Algoritmo de Aprendizagem Para Redes Neurais Estocásticas:G-Networks

Frederico Samartini Queiroz Alves[1], **Antônio de Pádua Braga**[1]

[1]Departamento De Engenharia Elétrica - Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos 6627 - Belo Horizonte-MG CEP 31270-010 -Brazil

Fredsamartini@yahoo.com.br, apbraga@cpdee.ufmg.br

Abstract. *A new learning algorithm is proposed for the stochastic network which was developed by Erol Gelenbe. The G-networks, as these stochastic networks were called, are in fact a queueing system with positive and negative customers which model the inhibitory and excitatory pulses of a neural network. This kind of approach becomes attractive as it has a closer analogy with biological systems. We aim to obtain a learning process to solve classification tasks. The scheme developed here is a learning method which compare the output at all neurons before deciding the classification of it. Intervals' upper and down Limits for each neuron are defined and they are updated during the training faze to help in this choice.*

Resumo. *Um novo algoritmo de aprendizagem é proposto para a rede estocástica desenvolvida por Erol Gelenbe. Essa rede foi denominada de G-networks, e é na verdade um sistema de filas com chegadas positivas e negativas que modelam os pulsos inibitórios e exitatórios de uma rede neural. Essa analogia se torna atraente por ser mais próxima dos sistemas biológicos. Nosso objetivo aqui é obter um processo de aprendizagem que possa ser usado em problemas de classificação. Logo, desenvolvemos um método de aprendizagem que compara a saída em cada neurônio antes de decidir à qual classe esse pertence. Para ajudar nessa decisão, são definidos limites superiores e inferiores para cada neurônio. Durante a faze de treinamento esses limites são atualizados.*

1. Introdução

Para este trabalho proponho um novo algoritmo de aprendizado relativo a um modelo de rede estocástica que ajuda no modelamento de redes neurais. Essas redes estocásticas são modeladas através de uma rede de Filas em que existem chegadas positivas e chegadas negativas, que representam os sinais excitatórios e os inibitórios respectivamente. Nesse tipo de modelagem, é considerado que os pulsos que formam os sinais em redes neurais são analogamente os clientes que chegam a um sistema de filas. Esse tipo de rede de filas onde existem clientes negativos e positivos foi chamado de G-Networks [Gelenbe 1994],[Gelenbe 1989] e [Gelenbe 1991].

As G-networks foram aplicadas à problemas de classificação de padrões, entretanto, os seus parâmetros foram todos determinados através de métodos empíricos ou através de analogias com parâmetros que foram obtidos através de modelos específicos de redes neurais artificiais que já tinham sido consagradas para esses casos. Neste estudo

discutiremos um modelo de aprendizagem que consiste em determinar limites, inferiores e superiores, para cada neurônio da camada de saída. Esses limites são ajustados a cada iteração e depois são usados para definir, através de métodos de comparações, à qual classe cada entrada vai ser vinculada. Dessa maneira pretendemos aumentar o poder de atuação do modelo proposto quando aplicado a problemas de classificação.

2. Modelo G-networks

2.1. O modelo Neural e o modelo de Filas

G-networks é uma analogia feita entre a terminologia existente de Redes Neurais com Teoria de filas. Para uma rede neural contendo n neurônios aonde sinais positivos e negativos chegam e saem, foi criado um modelo similar de filas no qual clientes chegam a uma determinada unidade. Esses clientes vão sendo colocados em uma fila única, e podem ser retirados dessa a qualquer momento, o que foi representado através da chegada de clientes negativos. Cada sinal que chega a uma unidade pode vir de fora do sistema ou mesmo vir de uma outra unidade de dentro do sistema. Os sinais positivos que chegam a uma unidade vão sendo somados um a um, analogamente os clientes são acumulados na fila. O número de clientes nessa fila é equivalente ao potencial elétrico que existe em um neurônio. Quando um sinal negativo chega ao sistema, ou seja, tiramos uma pessoa da fila, estamos diminuindo o nível potencial desse neurônio. Portanto, o estado desta rede é representado por um vetor formado pelo potencial elétrico em cada neurônio, ou similarmente, pelo tamanho da fila em cada unidade [Gelenbe 1991].

Em um determinado momento, que acontece de acordo com uma distribuição exponencial, se o potencial de um neurônio for positivo, esse dispara um pulso que pode ir para fora do sistema ou para um outro neurônio, e então o seu potencial vai para zero.

