Redes Neurais Artificiais para Detecção de *Web Spams*

Renato M. Silva[1], **Tiago A. Almeida**[2], **Akebo Yamakami**[1]

[1]Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (UNICAMP) – Campinas – SP – Brasil

[2]Departamento de Computação (DComp)
Universidade Federal de São Carlos (UFSCar) – Sorocaba – SP – Brasil

{renatoms,akebo}@dt.fee.unicamp.br, talmeida@ufscar.br

Abstract. *Web spams has become a major problem for Internet users, causing personal and economic losses. Fortunately, several methods have been proposed in the literature for automatic detection of this plague. However, the constant improvement of techniques used by spammers requires that the filtering approaches be more generic, efficient and with high capacity of adaptation. Given this scenario, this paper presents a performance evaluation of multi-layer perceptron artificial neural networks employed to solve such a problem.*

Resumo. *Web spam tem se tornado um problema frequente na vida dos usuários da Internet, provocando prejuízos pessoais e econômicos. Vários métodos vêm sendo propostos para detecção de web spam, porém a alta velocidade de aperfeiçoamento das técnicas usadas pelos spammers exige que os métodos de classificação sejam cada vez mais genéricos e eficientes. Diante disso, esse trabalho apresenta uma análise de desempenho de redes neurais perceptron de múltiplas camadas empregadas para combater esse problema.*

1. Introdução

Nos últimos anos, a *web* vem crescendo de maneira impressionante. Consequentemente, os motores de busca tornam-se cada vez mais importantes, pois ajudam os usuários a encontrar as informações desejadas, visando apresentar os resultados de forma organizada, rápida e eficiente. Porém, existem métodos mal intencionados que tentam burlar os métodos de busca manipulando o *ranking* de relevância das páginas. Isso deteriora os resultados da busca, frustra os usuários e os expõe a conteúdos inadequados e inseguros. Essa técnica enganosa é conhecida como "*web spamming*" [Svore et al. 2007].

Um *Web spam* pode conter conteúdo *spam* e/ou *spam links*. Segundo [Gyongyi and Garcia-Molina 2005], um exemplo de conteúdo *spam* é uma página de pornografia que contém milhares de palavras-chaves em sua página inicial sobre assuntos que não tem ligação com o conteúdo pornográfico e deixa essas palavras invisíveis por meio de recursos das linguagens de programação para *web*. Já, [Shen et al. 2006] explica que a técnica de *spam links* consiste em adicionar milhares de *links* para as páginas que pretende-se promover. Isso eleva o *ranking* das páginas em motores de busca que classificam sua relevância usando a relação da quantidade de *links* que apontam para ela.

Um dos menores problemas causados pela técnica de *web spamming* diz respeito ao incômodo causado por forjar respostas inesperadas, promovendo o anúncio de páginas indesejadas [Shen et al. 2006]. Porém, existem problemas mais graves que ameaçam os

usuários, uma vez que o *web spam* podem conter conteúdos maliciosos que instalam *malwares* nos computadores das vítimas [Egele et al. 2011].

Felizmente, algumas propostas para a detecção automática de *web spam* vêm sendo apresentadas na literatura. Entre elas, as mais utilizadas são as técnicas de aprendizado de máquina, tais como seleção de conjuntos (*ensemble selection*) [Erdélyi et al. 2011, Geng et al. 2007], *clustering* [Castillo et al. 2007, Largillier and Peyronnet 2010], floresta aleatória [Erdélyi et al. 2011], *boosting* [Erdélyi et al. 2011, Geng et al. 2007], máquinas de vetores de suporte [Svore et al. 2007] e árvores de decisão [Castillo et al. 2007, Gan and Suel 2007].

Porém, não foram encontrados trabalhos que usam as redes neurais artificiais (RNAs), uma das técnicas de reconhecimento de padrões mais conhecidas e bem sucedidas da literatura, para a classificação de *web spam*. Para preencher essa importante lacuna, o objetivo deste trabalho é apresentar uma análise do desempenho obtido por tais métodos, mais especificamente por meio da rede perceptron de múltiplas camadas (MLP - *multilayer perceptron*) usando o algoritmo de treinamento *backpropagation* com gradiente descendente e o algoritmo de segunda ordem Levenberg-Marquardt.

