Prototipação de um Subagente Adaptativo Baseado em Redes Neurais

Elvis Melo Vieira
elvis@npd.ufsc.br

Solange Teresinha Sari
solange@npd.ufsc.br

Universidade Federal de Santa Catarina - UFSC
Núcleo de Processamento de Dados - NPD
Campus Universitário- Trindade
Florianópolis - SC

Resumo

As ferramentas de gerência de redes na sua maioria são autômatos que respondem a um evento ocorrido na rede com uma ação prevista algoritmicamente. Entretanto, este tipo de solução pode apresentar alguns problemas relacionados com o tamanho de memória e quantidade de tempo de cpu ocupados e principalmente, com a necessidade de uma base de conhecimento muito grande. Para sanar tais problemas é proposto neste trabalho a implementação de um Subagente Adaptativo baseado em Redes Neurais Artificiais, tendo em vista que este produto apresenta características de Aprendizagem e Generalização.

Abstract

The most of tools for network management are automatos which answer to an event ocorred in the network brought an action deducted by algoritm. But, this soluction can to presents problems to related to size of memory, cpu time and very knowlogment base need to applications. A Adapt Subagent is based to Artificial Neural Networks. Because presents Learnning and Generalization features, it don't have this problems.

1. Introdução

A Inteligência Artificial procura solucionar problemas facilmente resolvidos pelo cérebro humano, mas de solução algorítmica complexa. Tome-se como exemplo o jogo de xadrez, onde um lance pode ter uma enorme quantidade de possibilidades. O cálculo algorítmico de qual a melhor solução pode consumir muitos recursos da máquina como tempo de ocupação da CPU e quantidade de memória RAM utilizada.

Outro exemplo, são os Sistemas Especialistas, muito utilizados nos últimos anos. Estes sistemas usam o enfoque algorítmico para capturar a experiência de um especialista humano

usando dois componentes: (1) uma lista regras , (2) uma maneira de inferir e tirar conclusões sobre cada regra e alguns fatos fornecidos para responder uma questão. Por exemplo, tendo-se a seguinte regra: *se* X é filho de A e Y é filho de A, *então* X e Y são irmãos, e, os seguintes fatos: Maria é filha de Joana e Fred é filha de Jaques; então usando a regra definida, poderia-se concluir que Maria e Fred são irmãos.

Além disso, nem sempre um enfoque algoritmo se mostra adequado. Tome-se o caso de reconhecimento da escrita humana. Algoritmicamente, a solução para reconhecimento de um texto (sem entendimento semântico ou sintático), passaria pelo reconhecimento de cada letra. Mas isto obrigaria que fossem armazenadas em uma base de dados, todas as possíveis formas que uma determinada pessoa possa escrever cada caracter, por exemplo, a letra A. Obviamente isto é impossível, pois nada garante que no texto a ser reconhecido, tenha somente caracteres armazenados na base. A pessoa que escreveu pode ter desenhado um caracter com uma ligeira modificação de um forma existente na base de dados.

Algo semelhante acontece com as aplicações de gerência de redes de computadores. A maioria são autômatos que simplesmente respondem com uma ação, prevista algoritmicamente, a eventos ocorridos nos objetos gerenciados. Para eventos extraordinários e não previstos, o gerente da rede, o ser humano, deve tomar a decisão adequada.

A experiência humana está armazenada numa rede de células neurais. Os métodos de "cálculo" ou "dedução" são totalmente diferentes dos métodos algorítmicos. As Redes Neurais Artificiais (RNAs) inspiradas no cérebro humano, de um modo mais restrito, também podem apreender e portanto, podem se adaptar a mudanças do contexto a que foram treinadas.

A construção de aplicações de gerência de redes usando RNAs, pode desta forma, originar gerentes e agentes que podem se adaptar diante das constantes mudanças ocorridas numa rede.

2. Gerência de Redes

O crescimento do tamanho das redes, do seu número de usuários e dos serviços disponíveis, torna a gerência operacional de redes um elemento dinâmico e vital para o planejamento de outras áreas da informática.

Com o dinamismo existente numa rede de computadores, a equipe de administração e gerência deve estar preparada para solucionar os constantes problemas que muitas vezes são inevitáveis. Uma gerência operacional eficaz é imprescindível para que as falhas sejam solucionadas o mais rapidamente possível, preferencialmente, sem que o usuário sinta alguma anormalidade.

Estudos realizados por universidades, empresas e órgãos de padronização originaram os modelos de gerência, segundo os quais, são implementadas plataformas (comerciais ou não) que servem como ferramentas de gerenciamento de uma rede. Através destas plataformas podem-se gerenciar vários aspectos da rede como gerenciamento de falhas, performance, configuração, etc.

2.1 Modelos de Gerência

Um modelo de gerência visa organizar e montar uma estrutura, através da qual, a

1. Um ou mais **nós gerenciáveis**, cada um contendo um agente que responde a pedidos ou envia notificações ao(s) gerente(s);
2. No mínimo uma **estação de gerenciamento** de rede (ou NMS - Network Management Station), executando uma (ou mais) aplicação de gerenciamento, freqüentemente denominada de **gerente**(s), o qual(s) envia pedidos e recebe respostas ou notificações dos agentes.
3. A **Estrutura de Informação de Gerenciamento** (ou *SMI-Struture Management Information*) definindo as regras de como os **objetos gerenciáveis** são descritos e como ter acesso às operações do protocolo de suporte. O conjunto de definições das informações de gerenciamento sobre recursos é especificado na **Base de Informações de Gerenciamento (ou MIB** - Management Information Base).
4. As **operações de gerenciamento,** implementadas pelo **protocolo** suporte, especificando as primitivas disponíveis para o usuário manipular as informações de gerenciamento.

