baseado no sistema imune: uma aplicação para a identificação de Spam”; 01/2005

[17] DE-CASTRO, L. N. Engenharia Imunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais. Tese (Doutorado) — UNICAMP - Faculdade de Engenharia Elétrica e de Computação, 2001.

[18] M. J. S. Valença, Fundamentos de Redes Neurais – Exemplos em JAVA, Livro Rápido, 2º edição (2009), 355-360.

[19] Braga, Antônio de Pádua; Carvalho, Andre Ponce de Leon F. Ludermir, Teresa Bernarda. Redes Neurais Artificiais: Teoria e aplicações. Rio de Janeiro: LTC, 2000

[20] E. Avci, An expert system based on Wavelet Neural Network-Adaptive Norm Entropy for scale invariant texture classification, Expert Systems with Applications 32, No. 3,(2007), 919-926.

[21] Blake, C; Mertz, C. Repository of Machine Learning Databases. Disponível em <http://www.ics.uci.edu/~mlearn/MLSummary.html> Acesso em Novembro, 2009

[22] B. Mulgrew, "Applying Radial Basis Functions", IEEE Signal Processing Magazine, pp 50-65, 1996.

[23] Sivanadyan, Thiagarajan. Detecting spam e-mails using neural networks. Artigo Técnico, University of Wisconsin-Madison, Disponível em <http://www.ece.wisc.edu/~hu/ece539/project/f03/index.html>, Acesso em 11/2009.

Anexo I

Atributos utilizados para representação da Base de Dados

Frequência de palavras:

1. word_freq_make
2. word_freq_address
3. word_freq_all
4. word_freq_3d
5. word_freq_our
6. word_freq_over
7. word_freq_remove
8. word_freq_internet
9. word_freq_order
10. word_freq_mail
11. word_freq_receive
12. word_freq_will
13. word_freq_people
14. word_freq_report
15. word_freq_addresses
16. word_freq_free
17. word_freq_business
18. word_freq_email
19. word_freq_you
20. word_freq_credit
21. word_freq_your
22. word_freq_font
23. word_freq_000
24. word_freq_money
25. word_freq_hp
26. word_freq_hpl
27. word_freq_george
28. word_freq_650
29. word_freq_lab
30. word_freq_labs
31. word_freq_telnet
32. word_freq_857
33. word_freq_data
34. word_freq_415
35. word_freq_85
36. word_freq_technology
37. word_freq_1999
38. word_freq_parts
39. word_freq_pm
40. word_freq_direct
41. word_freq_cs
42. word_freq_meeting
43. word_freq_original
44. word_freq_project
45. word_freq_re
46. word_freq_edu
47. word_freq_table
48. word_freq_conference

Frequência de caracteres:

49. char_freq_;
50. char_freq_!
51. char_freq_(
52. char_freq_$
53. char_freq_[
54. char_freq_#

Frequência letras maiúsculas:

55. capital_run_length_average
56. capital_run_length_longest
57. capital_run_length_total