Este artigo está estruturado da seguinte forma: definições e características das RNAs são oferecidas na Seção 2. Na Seção 3 são expostos o protocolo experimental e os principais resultados obtidos. Por fim, conclusões são descritas na Seção 4.

2. Redes Neurais Artificiais

As RNAs são sistemas paralelos distribuídos e constituídos de unidades de processamento simples, chamadas neurônios, que tem capacidade computacional relacionada a aprendizagem e generalização. Um modelo básico de RNA possui os seguintes componentes: um conjunto de sinapses, um integrador, uma função de ativação e um bias [Haykin 2001].

Existem vários tipos de RNAs, tais como perceptron simples, Adaline, MLP, redes de Kohonen, redes recorrentes, entre outras. Entre elas, uma das que vem obtendo maior sucesso em problemas de reconhecimento de padrões é a rede MLP [Haykin 2001].

2.1. Redes neurais perceptron de múltiplas camadas

Uma MLP é uma rede perceptron que possui um conjunto de unidades sensoriais que formam a camada de entrada, uma ou mais camadas intermediárias de neurônios computacionais e uma camada de saída [Haykin 2001]. Por padrão, o seu treinamento é supervisionado e usa o algoritmo *backpropagation* (retropropagação do erro), que tem a função de encontrar as derivadas da função de erro com relação aos pesos e bias da rede. Esse algoritmo pode ser resumido em duas etapas: *forward* e *backward* [Bishop 1995].

Na etapa *forward*, o sinal é propagado pela rede, camada a camada, da seguinte forma: $u_j^l(n) = \sum_{i=0}^{m^{l-1}} w_{ji}^l(n) y_i^{l-1}(n)$, sendo $l = 0, 1, 2, ..., L$ o índice das camadas da rede. Quando $l = 0$, ele representa a camada de entrada e quando $l = L$ representa a camada de saída. Já, $y_i^{l-1}(n)$ é a função de saída do neurônio i na camada anterior $l-1$, $w_{ji}^l(n)$ é o peso sináptico do neurônio j na camada l e m^l é a quantidade de neurônios na camada l. Para $i = 0$, $y_0^{l-1}(n) = +1$ e $w_{j0}^l(n)$ representa o bias aplicado ao neurônio j da camada l. A saída do neurônio é dada por: $y_j^l(n) = \varphi_j(u_j^l(n))$, onde φ_j é a função de ativação do neurônio j. O erro deve pode calculado por: $e_j^l(n) = y_j^l(n) - d(n)$, sendo que $d(n)$ é a saída desejada para o padrão de entrada $x(n)$.

Na etapa *backward*, inicia-se a derivação do algoritmo *backpropagation*, a partir da camada de saída, onde tem-se: $\delta_j^L(n) = \varphi'_j(u_j^L(n))e_j^L(n)$, sendo φ'_j a derivada da função de ativação. Para $l = L, L-1, ..., 2$, calcula-se: $\delta_j^{l-1}(n) = \varphi'_j(u_j^{l-1}(n)) \sum_{i=1}^{m^l} w_{ji}^l(n) * \delta_j^l(n)$, para $j = 0, 1, ..., m^l - 1$.

Para maiores detalhes sobre as MLPs, consulte [Bishop 1995, Haykin 2001].

2.2. Algoritmo de Levenberg-Marquardt

O algoritmo de Levenberg-Marquardt é um método de otimização e aceleração da convergência do algoritmo *backpropagation*. Ele é considerado um método de segunda ordem, assim como os métodos do gradiente conjugado e do método quase-Newton, pois utiliza informações sobre a derivada segunda da função de erro [Bishop 1995].