2.2. Modelo matemático

Gelenbe em seu modelo propôs o seguinte modelo matemático: sempre que um neurônio i dispara, o que acontece de acordo com uma distribuição exponencial com taxa r(i), o sinal que deixa esse neurônio vai para um neurônio j com probabilidade $p^+(i,j)$ quando o sinal sai positivo e $p^-(i,j)$ quando o sinal é negativo. Consequentemente a probabilidade de um sinal sair de um neurônio i e ir para um neurônio j é p(i,j) = $p^+(i,j)$ + $p^-(i,j)$. O sinal pode também não ir para um outro neurônio, e sim deixar a rede com uma probabilidade d(i).Obviamente:

$$\sum_j p(i,j) + d(i) = 1 \tag{1}$$

A taxa de disparo r(i) é obtida através da seguinte forma:

$$r(i)p^+(i,j) = w_{ij}; \qquad se\ w_{ij} > 0 \tag{2}$$

$$r(i)p^-(i,j) = w_{ij}; \qquad se\ w_{ij} < 0 \tag{3}$$

Onde wij é o peso sináptico entre o neurônio i e o j. Então a taxa r(i) nada mais é que a soma de todos os pesos sinápticos dos neurônios que fazem conexão com o neurônio i.

$$r(i) = \sum_j |w_{ij}| \tag{4}$$

O neurônio só irá disparar de acordo com a taxa de disparo r(i) se no momento de disparo esse estiver com o potencial elétrico positivo, se não, nesse determinado momento ele não disparará. A probabilidade qi de que um neurônio tenha potencial positivo e, portanto, dispare, é descrita abaixo.

$$q_i = \frac{\lambda^+(i)}{\lambda^-(i) + r(i)} \tag{5}$$

As taxas λ^+ e λ^- são respectivamente as taxas de chegadas de sinais positivos e de sinais negativos em um neurônio i qualquer vinda de outros neurônios somadas às taxas de sinais também positivos ou negativos que vêm de fora do sistema. Estas estão descritas abaixo:

$$\lambda^+(i) = \sum_j q_j r(i) p^+(j,i) + \Lambda^+(i) \tag{6}$$

$$\lambda^-(i) = \sum_j q_j r(i) p^-(j,i) + \Lambda^-(i) \tag{7}$$

As taxas Λ^+ e Λ^- representam as chegadas de sinais positivos e negativos vindos de fora do sistema que chegam no neurônio i. A taxa Λ^- é comparada ao limiar de ativação do modelo conexionista e Λ^+ é a entrada do sistema.

2.3. Dificuldades do modelo

O modelo tinha algumas limitações que o tornava fraco em relação aos modelos já existentes usando redes neurais. Entre essas estão:

- O modelo depende dos pesos das sinapses Wij que podem ser, por exemplo, obtidos de forma ótima através de redes neurais. Isso faz com que o modelo fique dependente de algum outro que tenha capacidade, que esse não tem, de aprendizado.
- A taxa de disparo dos neurônios de saída assim como as taxas de chegada de sinais, positivos ou negativos, vindos de fora do sistema, ou seja, ”do meio externo” são definidas de modo empírico.
- O modelo não é capaz de por si só chegar a uma solução desejável ou ótima, em conseqüência mais uma vez, do fato de ter sido desenvolvido com algoritmos de aprendizagem e por que a quantidade de parâmetros livres é grande, o que dificulta qualquer tentativa de criação de um modelo geral que tenha traços de inteligência.

Neste trabalho fixamos alguns parâmetros que antes eram livres e construímos um algoritmo que muna o modelo com uma ferramenta de aprendizagem, para que possamos generaliza-lo para casos não específicos.

3. Inicialização do Modelo Proposto

Uma generalização para o uso de G-Network na resolução de problemas de classificação de padrões é muito útil, para isso, primeiramente definimos algumas regras para a inicialização dos parâmetros e apresentamos mais a frente nesta seção. Na próxima desenvolvemos o método de treinamento. Está ordem deve ser respeitada, pois o treinamento é dependente das inicializações que descreveremos aqui. Primeiramente devemos determinar o número de parâmetros na camada de entrada e depois o número na camada de saída. Para a camada de entrada o número de n neurônios deve ser o mesmo que o número de atributos de entrada. O número m de neurônios na camada de saída deve ser igual ao número de classes existentes no problema. Depois de determinada a quantidade de neurônios, que é igual ao número de neurônios na primeira camada mais os da camada de saída, ou seja, Quantidade de Neurônios = n +m, partimos para a definição das probabilidades de transição p(i,j). Consideramos que o sinal que parte de um neurônio i tem igual probabilidade de ir para qualquer um dos neurônios j, portanto, p(i,j)=1/m. Como o sinal que sai de um neurônio pode ser negativo ou positivo mas nunca as duas coisas de uma só vez, avaliamos que a probabilidade dele ser positivo ou negativo é $\frac{1}{2}$. Portanto:

