AUTARQUIA ASSOCIADA À UNIVERSIDADE DE SÃO PAULO

IDENTIFICAÇÃO DE PADRÕES EM SISTEMAS SUPERVISÓRIOS DE INSTALAÇÕES DE REATORES NUCLEARES E EM SISTEMAS DE GASODUTOS UTILIZANDO MAPAS AUTO-ORGANIZÁVEIS

WALDEMAR DORASKEVIČIUS JUNIOR

Dissertação apresentada como parte dos requisitos para obtenção do Grau de Mestre em Ciências na Área de Tecnologia Nuclear - Reatores.

Orientador:
Dr. Benedito Dias Baptista Filho

São Paulo
2005

INSTITUTO DE PESQUISAS ENERGÉTICAS E NUCLEARES
Autarquia associada à Universidade de São Paulo

IDENTIFICAÇÃO DE PADRÕES EM SISTEMAS SUPERVISÓRIOS DE INSTALAÇÕES DE REATORES NUCLEARES E EM SISTEMAS DE GASODUTOS UTILIZANDO MAPAS AUTO-ORGANIZÁVEIS

WALDEMAR DORASKEVIČIUS JUNIOR

Dissertação apresentada como parte dos requisitos para obtenção do Grau de Mestre em Ciências na Área de Tecnologia Nuclear – Reatores

Orientador:
Dr. Benedito Dias Baptista Filho

SÃO PAULO
2005

INSTITUTO DE PESQUISAS ENERGÉTICAS E NUCLEARES
Autarquia associada à Universidade de São Paulo

IDENTIFICAÇÃO DE PADRÕES EM SISTEMAS SUPERVISÓRIOS DE INSTALAÇÕES DE REATORES NUCLEARES E EM SISTEMAS DE GASODUTOS UTILIZANDO MAPAS AUTO-ORGANIZÁVEIS

WALDEMAR DORASKEVIČIUS JUNIOR

Dissertação apresentada como parte dos requisitos para obtenção do Grau de Mestre em Ciências na Área de Tecnologia Nuclear – Reatores

Orientador:
Dr. Benedito Dias Baptista Filho

SÃO PAULO
2005

A todos os pesquisadores dos
países em desenvolvimento

AGRADECIMENTOS

Agradeço ao meu orientador Benedito Dias Baptista Filho, pessoa de extrema capacidade e solícita. Com disposição e tolerância aos gênios e geniosos.

À Petrobras, especificamente à Tecnologia de Informação, Regional São Paulo – Sul, por me ceder tempo e recursos para este trabalho, à custa de amputações de homens-hora de outros projetos.

À Comissão de Pós-Graduação (CPG) do Instituto de Pesquisas Energéticas e Nucleares – IPEN, por ter compreendido a necessidade de estender o tempo nesta Dissertação, e de me aturar com tantos pedidos e atrasos.

Aos docentes e pesquisadores do Instituto de Pesquisas Energéticas e Nucleares – IPEN e da Escola Politécnica da Universidade de São Paulo – EPUSP, pelos conhecimentos transmitidos e pela paciência de me atender e dirimir dúvidas.

À Fundação de Amparo à Pesquisa do Estado de São Paulo – FAPESP, pelo apoio financeiro no projeto para viabilizar o desenvolvimento de um protótipo, Processo n° 04/05150-1.

Aos colegas Renato Emeric Levai, Jacir Massayuki Murasaki, Valéria Boquinpani, Jorge "do Limão", Diana Rosa da Silva Tavares, André Martins Tude, Lilian Alves Martins e Carlos César Rodrigues de Souza pelo apoio moral e intelectual dado a este trabalho.

À minha mãe, Regina Smera Doraskevičius.

E concluo, agradecendo a todos aqueles que não foram citados, mas que ajudaram no andamento e conclusão deste trabalho.