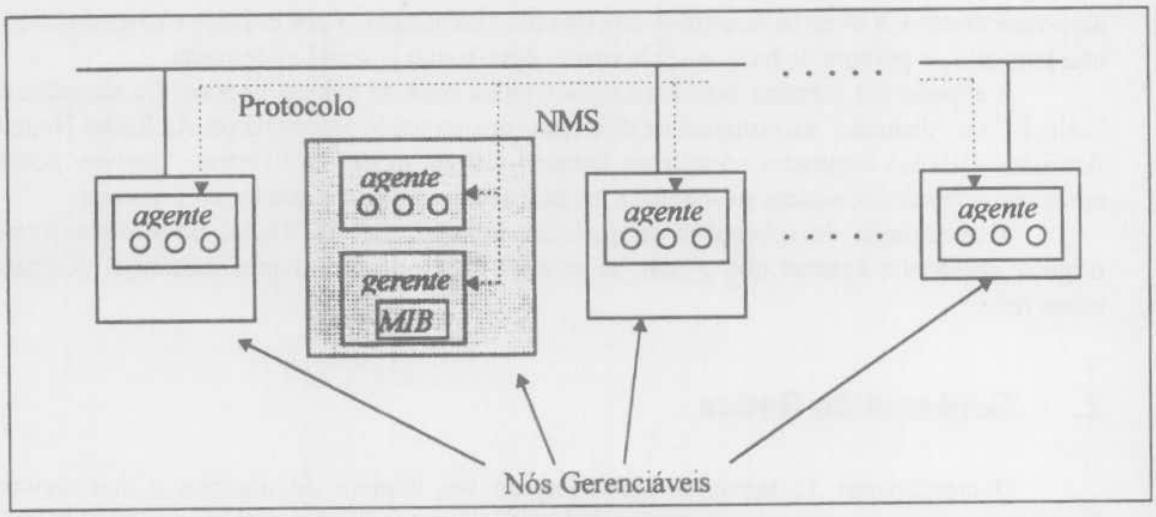

Figura 1 - Modelo de Gerência

2.2 Modelos de Gerência Padrões

Dois modelos de gerência são mais relevantes no contexto atual: o **Modelo OSI** da ISO e o **Modelo Internet** [09], que diferem basicamente na SMI empregada e nas operações do protocolo de gerenciamento. A estrutura de informações de gerenciamento do modelo Internet é mais simples, onde as próprias variáveis são consideradas como objetos, enquanto que no modelo OSI, são atributos de objetos. Portanto, é dito que o modelo *Internet* é baseado em objetos e o modelo *OSI* é orientado a objetos. Neste trabalho, enfoca-se o modelo de gerência *Internet*, por este ser o mais adotado para gerência de redes locais de computadores.

Os objetos no Modelo de Gerência *Internet* são especificados usando-se a *SMI* [10], que apresenta os objetos da MIB da Internet. O protocolo usado para manipular as informações de gerência no modelo *Internet* é o *Simple Network Management Protocol-*

Inicializadas pelo gerente: ***Get***, para obter o valor de uma variável; ***GetNext***, para obter, sequencialmente, os valores de uma variável; e ***Set***, para atribuir um novo valor a uma variável.

Inicializada pelo agente: ***Trap***, para informar o gerente da ocorrência de algum evento.

3. Redes Neurais Artificiais - RNAs

As redes neurais artificiais são inspiradas no cérebro humano, ou seja, ela é composta por elementos que realizam funções elementares análogas aos neurônios. Estes elementos são arranjados numa estrutura que procura imitar a estrutura da células no cérebro. Desta forma, as redes neurais artificiais apresentam algumas capacidades semelhantes ao humano, como por exemplo:

Aprendizagem: as RNAs podem modificar seu comportamento em resposta a eventos ou fatos que ocorrem no meio ambiente. Estes eventos ou fatos fornecem um conjunto de entradas (talvez acompanhadas de um conjunto de saídas desejadas), que provocam um auto-ajuste, através de um algoritmo de treinamento, para produzir um conjunto de respostas adequado e consistente com este novo padrão de entrada.

Generalização: depois de treinada, a resposta da rede pode ser insensível a pequenas variações na sua entrada. Esta característica é fornecida pela estrutura da rede e não devido a algum algoritmo intrínseco. A generalização é útil para tratar com pequenas variações de um padrão, ou seja, facilita o tratamento de problemas relacionados a um mundo imperfeito.

3.1 Neurônio Artificial

O neurônio artificial procura imitar o neurônio biológico. A figura 2, mostra um modelo que implementa a idéia do neurônio artificial.