Considerando que a função de erro usada na rede MLP é dada pelo erro quadrático médio (EQM) [Haykin 2001], a equação usada no método de Gauss-Newton para atualização dos pesos e consequente minimização do EQM é $W_{i+1} = W_1 - H^{-1}\nabla f(W)$, onde o gradiente $\nabla f(W)$ pode ser representado por $\nabla f(W) = J^T e$ e a matriz Hessiana H pode ser calculada por $\nabla^2 f(W) = J^T J + S$, sendo J uma matriz Jacobiana e $S = \sum_{i=1}^{n} e_i \nabla^2 e_i$. Supondo que S é um valor pequeno se comparado ao produto de J, a Hessiana pode ser representada por $\nabla^2 f(W) \approx J^T J$. Então, a atualização dos pesos no método de Gauss-Newton pode ser expressada por $W_{i+1} = W_1 - (J^T J)^{-1} J^T e$.

Como a matriz Hessiana simplificada não pode ser invertida, o algoritmo de Levenberg-Marquardt atualiza os pesos usando a equação $W_{i+1} = W_i - (J^T J + \mu I)^{-1} J^T e$, sendo que I é a matriz identidade e μ um parâmetro que torna a matriz Hessiana definida positiva.

Maiores detalhes sobre o algoritmo de Levemberg-Marquardt podem ser encontrados em [Bishop 1995, Hagan and Menhaj 1994].

3. Experimentos e Resultados

Os experimentos foram realizados com a coleção WEBSPAM-UK2006[1]. Trata-se de uma base de dados de *web spam* que possui 105.896.555 páginas hospedadas em 114.529 *hosts*. Foram utilizados 6509 vetores de características rotulados como *ham* e 1978 rotulados como *spam*. Esses vetores são fornecidos aos times participantes do *Web Spam Challenge* [2] (competição de técnicas de detecção de *web spam*). Cada vetor representa um host e é composto por 96 características propostas em [Castillo et al. 2007].

3.1. Configurações Gerais

Neste trabalho foram avaliadas as RNAs MLP treinada com o algoritmo *backpropagation* e método do gradiente descendente (MLP-GD) e MLP treinada com o método de Levemberg-Marquardt (MLP-LM).

Todas as redes MLP implementadas usam uma única camada intermediária e possuem um neurônio na camada de saída. Além disso, adotou-se uma função de ativação

[1] *Yahoo! Research: "Web Spam Collections"*. Disponível em `http://barcelona.research.yahoo.net/webspam/datasets/`.

[2] *Web Spam Challenge:* `http://webspam.lip6.fr/`

linear para o neurônio da camada de saída e uma função de ativação do tipo tangente hiperbólica para os neurônios da camada intermediária. Dessa forma, os dados usados para a classificação foram normalizados para o intervalo $[-1, 1]$.

Em relação aos parâmetros das MLPs, em todas as simulações, os critérios de parada adotados foram: número de épocas maior que um limiar θ, EQM do conjunto de treino menor que um limiar γ ou aumento do EQM do conjunto de validação (verificado a cada 10 épocas). Em todas as simulações com a MLP-GD, adotou-se os seguintes parâmetros: $\theta = 3000$, $\gamma = 0.001$, passo de aprendizagem $\alpha = 0.01$ e 100 neurônios na camada intermediária. Já na MLP-LM, os parâmetros adotados foram: $\theta = 3000$, $\gamma = 0.001$, passo de aprendizagem $\alpha = 0.01$ e 50 neurônios na camada intermediária. Tanto a arquitetura adotada quanto os parâmetros da rede foram definidos de forma empírica.

Cada classificador foi avaliado por meio do método de validação cruzada k-*folds*, usando k igual a 5 e foram calculadas as seguintes medidas de desempenho, amplamente empregadas na literatura: acurácia, sensitividade, especificidade, precisão e f-medida.

3.2. Resultados

Nas simulações com a MLP-GD o EQM obtido foi, em média, 0.13. Já, com a MLP-LM o EQM obtido foi, em média, 0.05. A Tabela 1 apresenta os resultados da classificação.