Ao rumar para o estudo das redes neurais, percebi que, além dos mecanismos qualitativos, é muito forte o aspecto quantitativo para o sucesso de uma inteligência, natural ou artificial. Portanto, o fenômeno inteligência não é tão inatingível. Daí, concluo que, a inteligência, sendo fácil, precisamos ir além.

IDENTIFICAÇÃO DE PADRÕES EM SISTEMAS SUPERVISÓRIOS DE INSTALAÇÕES DE REATORES NUCLEARES E EM SISTEMAS DE GASODUTOS UTILIZANDO MAPAS AUTO-ORGANIZÁVEIS

Waldemar Doraskevičius Junior

RESUMO

Mapas Auto-Organizáveis (*Self-Organizing Map*, SOM) de Kohonen foram estudados, implementados e testados com o objetivo de desenvolver, para a área de energia, uma ferramenta eficaz especialmente para a identificação de transientes em reatores nucleares e para a supervisão logística de redes de gasodutos, classificando operações e identificando transientes ou anormalidades. O sistema digital para o teste foi desenvolvido em plataforma Java, pronto para a Web, pela portabilidade e escalabilidade, e por pertencer às plataformas de desenvolvimento livre. Executado em computadores pessoais, o sistema mostrou resultados satisfatórios no auxílio à tomada de decisões ao classificar condições de operação (dados provenientes de simulador) do reator IRIS (*International Reactor Innovative and Secure*) e ao classificar qualidades de operação na malha Sudeste (da Transpetro – Brasil) de gasoduto. Muitas adaptações foram necessárias para os negócios em questão, como a utilização de novas topologias para a camada de saída da rede neural artificial e preparações especiais dos dados de entrada.

PATTERNS IDENTIFICATION IN SUPERVISORY SYSTEMS OF NUCLEAR REACTORS INSTALLATIONS AND GAS PIPELINES SYSTEMS USING SELF-ORGANIZING MAPS

Waldemar Doraskevičius Junior

ABSTRACT

Self-Organizing Maps, SOM, of Kohonen were studied, implemented and tested with the aim of developing, for the energy branch, an effective tool especially for transient identification in nuclear reactors and for gas pipelines networks logistic supervision, by classifying operations and identifying transients or abnormalities. The digital system for the test was developed in Java platform, for the portability and scalability, and for belonging to free development platforms. The system, executed in personal computers, showed satisfactory results to aid in decision taking, by classifying IRIS (International Reactor Innovative and Secure) reactor operation conditions (data from simulator) and by classifying Southeast (owner: Transpetro – Brazil) gas pipeline network. Various adaptations were needed for such business, as new topologies for the output layer of artificial neural network and particular preparation for the input data.

SUMÁRIO

Página

LISTA DE TABELAS

Página

LISTA DE FIGURAS

Página

LISTA DE ABREVIATURAS E SIGLAS

.class	–	Arquivo Java compilado
.doc	–	Arquivo compatível com Microsoft Word
.java	–	Arquivo texto com código-fonte Java
.m	–	Extensão de nome de aquivo significando script para o MATLAB
.txt	–	Extensão de nome de arquivo siginificando texto sem formatação
.xls	–	Arquivo compatível com Microsoft Excel
ADALINE	–	*Adaptative Linear Element*
ANP	–	Agência Nacional do Petróleo, Gás Natural e Biocombustíveis
ATP	–	Adenosina trifosfato
BD	–	Banco de dados
BDEMQ	–	Banco de Dados de Estoque, Movimentação e Qualidade
BDO	–	Banco de Dados de Informações Operacionais
CPU	–	*Central Processing Unit*
CY	–	*City-Gate*
Eq.	–	Equação
GASAN	–	Gasoduto Mauá - Santos
GASBEL	–	Gasoduto Rio de Janeiro - Belo Horizonte
GASBOL	–	Gasoduto Brasil - Bolívia
GASENE	–	Gasoduto Sudeste - Nordeste
GASPAL	–	Gasoduto Mauá - Paulínia
GN	–	Gás natural
HTML	–	*Hyper Text Markup Language*
HTTP	–	*Hypertext Transfer Protocol*
I/O	–	*Input/output* (fluxo de entrada/saída de dados de/para um arquivo)
iGás	–	Sistema Integrado de Gás Natural
iLab	–	*Information Laboratory*
IRIS	–	*International Reactor Innovative and Secure*
ISO	–	*International Organization for Standardization*