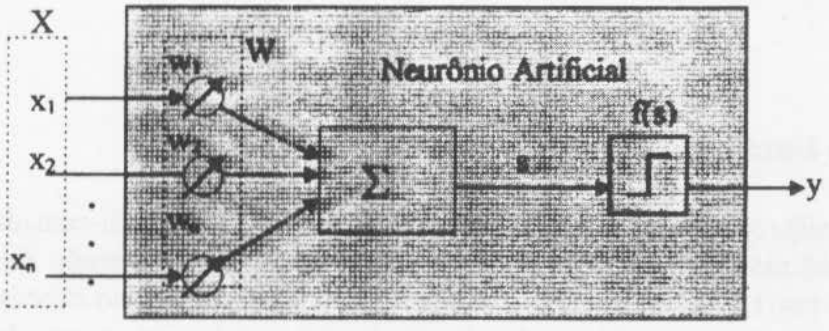

Figura 2 - Neurônio Artificial

Este neurônio recebe um conjunto de sinais da entrada X ($x_1,x_2,...,x_n$). A média ponderada s entre os elementos deste vetor e os pesos da conexões W ($w_1,w_2,...,w_n$), ou seja,

$$\underline{n}$$

é então aplicada à função de ativação f. O resultado y=f(s) é chamado de ativação do neurônio. Esta função pode ser uma Linear, Passo, Rampa ou Sigmoidal, sendo que esta última é mais indicada pelo fato de ser contínua em todo domínio.

3.2 Redes Neurais Feedforward Multi-Camadas

Estudos neurológicos deduzem que o cérebro humano é composto por camadas de neurônios, os quais processam a informação e disparam um sinal de resposta para os neurônios da camada seguinte.Existem também as células que retornam a informação para as camadas anteriores, mas este tipo de neurônio não faz parte desta estrutura.

A figura 3 mostra uma rede *feedforward* de múltiplas camadas, ou seja uma rede com duas camadas de processamento, onde cada camada processa a informação e realimenta a camada seguinte.

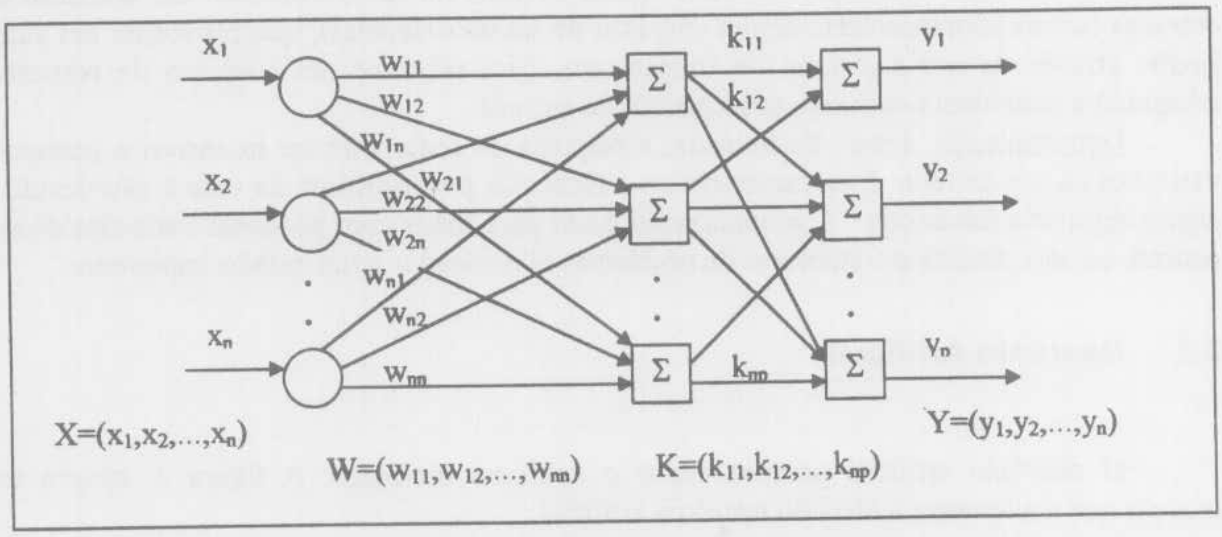

Figura 3 - Rede Neural de Múltiplas Camadas

3.3 O Modelo Backpropagation

A rede *Backpropagation* é uma rede *feedforward* [14] multi-camadas com um algoritmo de aprendizado denominado *Backpropagation*, o qual é baseado na regra delta generalizada. Esta regra procura minimizar o erro entre a saída obtida no processamento e a saída desejada. A cada iteração do algoritmo são reajustados os pesos das conexões proporcionalmente ao erro ocorrido, sendo que esta proporção é denominada Taxa de Aprendizado. Para aplicação deste algoritmo é necessário que a função de ativação do neurônio seja derivável em todos os seus pontos, para tanto é utilizado as funções sigmóide ($f(s)=\frac{1}{\ }$) ou tangente hiperbólica.

De modo geral, as rede neurais artificiais possuem três modos de execução :

1. Treinamento : a partir de um conjunto de treinamento (entrada e saída), o algoritmo é processado até que o erro seja menor que um valor pré-estabelecido, a tolerância. O resultado do treinamento é uma matriz de pesos das conexões.
2. Teste : um conjunto de teste (entrada e saída) diferente do conjunto de treinamento é constituído e processado, utilizando a matriz de pesos obtida anteriormente no treinamento. A saída obtida é comparada com a saída desejada, obtendo assim a taxa de acerto. Caso esta taxa seja insuficiente é necessario novo treinamento.
3. Execução : considera-se a rede treinada, então a partir de um conjunto (entrada) é possível obter a saída correspondente.