$$p(i,j) = \frac{(-1)^{1\ ou\ 2}}{m} \tag{8}$$

Onde o expoente do numerador tem cinqüenta por cento de chance de ser 1 e cinqüenta por cento de ser 2. O leitor deve estar atento para o fato de que se o sinal da probabilidade acima for negativo, devemos ao calcular o q do neurônio que recebe esta taxa de cliente negativa, somada positivamente no denominador dessa. Quando o sinal que sai de um neurônio j é negativo, devemos considerar, quando este sinal vai para um neurônio i, como sendo parte da taxa λ^-.

Para cada atributo de entrada x, no caso representada pela taxa Λ^+ fizemos uma normalização para que não haja perda de detalhamento dos dados, pois a saída de cada neurônio i é dada pela probabilidade q(i) que é no máximo 1. Se a entrada for muito maior que a taxa de disparo mais a taxa de chegada de clientes negativos, como pode ser visto pela equação (5), a saída q(i) do neurônio i ficaria saturada. Logo, depois de passado pelo processo de normalização o atributo de entrada x será:

$$\Lambda^+(i) = \frac{x(i) - \min(\vec{x}(i))}{\max(\vec{x}(i)) - \min(\vec{x}(i))} \tag{9}$$

Consideramos a taxa Λ^- de chegada de clientes negativos vindos de fora do sistema igual a um para todos os neurônios. Essa escolha é o mesmo que igualar o limite de ativação do neurônio também a um. Em seguida inicializamos a taxa r(i) de disparo de cada neurônio aleatoriamente.

A probabilidade d(i) do neurônio disparar para fora do sistema é igual a zero para todos os neurônios de entrada, e igual a um para os de saída.

4. Algoritmo de Aprendizagem por Comparações

O algoritmo de aprendizagem detalhado aqui é uma tentativa de criar regras para a escolha da classe ao qual determinada entrada vai ser vinculada. O procedimento de aprendizagem

é feito ”off line” e consiste na criação de limites inferiores e superiores para os valores obtidos em cada neurônio da camada de saída.

Nós estamos interessados aqui em ajustar através de uma quantidade de dados de treinamento esses limites de aceitação, e depois relaciona-los a cada classe.

Suponha um sistema com c classes e, por conseguinte, c neurônios na camada de saída. Teremos nesse caso duas matrizes cxc. Uma é a matriz com os limites superiors e a outra é a matriz com os limites inferiores.

$$Ls = \begin{bmatrix} Ls_{1,1} & Ls_{1,2} & ... & Ls_{1,c} \\ Ls_{2,1} & Ls_{2,2} & ... & Ls_{2,c} \\ . & & & . \\ . & & & . \\ . & & & . \\ Ls_{c,1} & Ls_{c,2} & ... & Ls_{c,c} \end{bmatrix} \tag{10}$$

Tanto na matriz Ls acima, quanto na matriz Li apresentada abaixo, as c colunas representam as classes um a c, e as linhas de um a c representam os neurônios de saída.

$$Li = \begin{bmatrix} Li_{1,1} & Li_{1,2} & ... & Li_{1,c} \\ Li_{2,1} & Li_{2,2} & ... & Li_{2,c} \\ . & & & . \\ . & & & . \\ . & & & . \\ Li_{c,1} & Ls_{c,2} & ... & Li_{c,c} \end{bmatrix} \tag{11}$$

Para cada ponto de treinamento ajustamos os elementos das matrizes Ls e Li da seguinte forma:

1. Para cada ponto de entrada teremos q(i) saídas, para i=1,...,c. Como este é um esquema de aprendizagem supervisionado, temos que saber previamente à qual classe esse ponto de entrada pertence e então ajustaremos os valores da coluna referente à essa classe nas matrizes Ls e Li.

2. Considerando que o ponto de treinamento k seja pertencente à classe 1. Então temos que:

$$Ls_{j,1} = q(j,k) + eta \quad paraj = 1, ..., c \tag{12}$$

e

$$Li_{j,1} = q(j,k) - eta \quad paraj = 1, ..., c \tag{13}$$

Onde eta é um fator para determinação dos limites.