J2SE	_	*Java 2 Platform, Standard Edition*
JDBC	_	*Java Database Connectivity*
JRE	_	*Java Runtime Environment*
JSP	_	*Java Server Pages*
JVM	_	*Java Virtual Machine*
LMS	_	*Least Mean Square*
LOCA	_	*Loss of Coolant Accident*
LTP	_	*Long-Term Potentiation*
MATLAB	_	*Matrix Laboratory*
MB	_	Megabyte
MME	_	Ministério de Minas e Energia
PCA	_	*Principal Component Analysis*
PCI	_	Poder Calorífico Inferior
PCS	_	Poder Calorífico Superior
PI	_	*Plant Information*
QDC	_	Quantidade Diária Contratada
QDP	_	Quantidade Diária Programada
R/3	_	*Software real time* em arquitetura 3 camadas
RAM	_	*Random Access Memory*
RECAP	_	Refinaria de Capuava
REM	_	*Rapid Eye Movement*
REPLAN	_	Refinaria de Paulínia
RNA	_	Rede Neural Artificial
RPBC	_	Refinaria Presidente Bernardes - Cubatão
SAG	_	Sistemas de Atividades do Gás
SAP	_	*System, Anwendungen, Produkte in der Datenverarbeitung*
SCADA	_	*Supervisory Control and Data Acquisition*
SCM	_	Superintendência de Comercialização e Movimentação da ANP
SCRAM	_	*Safety Control Rod Axe Man*
SDK	_	*Software Development Kit*
SDP	_	Superintendência de Desenvolvimento e Produção da ANP
SGBD	_	Sistema Gerenciador de Banco de Dados
SIG	_	Sistema de Informação Geográfica

SOFM	_	*Self-Organizing Feature Map*
SOM	_	*Self-Organizing Map*
SQL	_	*Structure Query Language*
SRP	_	Superintendência de Refino e Processamento de Gás Natural da ANP
TBG	_	Transportadora Brasileira Gasoduto Bolívia-Brasil S. A.
TCP/IP	_	*Transmission Control Protocol/Internet Protocol*
XML	_	*Extensible Markup Language*

LISTA DE SÍMBOLOS

A	_	espaço discreto de saída
b	_	fator de ajuste temporal
C	_	quantidade de componentes ou sinapses
C_R	_	fator de ajuste do numerador do argumento da exponencial da gaussiana
$d_{j,i}$	_	distância entre as unidades neurais i e j
$\mathbf{d}^p$	_	valor devido ou desejado para uma saída
$e(y_j)$	_	termo de esquecimento
E^p	_	função de energia ou função de erro quadrático
g	_	função de ativação ou de compressão
h	_	subscrito, refere-se à(s) camada(s) oculta(s) (*hidden*) da rede neural
$h_{j,i(\mathbb{x})}$	_	função de vizinhança aplicada entre a unidade vencedora i e j, para $\mathbf{x}$
i	_	subscrito que se refere à camada de entrada (*input*) - componentes
$i(\mathbf{x})$	_	índice do neurônio que possui o vetor peso mais próximo de $\mathbf{x}$
n	_	número de iteração (época)
N_L	_	número de unidades neurais na camada L (*layer*)
o	_	subscrito que se refere à camada de saída (*output*) de uma rede neural
p	_	padrão amostral
P	_	pressão
$\mathbb{I}_j$	_	posição topológica da unidade neural j
s_j	_	intensidade da entrada para uma unidade neural j
t	_	tempo
T	_	temperatura
V	_	volume
$\mathbf{w}^j$	_	vetor de código ou peso da unidade neural j
$\boldsymbol{X}$	_	conjunto dos vetores de entrada
X	_	espaço contínuo de entrada
$\mathbf{x}^p$	_	vetor de entrada ou de característica p
$\mathbf{y}_L{}^p$	_	saída da camada L
α	_	fator do termo que determina a influência da atualização anterior na atual