Algoritmo de Treinamento(Resumo): cada vetor de entrada tem um vetor de saída desejado correspondente e, ambos formam um par de treinamento. O conjunto de treinamento é composto por todos os pares de treinamento usados para a RNA *Backpropagation*. Um requisitos do algoritmo de treinamento backpropagation é que os valores iniciais atribuídos aos pesos sejam randômicos. Portanto, satisfeito este requisito inicial, os passos do algoritmo são os seguintes:

Repita
  Para cada par de vetores no conjunto de treinamento faça
    1. Aplique à entrada da RNA, o vetor de entrada constante próximo par de treinamento
    2. Calcule a saída obtida da RNA
    3. Calcule o erro obtido entre a saída obtida na RNA e a saída desejada constante no par de treinamento.
    4. Ajuste os pesos da rede de maneira que minimize o erro obtido
  Fim para
Até que o erro obtido esteja dentro da tolerância desejada

Nos passos 1 e 2, o sinal propaga-se na RNA da entrada para a saída e portanto, constituem um Passo Forward. Enquanto isso, os passos 3 e 4 constituem um Passo Backward. Estes passos podem ser melhor descritos em [06].

3.4 Características de Desempenho de Redes *Feedforward*

O método de treinamento de tais redes é muito lento devido ao fato que o conjunto dos padrões de treinamento são apresentados diversas vezes para que os pesos se estabeleçam de modo a classificar corretamente os padrões de entrada.

A rede apresenta alto desempenho quando classifica padrões similares aos do conjunto de treinamento, entretanto, não possui a habilidade de criar novas categorias de padrões [14].

4. Modelo de Gerência Adaptativa

Numa primeira análise, surgem três possibilidades de implementação de adaptação, no modelo de gerência tradicional:

i) Implementar um modelo onde somente os gerentes são adaptivos

Esta implementação é a mais natural, pois a política de gerência é implementada pelos gerentes. Qualquer decisão provocada por um evento extraordinário deve estar de acordo com a política adotada pela administração da rede.

Além disso, a política de gerenciamento pode ser mudada com mais facilidade, pois grande parte dela é implementada no gerente.

ii) Implementar um modelo onde somente os agentes são adaptivos

Esta implementação tem a vantagem de ser mais eficiente. As decisões são tomadas localmente, sem precisar enfrentar o tráfego da rede e esperar pelo atendimento do gerente. Em algumas situações críticas, pode ser uma vantagem extremamente desejada.

Entretanto, parte da política de gerenciamento fica distribuída pelos agentes. Assim, as mudanças são mais difíceis, pois pode haver a necessidade de retreinamento dos agentes.

iii) Implementar um modelo onde tanto o gerente como os agentes são adaptativos

Tomando-se cuidado, com a implementação de agentes e gerente adaptativos pode-se obter uma solução com ganho de eficiência em situações críticas, mas com partes da política de gerência, estratégicas e suscetíveis a constantes mudanças, implementadas pelo gerente. O agente se adapta em questões sobre sua autonomia.

Além disso, esta solução não permite que sejam reutilizados os agentes já escritos, pois estes teriam que ser reescritos incluindo o código que implementa a parte adaptativa.

Um quarto enfoque, onde preserva-se a arquitetura do modelo, mas adiciona-se um terceiro elemento, denominado de subgerente-agente. Este enfoque procura principalmente, preservar o agente existente.

4.1 Implementação de um Subagente Adaptativo

Neste enfoque, é criado um módulo adaptativo, denominado de Subagente que irá atuar como um gerente limitado. Na ocorrência de certos eventos, em que o subagente consiga tomar uma decisão (reativa ou não), o evento não é repassado ao agente, em vista que uma ação pode ser tomada pelo subgerente-agente (figura 4).

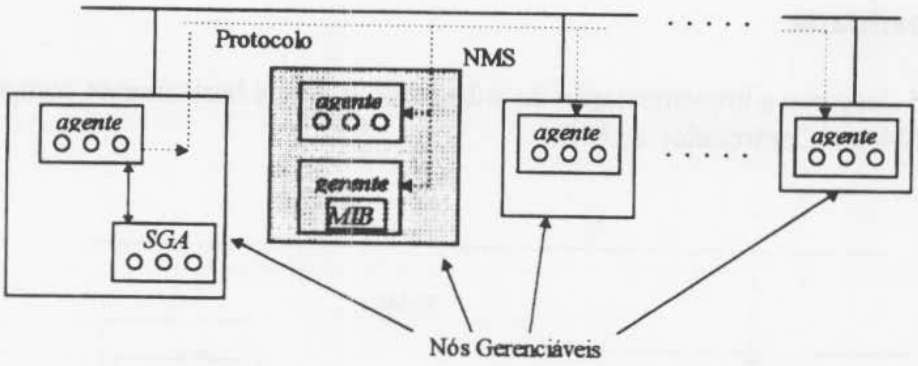

Figura 4 - Um Subgerente-Agente Adaptativo

Este módulo implementa a parte adaptativa necessária do agente. É Importante notar que o agente do fornecedor é conservado intacto. O subgerente-agente na ocorrência de certos eventos, como um subgerente tomando decisões e na ocorrência de outros, como um subagente repassando eventos para um agente. Devido a esta característica, o módulo é denominado de **Subgerente-Agente Adaptativo** (subagente).

4.2 Características do Subagente

O Subgerente-Agente Adaptativo introduz as seguintes características:

i) Filtragem de tráfego com o gerente

O tráfego entre o gerente e o nó gerenciado é otimizado, sendo que o subagente age como um filtro entre os dois. Por exemplo, se algum trap ocorrer e que o subagente pode tomar um decisão, não há a necessidade de passá-lo para o gerente através do agente.

ii) Tomada de decisões mais rápidas

Se algum evento ocorrer, como um alerta, requerendo uma ação rápida e que o subagente possa tomar, então haverá mais chances do problema ser solucionado. De outra forma, numa rede congestionada, por exemplo, o gerente poderia tomar a decisão tarde demais.

iii)Adaptação

O subagente está preparado para as mudanças de ambiente em que foi treinado. Por exemplo, pode-se treinar o subagente para tomar certas decisões que o administrador de rede tomava na console da aplicação gerente.