Tabela 1. Resultados obtidos pelas MLPs

	Acurácia	Sensitividade	Especificidade	Precisão	F-medida
MLP-GD:	0.853	0.532	0.952	0.775	0.631
MLP-LM:	0.903	0.692	0.969	0.874	0.772

Conforme pode ser observado na Tabela 1, nos resultados obtidos com a MLP-GD, apesar da alta taxa de acurácia (85.3%), a taxa de sensitividade foi inferior a 54%, enquanto que a especificidade foi de 95.2%. Tais valores indicam que a RNA implementada é muito boa na classificação de padrões pertencentes a classe ham, mas é ineficiente para dados da classe $spam$. Outra métrica importante, vastamente empregada na literatura, a f-medida, também obteve apenas um valor mediano.

Os resultados apresentados claramente indicam que a MLP-LM foi superior à MLP-GD. Note que, a taxa de acurácia atingiu 90.3% e a taxa de sensitividade alcançou quase 70%. Logo, conclui-se que esse classificador aumentou a capacidade de predição de dados da classe positiva. Além disso, a taxa de precisão subiu para 87.4% e, portanto, a chance de um padrão ser classificado como *spam* e realmente ser *spam* também aumentou. Em consequência, o valor da f-medida também teve um aumento significativo.

Após analisar como foram gerados os vetores de características por [Castillo et al. 2007], cogitou-se a hipótese de que poderiam haver redundâncias que estariam impactando nos resultados. Diante desse cenário, foi usado o método de Análise de Componentes Principais (PCA) [Qiu et al. 2003] que é amplamente empregado no processo de redução de dimensionalidade. Então, constatou-se que com apenas 23 dimensões é possível representar cerca de 99% das informações dos vetores de características.

Após reduzir os vetores de características de 96 para 23 dimensões, foram feitas novas simulações. Em média, o EQM obtido foi: 0.14 pela MLP-GD e 0.09 pela MLP-LM. Os resultados da classificação são apresentados na Tabela 2.

Apesar dos vetores de características obtidos pelo PCA representarem mais de 99% das informações dos dados originais, os resultados obtidos foram inferiores. Isso

Tabela 2. Resultados obtidos pelas MLPs após a redução de dimensionalidade

	Acurácia	Sensitividade	Especificidade	Precisão	F-medida
MLP-GD:	0.837	0.437	0.959	0.761	0.555
MLP-LM:	0.843	0.554	0.925	0.673	0.608

pode ser notado pela taxas de acurácia e f-medida menores do que as obtidas com os dados originais. Porém, na MLP-GD, a degradação dos resultados não foi tão expressiva e pôde ser compensada pelo ganho computacional, já que o número de dimensões que precisaram ser tratadas pela RNAs diminuiu mais que 50%.

A diferença constante entre as altas taxas de especificidade e as baixas taxas de sensitividade provavelmente poderia ser explicada pela grande diferença na quantidade de padrões presentes na classe *ham* em relação a classe *spam*. Desta forma, resolveu-se igualar a quantidade de dados nas duas classes para usar durante o treinamento das RNAs.

A partir dessa configuração de treinamento, foram realizadas novas simulações empregando os vetores de características originais. Em média, o EQM obtido foi: 0.12 para MLP-GD e 0.04 para MLP-LM. A Tabela 3 mostra os resultados da classificação.

Tabela 3. Resultados obtidos pelas MLPs usando classes de tamanhos iguais

	Acurácia	Sensitividade	Especificidade	Precisão	F-medida
MLP-GD:	0.811	0.793	0.828	0.814	0.803
MLP-LM:	0.899	0.902	0.897	0.892	0.897

Note que igualar a quantidade de representantes nas duas classes durante o treinamento melhorou os resultados da classificação. Isso pode ser observado pelo aumento do valor da f-medida, que foi bem superior ao encontrado nas demais simulações. Outro ponto a ser observado é que a taxa de sensitividade também melhorou, o que significa que os classificadores especializaram-se mais em acertar os padrões que são da classe *spam*.

Uma vez que em [Castillo et al. 2007] foi empregada a mesma base de dados e a mesma sistemática de validação utilizada neste trabalho, é possível comparar os resultados obtidos pelos autores com os obtidos pelas RNAs, conforme apresentado na Tabela 4.