$\delta_k{}^p$ _ regra delta generalizada ($\delta_h^p = g'(s_h^p)\sum_{o=1}^{N_o}\delta_o^p w_{ho}$)

$\delta_o{}^p$ _ regra delta generalizada para camada de saída ($\delta_o^p = \left(d_0^p - y_o^p\right)g_o'\left(s_o^p\right)$)

δ^p _ regra delta, diferença entre a saída alvo e a saída real para o padrão p

ζ_c _ fator de sensibilidade para a componente c do vetor de entrada

$\eta(n)$ _ taxa de aprendizagem ($\eta(0) = \eta_0$)

θ _ potencial de corte ou limiar de ativação

$\sigma(n)$ _ parâmetro de largura efetiva da vizinhança topológica ($\sigma(0) = \sigma_0$)

τ _ constante de tempo

$\mathbf{\Phi}$ _ mapa de característica, resultante de transformação não linear

Ψ _ conjunto de amostras com entradas com os valores desejados

1 INTRODUÇÃO

Para fazer frente ao desafio de se extrair informação de volumes cada vez maiores de dados, se faz necessário o desenvolvimento de ferramentas que nos ofereçam a capacidade de analisar com rapidez e eficácia essa grande massa de dados. Atualmente a geração de dados ocorre via as muitas interfaces amigáveis com o usuário ou pela automação, com sensores de resoluções sempre crescentes. O armazenamento fica por conta das tecnologias de banco de dados como a compressibilidade e otimização das *Plant Informations*, os SGBDs (Sistemas Gerenciadores de Banco de Dados) – como Oracle e MySql – e os SIGs (Sistemas de Informação Geográfica). A recuperação fácil dos dados se deve às linguagens não-procedurais de consulta às bases como a SQL (*Structured Query Language*) e às interfaces amigáveis com o usuário para a seleção de dados.

As maneiras de transformar a massa de dados em informação podem ser estatísticas ou através das diversas abordagens de inteligência artificial e de sistemas complexos. A inteligência artificial foi o caminho escolhido para esta dissertação que envolve a computação aplicada à geração e distribuição de energia, abordando as redes neurais artificiais (RNA), mais especificamente uma rede neural para classificação de padrões, os Mapas Auto-Organizáveis, do tipo desenvolvido por Teuvo Kohonen a partir de 1979.

Os modelos estatísticos continuam sendo os mais utilizados até hoje, porém possuem o inconveniente de exigir hipóteses *a priori* e atuam como filtros lineares, como conseqüência alteram as características dos dados, podendo esconder padrões e acidentalmente criar outros (Openshaw e Turton, 1996). Já as redes neurais artificiais não são mecanismos automáticos de análise exploratória de dados, não necessitam de nenhuma hipótese *a priori* e não são métodos lineares.

Na RNA do tipo Mapa Auto-Organizável ou SOM[1] ocorre competição entre seus neurônios, como em algumas áreas do córtex cerebral, tais como táctil, visual, auditivo e olfativo (Nadel *et al.*, 1989). Nessas regiões os neurônios formam agrupamentos ordenados topologicamente, neurônios vizinhos se comunicam por colaterais e tendem a

[1] Self-Organizing Map

responder a padrões semelhantes. A rede neural de um SOM deve possuir dimensão espacial menor ou igual a do espaço de entrada (ou vetor de dados) para a qual estes serão mapeados, compactando os dados para dimensões menores, assemelhando-se à Análise de Componente Principal, porém não-linear. Assim a RNA do SOM identifica os padrões de entrada, agrupa e correlaciona os dados a regiões específicas da grade de neurônios. É por essa ordenação topológica que o SOM é considerado uma boa ferramenta para análise exploratória de dados (Openshaw e Turton, 1996).