Com o subagente, o nó gerenciável passa a adquirir alguma autonomia com relação ao gerente, principalmente em questões não críticas.

4.3 Funcionalidade

A figura 5 descreve a implementação do subagente, o qual é basicamente composto por dois módulos: RNA e o Controlador de MOs.

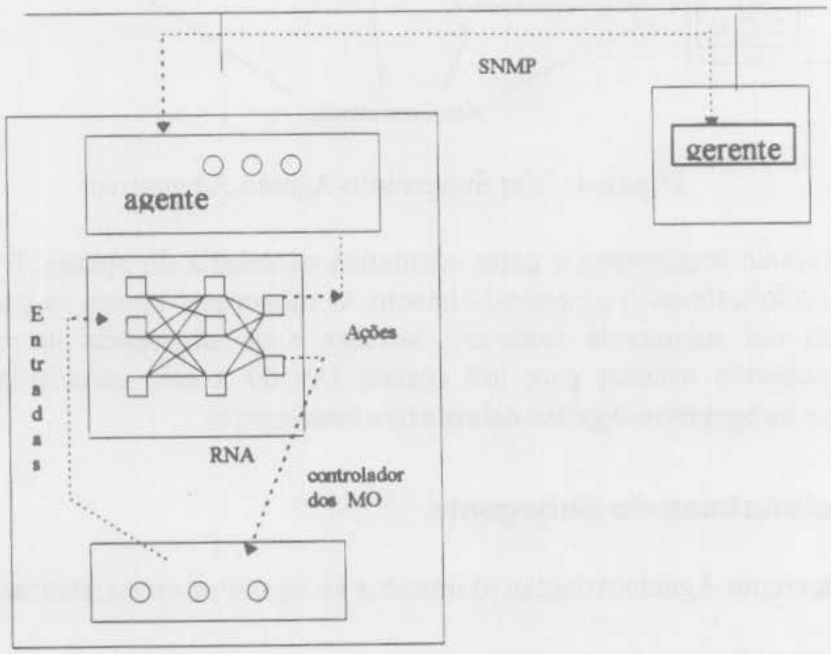

Figura 5 Implementação do subagente com RNAs

O Subgerente-Agente usa o protocolo SMUX [09] para se comunicar com o agente. Este protocolo permite que o subagente registre no agente, quais os objetos gerenciáveis (MOs) que ele controla. Depois do registro, o agente repassa todas as mensagens SNMP ao subagente, as quais são referentes aos MOs por ele controlados. Desta forma, ele deve atender as operações *set*, *get* e *get-next*, além de poder enviar *traps*.

O módulo controlador dos MOs implementa a interface entre o subagente e os recursos reais representados pelos MOs. Neste módulo são realizadas a coleta de informações nos MOs e montados os vetores de entrada para a RNA. Outra função realizada pelo módulo controlador é realizar as ações resultantes na saída do módulo RNA sobre os MOs

4.4 Treinamento do Subagente

O subagente, antes de estar pronto para operar, controlando os MOs sob sua tutela, ele deve ser treinado. Esta operação é *off-line*, ou seja, o subagente deve ser posto fora de ação, para ser treinado.

O treinamento deve ser realizado com um conjunto de treinamento composto por pares de vetores de treinamento. Cada par de treinamento deve conter um vetor de entrada e um vetor de saída desejada. O vetor de entrada representa os possíveis eventos que podem ocorrer e o vetor de saída desejada, as possíveis ações a serem tomadas para cada evento.

O conjunto de treinamento é armazenado em um arquivo, sendo que somente é consultado pelo subagente na fase de treinamento. Toda vez que for alterado este arquivo, novo treinamento deve ser efetuado.

4.5 Pré-processamento das Entradas e Saídas

Para o bom desempenho da rede neural é necessário um pré-processamento dos valores de entrada, diminuindo a complexidade dos padrões.

O pré-processamento das entradas para esta aplicação em particular está definida no ítem 6, onde é exemplificado um subagente para gerenciar *daemons* do Unix.

5. Implementação do Subagente

O subagente está sendo implementado numa plataforma de gerência IBM SystemView que segue o modelo de gerência Internet. A estação NMS é uma estação RS/6000 que executa o gerente SNMP NetView/6000 do SystemView. O subagente executa em outra estação de trabalho RS/6000, juntamente com o agente SNMP original.

A implementação, seguindo um projeto orientado a objetos, consiste basicamente em um processo Unix, construído através de um programa em C++. Várias classes de objetos foram desenvolvidas, sendo que podemos agrupá-las em 4 grupos principais: Classes de Gerência SNMP, Classes de Suporte e Classes de Redes.

Além destas classes específicas a implementação do subagente, foram necessárias a implementação de classes de objetos comuns a muitas aplicações como filas, pilhas, vetores, árvores, matrizes, etc.