Tabela 4. Comparação dos resultados das MLPs com resultados da literatura

Classificador	Características	F-medida
Árvores de decisão [Castillo et al. 2007]	conteúdo	0.683
Árvores de decisão [Castillo et al. 2007]	conteúdo e *links*	0.763
Melhores resultados obtidos pelas MLPs		
MLP-GD + classes com tamanhos iguais	conteúdo	0.803
MLP-LM + classes com tamanhos iguais	conteúdo	**0.897**

Conforme apresentado em [Castillo et al. 2007], usando o método de árvores de decisão, o maior valor da f-medida obtida na classificação de *web spam* por análise do conteúdo foi igual a 0.683. Em outro teste, os autores analisaram as características relacionadas tanto ao conteúdo das páginas quanto aos *links*, o que aumentou o valor da f-medida para 0.763. Porém, usando RNAs MLP, mesmo considerando apenas características relacionadas ao conteúdo, foi obtido um resultado superior nas simulações com a mesma quantidade de padrões para as duas classes durante o treinamento.

4. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma análise de desempenho da classificação de *web spam* usando redes neurais perceptron de múltiplas camadas. Em geral, observou-se que a rede com gradiente descendente é inferior à rede com o método de Levenberg-Marquardt. Contudo, os resultados obtidos por ambos os métodos, usando classes com mesmo número de representantes, foram superiores aos resultados consolidados apresentados em [Castillo et al. 2007].

Outro ponto importante a ser observado, é que treinar as redes neurais com classes de tamanhos muito diferentes faz os resultados convergirem em benefício da classe com maior número de representantes, o que aumenta a taxa de falsos positivos ou de falsos negativos. Logo, é importante balancear a quantidade de representantes de cada classe.

Em trabalhos futuros serão pesquisadas outras bases de dados e características para discriminação das classes *spam* e *ham*. Também será analisada a viabilidade do emprego de outras redes neurais, em especial a rede de Kohonen, já que alguns trabalhos na literatura indicam ter obtido êxito na classificação de *spam* enviados por email.

Referências

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press.

Castillo, C., Donato, D., & Gionis, A. (2007). Know your neighbors: web spam detection using the web topology. In *Proc. of the SIGIR*, 423–430, Amsterdam, The Netherlands.

Egele, M., Kolbitsch, C., & Platzer, C. (2011). Removing web spam links from search engine results. *Journal in Computer Virology*, 7:51–62.

Erdélyi, M., Garzó, A., & Benczúr, A. A. (2011). Web spam classification: a few features worth more. In *Proc. of the 2011 WebQuality*, 27–34, Hyderabad, India.

Gan, Q. & Suel, T. (2007). Improving web spam classifiers using link structure. In *Proc. of the 3rd AIRWeb*, 17–20, Banff, Alberta, Canada.

Geng, G., Wang, C., Li, Q., Xu, L., & Jin, X. (2007). Boosting the performance of web spam detection with ensemble under-sampling classification. In *Proc. of the 14th FSKD*, 583–587, Haikou, China.

Gyongyi, Z. & Garcia-Molina, H. (2005). Spam: it's not just for inboxes anymore. *Computer*, 38(10):28–34.

Hagan, M. T. & Menhaj, M. B. (1994). Training feedforward networks with the marquardt algorithm. *Neural Networks, IEEE Trans. on*, 5(6):989–993.

Haykin, S. (2001). *Redes Neurais - Príncipios e Práticas*. Bookman, São Paulo.

Largillier, T. & Peyronnet, S. (2010). Lightweight clustering methods for webspam demotion. In *Proc. of the IEEE/WIC/ACM IAT*, 98–104, Toronto, Canada.

Qiu, B., Prinet, V., Perrier, E., & Monga, G. (2003). Multi-block pca method for image change detection. In *Proc. of the 12th IAP*, 385–390.

Shen, G., Gao, B., Liu, T., Feng, G., Song, S., & Li, H. (2006). Detecting link spam using temporal information. In *Proc. of the 6th ICDM*, 1049–1053.

Svore, K. M., Wu, Q., & Burges, C. J. (2007). Improving web spam classification using rank-time features. In *Proc. of the 3rd AIRWeb*, 9–16, Banff, Alberta, Canada.