3. Para o ponto K+1 teremos novos valores para q e então ajustaremos os limites superiores e inferiores da seguinte forma:

$$Ls_{j,1} = \max[(q(j,k+1) + eta), Ls_{j,i}] \quad paraj = 1, ..., c \tag{14}$$

e

$$Li_{j,1} = \min[(q(j,k+1) - eta), Li_{j,i}] \quad paraj = 1, ..., c \tag{15}$$

Quando o ponto de entrada for relativo à outra classe que não seja a primeira devemos ajustar nas matrizes de limites acima os limites que estejam nas colunas referentes às respectivas classes.

4. Depois de definida as matrizes Ls e Li podemos iniciar o procedimento de classificação. Para sabermos a qual classe um determinado ponto pertence temos que avaliar todos os intervalos de classificação para cada classe para todos os q's de saída. Teremos no total c^2 intervalos que são definidos da seguinte forma. Para

$$I_{j,m} = Ls_{j,m} - Li_{j,m} \tag{16}$$

Temos a seguinte matriz de intervalos:

$$I = \begin{bmatrix} I_{1,1} & ...I_{1,m}... & I_{1,c} \\ I_{2,1} & ...I_{2,m}... & I_{2,c} \\ . & & . \\ . & ...I_{j,m}... & . \\ . & & . \\ I_{c,1} & ...I_{c,m}... & I_{c,c} \end{bmatrix} \tag{17}$$

Sendo

$$H_{j,m} = \begin{cases} 0 \; seq(j) \notin I_{j,m} \\ 1 \; seq(j) \in I_{j,m} \end{cases} \quad para \;\; j \; e \; m \; = \; 1, ...c \tag{18}$$

Teremos uma matriz H de uns e zeros formadas pelos resultados de $H_{j,m}$. Ou seja, para cada valor q(j) de saída teremos que avaliar se esse se encaixa nos limites de cada classe. Faremos isso para todos os q's. A classe que será atribuída ao conjunto de atributos vigentes, será a que tiver todos os elementos na coluna, da matriz H, referentes a ela, igual a um.

5. Quando mais de uma classe for atribuída a um conjunto de atributos de entrada, temos que definir qual é a classe mais provável. Essa será a que tiver a menor norma do somatório de seus intervalos.

$$N_m = ||H_m||_2 = \sqrt{\sum_j (H_{j,m})^2} \tag{19}$$

Onde m é a coluna de H referente as classe concorrente. A classe vencedora é a que tiver menor valor, logo:

$$Classe = min(N_m) \tag{20}$$

Exemplo 1- Um exemplo simples para ilustrar o funcionamento do modelo de G-Network e do método para treinamento é apresentado abaixo. A figura 1 abaixo mostra uma rede de filas que usaremos para solucionar o problema do X-or e do X-nor. Dois pares de entradas binárias x_1 e x_2 são dadas e estas produzem o par de saídas y_1 e y_2 que representam as probabilidades de disparo q_3 e q_4 respectivamente. A primeira representa

a função X-or e a segunda a função X-nor. Na Figura 1, mostramos quatro neurônios. Os neurônios um e dois são os de entrada e os neurônios três e quatro são os de saída. A taxa L(i) mostrada na figura é na verdade a taxa Λ^- de chegada de clientes negativos vindos de fora do sistema e que igualamos a um em todos os neurônios. De acordo com o processo de iniciação proposto acima e com as regras para o treinamento temos:

- $|p(1,3)| = |p(1,4)| = \frac{1}{2}$; O modulo foi colocado porque o sinal de cada uma é aleatório. Lembrando que, o sinal negativo da probabilidade está representando o tipo de pulso (excitatório ou inibitório);
- $|p(2,3)| = |p(2,4)| = \frac{1}{2}$; Pelos mesmos motivos acima;
- r(1) = r(2) = r(3) = r(4) = escolha aleatória entre 0-1;
- Colocamos o eta=0.01.

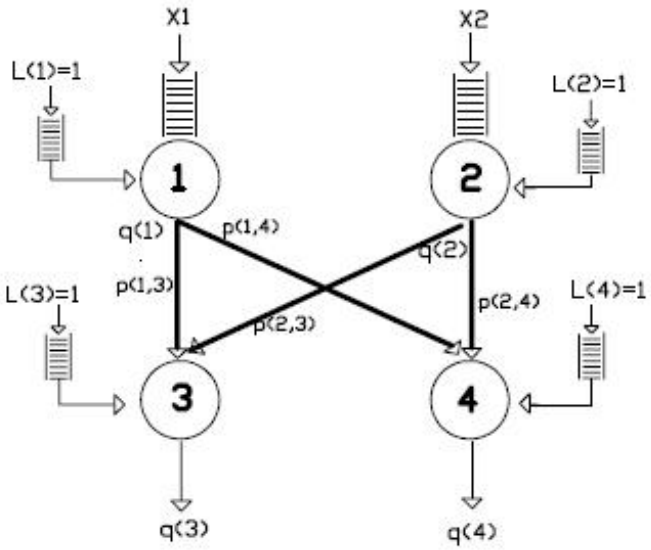

Figure 1. Modelo de G-network usado para representar a função X-or e X-nor

Abaixo está a tabela com os valores obtidos:

Table 1. Dados de Entrada

Tabela 1 - Dados de Entrada				
x_1	0	0	1	1
x_2	0	1	0	1

A matriz de limites superiores e inferiores obtida está abaixo:

$$Ls = \begin{bmatrix} 0.2112 & 0.1820 \\ 0.1990 & 0.3589 \end{bmatrix}$$

$$Li = \begin{bmatrix} -0.0100 & -0.0100 \\ 0.1499 & -0.0100 \end{bmatrix}$$

Table 2. Valores de saída dos neurônios 3 e 4

Tabela 2-Valores de saída nos neurônios 3 e 4				
q_3	0	0	0.2012	0.172
q_4	0	0.1599	0.189	0.3489

Table 3. Resultados da classificação

Tabela 3 - Resultado da classificação				
y_1	0	1	1	0
y_2	1	0	0	1

$$q_3 = \frac{q_1 r(1) p^+(1,3)}{q_2 r(2) p^-(2,3) + \lambda^-(3) + r(3)} \tag{21}$$

$$q_4 = \frac{q_1 r(1) p^+(1,4) + q_2 r(2) p^+(2,4)}{\lambda^-(4) + r(4)} \tag{22}$$

Observe que só a probabilidade p(2,3) teve o sinal negativo, e, portanto, a passamos para o denominador da função com valor positivo. Para esse modelo conseguimos representar a função X-or e X-nor sem nenhum erro.

Implementamos esse modelo também para casos onde haviam mais tipos de classes, onde a quantidade de dados para treinamento era pequeno e as classes eram sobrepostas. Nesse caso não obtivemos sucesso usando o modelo de aprendizagem proposto aqui. Na seção seguinte propomos a utilização de um outro modelo. Este poderia ser usado em conjunto com o atual para aumentar o poder desta ferramenta.

5. Algoritmo de Aprendizagem por Backpropagation

Uma alternativa ao algoritmo de aprendizado por comparações desenvolvido acima, é o desenvolvimento de uma rotina de treinamento que utiliza o método backpropagation para ajuste de parâmetros. No caso só o parâmetro r deve ser ajustado, já que os outros parâmetros já foram fixados, como mostrados na seção 3, acima.

6. Conclusão

Realizamos, neste trabalho, um estudo sobre as redes de filas chamadas G-networks desenvolvidas por Erol Gelenbe. Concluímos que, os modelos em que se utilizam essas redes com o objetivo de representar redes neurais, só podem ser usados para casos específicos, pois, os seus parâmetros são ajustados de forma empírica.

Buscando uma generalização dos modelos acima citados, criamos um método de inicialização dos parâmetros que são usados nessas redes de filas de forma que, alguns deles sejam fixados. Com isso, diminuímos a necessidade de treinamento para todos eles e também reduzimos a variância do sistema. Criamos, também, uma rotina de treinamento

para que pudéssemos resolver problemas de classificação de padrões sem ter que depender de atividades empíricas.

O modelo não foi suficiente apenas para resolver os problemas em que o número de classes era muito grande ou quando essas eram sobrepostas. Espera-se que, para a obtenção de melhores resultados, junto ao modelo de aprendizado por comparação, seja necessário ajustar os parâmetros r (taxa de disparo de cada neurônio), por exemplo, através de um método backpropagation.

References

Gelenbe, E. (1989). Random neural networks with negative and positive signals and product form solution. *Neural Comp*, 1:502–510.

Gelenbe, E. (1991). Queueing networks with negative and positive customers and product form solution. *J. App. Prob.*, 28:656–663.

Gelenbe, E. (1994). G-networks: a unifying model for neural and queueing networks. *Annals of Operations Research*, 48:433–461.

Jang, J.-S. R., Sun, C.-T., and Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing*. Prentice-Hall, Inc.

Kleinrock, L. (1976a). *Queueing Systems Vol. I.* Wiley and Sons, New York, NY.

Kleinrock, L. (1976b). *Queueing Systems Vol. II.* Wiley and Sons, New York, NY.

Rumelhart, D. E., McClelland, J. L., and the PDP Research Group (1986). Parallel distributed processing. *Bradford Books and MIT Press*, 1 and 2.