5. 1 Classes de Gerência SNMP

São as classes utilizadas para implementar os objetos presentes no modelo de gerência SNMP. São apresentada a seguir:

- smuxPDU: modela a pdu do protocolo SMUX.
- smuxInt : interface para o SMUX.
- smuxPeer: é uma classe genérica (abstrata) que modela um processo que utiliza o protocolo SMUX. A partir dela, pode-se derivar novas classes para implementar programas que utilizem o protocolo SMUX (novos subagentes).
- MO : implementa um objeto gerenciado SNMP. Esta classe tem uma relação (containment) com as classes definidas antes: objectType e objectInst. Uma instância da classe MO contém dois atributos, um da classe objectType e o outro da classe objectInst, especificando, o tipo e a instância do MO.
- procMO: subclasse de MO, implementa um objeto gerenciável que representa um processo *daemon*.
- procCntrl: Implementa o controlador de MOs.
- MO : implementa um objeto gerenciado SNMP. Esta classe tem uma relação (containment) com as classes definidas antes: objectType e objectInst. Uma instância da classe MO contém dois atributos, um da classe objectType e o outro da classe objectInst, especificando, o tipo e a instância do MO.

5.2 Classes de Recursos de Redes Sistema Operacional

São as classes que implementam o suporte de rede TCP/IP necessário para o subagente. Estas classes são descritas abaixo:

- aplication - modela uma aplicação dentro da Interface Gráfica Motif . Uma instância de tal aplicação pode construir várias janelas gráficas dentro do Motif.
- graphic - é uma classe que implementa uma janela usada para plotar um gráfico. No subagente é usando para plotar o gráfico da evolução do treinamento, exibindo o erros em função das iterações.
- log - esta classe manipula todas as mensagens enviadas pelo subagente para o usuário.

5.3 Classes de Redes Neurais

Implementam os objetos necessários para a construção de uma rede neural *feedforward* multi-camadas com algoritmo de aprendizagem *backpropagation*.

- net - implementa um rede neural abstrata.
- superv - implementa uma rede neural com aprendizado supervisionado. É também uma classe abstrata.
- nsuperv - esta classe não é utilizada, mas foi colocada na hierarquia para genericidade, da mesma forma que a classe superv, nSuperv também uma classe abstrata.
- bp -implementa a uma rede neural que utiliza o algoritmo de aprendizagem backpropagation. Um ponto a ser notado, é que esta classe não é abstrata, como as demais, e portanto, instâncias de objetos podem ser construídos a partir dela.

6. Um Exemplo de Aplicação: subagente de daemons do UNIX

No Unix, *daemons* de rede são processos que implementam certos serviços de rede e ficam rodando enquanto o sistema está ativo. A performance e a detecção de falhas num nó de rede Unix está ligada diretamente com estes *daemons*. Por exemplo, em um determinado instante, um *daemon* pode estar roubando ciclos de CPU além do normal ou um daemon que implementa um serviço essencial de rede pode não estar rodando. Portanto a detecção destas situações e de outras semelhantes e a geração de uma ação reativa que minimize ou solucione o problema, pode ser realizada pela implementação de um subagente.

Os *daemons* do Unix mais comuns são os seguintes:

gated : processo de roteamento
named : servidor de nomes
inetd : inicializador de processos de redes

sendmail: roteador de mails
ypbind : processo cliente do NIS
ypserv : processo servidor do NIS
snmpd : agente snmp
slattach: processo que implementa uma interface slip
nfsd : processo que implementa o NFS
biod : processo que implementa o NFS
statd : processo que implementa o controle de estados dos arquivos do NFS
portmapper : inicializador de processos de redes que usam RPC.

Atributos comuns dos daemons

CPU time : fração do tempo total de CPU utilizada pelo processo.
mem size : fração do Tamanho de memória do sistema utilizada.
ws :tamanho em MB do working set utilizado pelo processo.
pri : Prioridade do Processo.
I/O : Fração da operações de I/O feitas pelo sistema.
state :Estado do Processo : Parado, esperando algum evento, executando, pronto para rodar, indeterminado,etc.

A manutenção do sistema operacional realizada por um gerente humano envolve, periodicamente, a observação dos atributos de cada processo e a tomada de uma ação visando ajustar o sistema, de forma que a performance permaneça dentro de certos índices aceitáveis. Por exemplo, se a observação do tempo de cpu gasto pelo processo inetd é muito alto em relação aos demais processos, uma possível ação seria abaixar a prioridade do processo em relação aos demais. Todo este procedimento de observação e tomada de decisões (limitadas) pode ser efetuado, em parte, por um Agente Adaptativo.

Existem varias possibilidades para se obter um conjunto de padrões de entrada que represente a observação efetuada pelo gerente humano e também, existem várias possibilidades de se forma um conjunto de padrões de saída que representam a sua tomada de decisões.

Portanto, uma análise criteriosa e o estabelecimento de uma metodologia deve ser realizado. Como o exemplo é simples para permitir o entendimento, quantidade de atributos tomados dos processos será restrida para a Prioridade (pri), Percentagem de CPU gasta (CPU), Percentagem de Memória Gasta (mem) e *Working Set* (ws) . Além disso, as ações serão restritas para que o agente somente possa alterar a prioridade do processo, enviar um trap SNMP de aviso ou um trap SNMP de erro ao gerente SNMP e para abortar o processo. O número de processos será também restrito para o named, gated, inetd e snmpd.

6.1 Formação dos Padrões de Entrada

A grande quantidade de valores possíveis numa rede neural, pode causar problemas e a rede pode não convergir no seu treinamento. Por exemplo, a Prioridade de um processo pode variar de 0 a 127, mas um processo com prioridade 60 e outro com 65, no Unix, pode-se dizer que ambos tem a mesma prioridade. Enquanto isso, um processo com prioridade 60 e outro com prioridade 10, tem prioridades bem diferentes.