A análise de processos termohidráulicos de reatores nucleares é uma área em que o SOM vem sendo largamente utilizado (Baptista, 2002), objetivando o desenvolvimento de sistemas de supervisão inteligentes capazes de alertar quando do surgimento de padrões desconhecidos ou relacionados a situações indesejáveis. Posteriormente, pesquisas de eficácia do SOM foram aplicadas para a análise de transientes de reatores nucleares, incluindo o reator de nova geração IRIS[2] (Baptista e Barroso, 2004). A utilização da energia nuclear para geração apresenta-se como uma importante alternativa contra a emissão de gases que colaboram para o efeito estufa (Brasil Nuclear, 2004). Além disso, o seu desenvolvimento proporciona uma estratégica tecnologia nacional capaz de ser aplicada a diversas áreas (Eletronuclear, Energia Nuclear no Brasil, 2003). Neste trabalho, além de estender a aplicação do SOM ao IRIS, consórcio de pesquisa do qual o Brasil faz parte, estendemos para a supervisão logística do gás natural, dessa maneira também colaborando com o programa brasileiro de incentivo do uso do gás natural, que possui a pretensão de que o gás natural aumente sua participação na matriz energética brasileira – expectativa de passagem dos 7,5% atuais para 15% em 2015, criando 200 mil empregos e movimentando US$ 4,5 bilhões até 2010 (Portal Gás&Energia, Reservas Nacionais, 2004).

[2] International Reactor Innovative and Secure

2 OBJETIVOS

O objetivo desta dissertação foi lançar uma nova e promissora ferramenta no auxílio à tomada de decisões durante o decorrer de um processo, mais especificamente, processos ligados à energia, sua produção e seu transporte num conceito amplo. Ferramenta esta que faz uso do Mapa Auto-Organizável de Kohonen para a extração de padrões a partir de um conjunto de entradas para assim analisar o comportamento e identificar a classe de novas entradas. O desenvolvimento desta ferramenta parte de estudos anteriores (Baptista e Barroso, 2003, 2004) que apontaram o uso do Mapa Auto-Organizável como promissor para reatores nucleares. Agora, neste trabalho, será construída em outra plataforma de sistemas digitais e estendida para outros processos.

São objetivos específicos e imediatos:

a) Construir o algoritmo de aprendizado não-supervisionado para o Mapa Auto-Organizável de Kohonen;
b) Projetar uma arquitetura de sistema digital;
c) Implementar o algoritmo para a arquitetura em plataforma de desenvolvimento livre, no caso, Java 2;
d) Realizar diversos treinamentos do sistema; e
e) Levantar sua eficácia e eficiência na classificação de padrões escondidos dentro um conjunto elevado de entradas com muitas dimensões.

São metas futuras:

a) Integrar este sistema, como módulo, com sistemas Web de supervisão logística do gás natural; e
b) Integrar este sistema, como módulo, a outros sistemas redundantes de análise de transientes de reatores nucleares.