Para restringir a quantidade de valores possíveis na entrada, os valores dos atributos dos processos são divididos em classes de valores. Para a prioridade, os valores podem variar de 0-127 e portanto serão divididos da seguinte forma:

Prioridade	Classe
0-24	4
25-49	3
50-74	2
75-99	1
100-127	0

Da mesma forma, os valores de CPU e Memória (percentagens) serão divididos em 4 Classes:

CPU, mem	Classe
0-19	0
20-39	1
40-59	2
60-79	3
80-100	4

O valor de *Working Set* pode variar dependendo do tamanho da memória da máquina, que no caso sera limitado em 32MB.

Working Set	Classe
0-2MB	0
2-4MB	1
4-8MB	2
8-16MB	3
16-32MB	4

Portanto, cada padrão de entrada será composto por um vetor de 16 elementos. Cada cada 4 elementos em seqüência deste vetor representa as classes dos valores dos atributos de um processo: Por exemplo, o processo named do vetor de entrada abaixo tem o valor de Prioridade 60.

pri	*cpu*	*mem*	*ws*	*pri*	*cpu*	*mem*	*ws*	*pri*	*cpu*	*mem*	*ws*	*pri*	*cpu*	*mem*	*ws*
2	0	0	1	2	0	0	1	2	0	0	1	2	0	0	1
named				*gated*				*inetd*				*snmpd*			

6.2 Formação dos Padrões de Saída

Uma grande quantidade de valores possíveis dos padrões de saída pode requerer que a rede neural possa separar os padrões de entrada em muitas classes possíveis. Isto também pode ocasionar que o treinamento da rede neural não tenha convergência.

Portanto, cada padrão de saída será composto por um vetor de 8 elementos. Cada seqüência de dois elementos irá representar as ações sobre um determinado processo.

Da mesma forma efetuada para os padrões de entrada, duas tabelas podem ser montadas , uma representando o incremento de prioridade dado a o processo:

Incremento de Prioridade	Classe
0	0
-16	1
-8	2
+8	3
+16	4

A outra tabela representa os traps e a decisão de abortar o processo:

Trap	Classe
Nenhum	0
Aviso ao gerente (necessita atenção)	1
Alerta ao gerente (Erro Grave)	2
Alerta ao gerente (Processo não operacional)	3
Abortar o processo + Alerta ao gerente	4

Por exemplo, o padrão de saída seguinte:

npri	*trap*	*npri*	*trap*	*npri*	*trap*	*npri*	*trap*
0	0	2	0	0	0	0	0
named		*gated*		*inetd*		*snmpd*	

indica uma ação que faz com que a Prioridade do *gated* seja diminuida de 8 e um aviso é enviado ao gerente. Observe que nos outros processos não é efetuada nenhuma ação.

6.3 Formação dos Arquivos de Padrões de Entrada e Saída

O conjunto de treinamento da rede são lidos de um arquivo. Cada par de treinamento é composto por um vetor de entrada e de saída (ação) correspondente, formados como descrito acima. O pares de treinamento do arquivo formam um pequena amostra dos possíveis pares de treinamento.

Um possível arquivo de treinamento seria o seguinte:

```
2 1 0 1  2 0 0 0  2 0 0 0  2 0 0 0    0 0 0 0  0 0 0 0
4 1 0 1  2 0 0 0  2 0 0 0  2 0 0 0    4 1 0 0  0 0 0 0
3 1 0 1  2 0 0 0  2 0 0 0  2 0 0 0    3 0 0 0  0 0 0 0
1 1 0 1  2 0 0 0  2 0 0 0  2 0 0 0    2 0 0 0  0 0 0 0
2 1 0 1  1 0 0 0  2 0 0 0  2 0 0 0    0 0 2 0  0 0 0 0
0 1 0 1  2 0 0 0  2 0 0 0  2 0 0 0    1 0 0 0  0 0 0 0
   .        .        .        .        .        .
   .        .        .        .        .        .
   .        .        .        .        .        .

2 1 0 1  2 0 0 3  2 0 0 0  2 0 0 0    0 0 1 4  0 0 0 0
```

Uma vez treinada, a rede pode ser testada. Para o teste, um arquivo semelhante ao arquivo de treinamento é montado. A rede então, lê o vetor de entrada de cada par do arquivo de teste e executa encontrando um vetor de saída. Este vetor de saída é comparado com o vetor de saída desejado (ação desejada) para verificar se o valor obtido está correto ou não. Desta forma, pode-se decidir se há a necessidade de um novo treinamento com outros parâmetros para a rede.

Tal arquivo de teste pode ter, por exemplo, os seguintes pares:

```
2 1 0 1  2 0 0 0  2 0 0 0  0 0 0 0    0 0 0 0  0 0 0 3
3 1 0 1  2 0 0 0  2 0 0 0  2 0 0 0    3 0 0 0  0 0 0 0
   .        .        .        .        .        .
   .        .        .        .        .        .
   .        .        .        .        .        .

2 1 0 1  2 0 0 0  2 4 0 0  2 0 0 0    0 0 0 0  1 2 0 0
2 1 0 1  2 0 0 0  2 0 0 0  2 0 3 0    0 0 0 0  0 0 1 2
```

6.4 Resultados Obtidos

Os resultados preliminares foram efetuados em um 486 DX2 66Hz, utilizando um simulador de redes neurais.

Para um treinamento de um conjunto com 36 padrões, com taxa de aprendizado igual a 0,5 e erro médio igual a 0,168, durante 61069 iterações; obtivemos uma taxa de acerto igual a 84 % em um conjunto de teste com 12 padrões.

Este resultado é encorajador para ampliações nos conjuntos de treinamento e teste, bem como uma reavaliação do pré-processamento das entradas da rede.

7. Conclusão

Além das características de um subagente já citadas, a sua implementação apresentou as seguintes características adicionais:

- Ocupação de pouca memória da máquina. Diferentemente de uma implementação algorítmica a ocupação de memória da máquina é pouca. Usando-se um sistema especialista, por exemplo, teria-se uma grande memória utilizada para armazenar a base de conhecimento necessária.
- Rapidez na geração das saídas, em resposta ao conjunto de entradas. Novamente, usando-se a um sistema especialista, as saídas podem demorar a serem geradas, pois um conjunto muito grande de regras podem ser necessárias para a pesquisa da solução.
- Um excelente grau de genericidade, ou seja, a implementação do subagente pode servir para um conjunto muito extenso de ambientes de problemas, bastando que o subagente seja retreinado para o novo ambiente e dependendo do caso, há a necessidade da escrita de uma nova classe (no exemplo exposto, a procControler) para a comunicação com o recurso real representado pelo Objeto Gerenciado.
- A adaptação apresentada pelo subagente traz como conseqüência, uma generalização do ambiente do problema. No caso do nosso exemplo, se num determinado momento, o Sistema Operacional está com um bom tempo de resposta, a pouca variação dos atributos dos daemons, de modo geral, não necessariamente provoca um efeito muito grande no sistema, permanecendo ainda com um bom tempo de resposta.

A apresentação destas características, justifica o estudo de aplicação de Redes Neurais ao ambiente de gerência de redes, em particular, procurando-se tirar proveito da sua característica adaptativa. Entretanto, é importante um estudo mais profundo sobre outros modelos de redes neurais, principalmente redes com aprendizado não supervisionado[14].

Um outro passo na continuidade deste estudo, é a realização de mais testes procurando identificar quais as situações aplicáveis ou não a uma gerência adaptativa. A classificação destas situações pode validar ou não o uso de redes neurais em gerência de redes e descobrir novas diretrizes para o seu emprego.

8. Referências Bibliográficas

01 ALEKSANDER, I. e MORTON, H. An Introdution to Neural Networking. London: Chapman & Hall, 1993.

02 BLUM, A. Neural Networks In C++ - An Object-Oriented Framework for Building Connectionist Systems. New York: John Wiley & Cons, 1992

03 COMER, D.E e Stevens, D. L. Internetworking with TCP/IP, v.1. New Jersey: Prentice Hall

04 COMER, D.E e Stevens, D. L. Internetworking with TCP/IP, v.2. New Jersey: Prentice Hall

05 CONGRESSO BRASILEIRO DE REDES NEURAIS. (1.:1994: Itajubá). Anais. Itajubá:CEMIG, 1994.

06 KOSKO, B. Neural Networks and Fuzzy Systems - A Dynamical Systems Approach to Machine Intelligence. New Jersey: Prentice-Hall Inc., 1992.

07 PAO,Y-H. Adaptive Pattern Recognition and Neural Networks. New York: Addison Wesley Publishing Company, 1989.

08 ROSE, M. T. The Simple Book - An Introdution to Management of TCP/IP - based internets. New Jersey: Prentice Hall, 1991.

09 ROSE, M. T. The Simple Book - An Introdution to Internet Management- Second Edition. New Jersey: Prentice Hall, 1991.

10 ROSE, M. T. Structure and Identification of Management Information for TCP/IP based internets.Request for Comments 1156. DDN, 1990.

11 ROSE, M. T. Management Information Base Network of TCP/IP based internets: MIB- II.Request for Comments 1158. DDN, 1990.

12 SCHOFFSTALL, M. L. e CASE, J. D e FEDOR, M. S A Simple Network Management Protocol .Request for Comments 1157. DDN, 1990.

13 SARI, S.T. Protótipo de Um Sistema de Reconhecimento de Padrões Conexionista Híbrido. Florianópolis, 1994. Dissertação de Mestrado em Engenharia de Produção - Universidade Federal de Santa Catarina.

14 SARI, S.T. e Loesh, C. Redes Neurais Artificiais - Fundamentos e Modelos. Blumenau:Editora da FURB, 1996.

15 SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES. (13.:1995:Belo Horizonte) Anais. Belo Horizonte:UFMG.

16 SOARES, L.F e LEMOS, G. e COLCHER S. Redes de Computadores - Das LANS, MANS e WANS às Redes ATM. Rio de Janeiro: Editora Campus, 1995.

17 TANENBAUM, A. S. Redes de Computadores - Segunda Edição. Rio de Janeiro: Editora Campus, 1994.

18. VIEIRA, E.M. e BOSCHETTA, M.C. e WESTPHALL, C. B. Controle de Acesso em Gerência de Segurança para a redeUFSC - Anais do SBRC 95. Belo Horizonte.

19 WASSERMAN, P.D. Neural Computing - Teory and Pratice. New York: Van Nostrand Reinhold, 1989.

20 WIDROW,B. 30 Years of Adaptive Neural networks: Perceptron, Madaline, and Backpropagation. IEEE, Vol. 78, n. 9, 1990. P.1415-42.