3 REVISÃO DA LITERATURA

3.1 Redes Neurais Artificiais

Uma Rede Neural Artificial (RNA) é constituída de unidades neurais ou elementos com conexões entre eles. Em analogia com o sistema nervoso, unidade neural seria o neurônio e as conexões seriam as sinapses. Uma conexão vem a ser a interação de uma unidade com uma entrada através de uma ponderação desta por um valor-peso e por um limiar de ativação, ou corte. Diante da apresentação de uma entrada ou estímulo, espera-se uma determinada resposta da rede – a saída. Esta resposta esperada se dará mediante treinamento ou ensino da rede, frente à apresentação de um conjunto de entradas. O treinamento pode ser supervisionado (a) ou não-supervisionado (b): (a) no treinamento supervisionado o instrutor fornece uma entrada, e a saída será comparada com um valor devido, um valor de referência ou um valor desejado para aquela entrada. Cada unidade neural terá os pesos das conexões e limiar corrigidos de acordo com erros ou distâncias para que diminuam na próxima iteração, esse pode ser um processo de correção de erro ou um processo de minimização de uma função custo; (b) no treinamento não-supervisionado, os pesos serão atualizados de acordo com a similaridade com a própria entrada, de modo que, a cada entrada apresentada, somente uma unidade será a vencedora e as demais unidades, principalmente as vizinhas, poderão sofrer a influência dela – caso em que a atualização dos pesos segue um processo de competição; ou os pesos podem ser atualizados em função dos sinais de entrada e saída de cada unidade neural – caso em que se utiliza o algoritmo hebbiano de aprendizado.

3.1.1 A inspiração biológica

O cérebro humano possui neurônios em quantidade da ordem de 100 bilhões, onde cada neurônio pode se conectar a milhares de outros, de 10 até 100 mil conexões, constituindo assim o sistema mais complexo do Universo. Essa quantidade imensa de neurônios faz nosso cérebro possuir extrema redundância. Filogeneticamente o cérebro pode ser dividido em três grandes componentes: primitivo (manutenção fisiológica, autopreservação, agressão), intermediário (emoções) e racional (intelecto), a cada são

associadas estruturas cerebrais típicas do centro para o exterior, respectivamente; porém se interconectam, inclusive nas funções. Quanto ao cérebro racional, sua estrutura de maior atividade é o córtex, objeto de maior atenção desta dissertação. Da gestação até os dois anos de idade, ocorre um crescimento elevado do número de neurônios (FIGURA 3.1), a partir daí, ao longo da vida, vamos perdendo algumas dezenas de milhares de neurônios por dia, sem reposição em termos de quantidade. Esta perda ocorre por uma necessidade básica de economia de energia - o cérebro consome cerca de 20% da energia do corpo. Uma vez que determinados caminhos neurais estejam consolidados através das conexões sinápticas, os neurônios excedentes, menos utilizados, podem ser eliminados, graças à redundância.

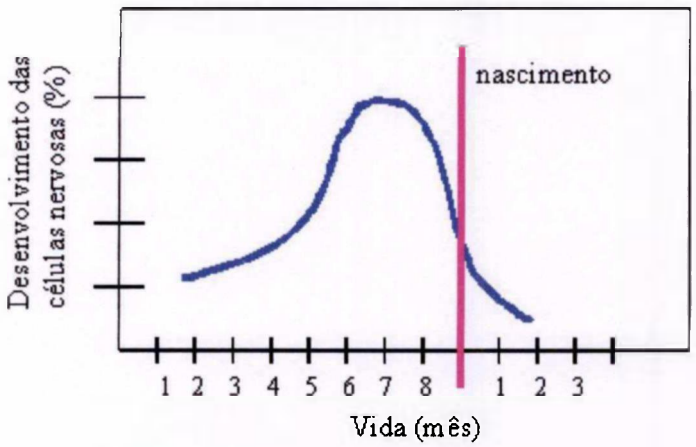


FIGURA 3.1 - Taxa de crescimento do número de células nervosas para humanos, durante a gestação e o pós-natal.

O cérebro primitivo é constituído pelas estruturas do tronco cerebral – bulbo, ponte e mesencéfalo, pelo cerebelo e pelos glânglios basais – neostriatum e globo pálido. Esse cérebro, delimitado pelas estruturas descritas, corresponde ao cérebro dos répteis (FIGURA 3.2).

O cérebro intermediário surgiu nos primitivos mamíferos, é atribuído às estruturas do sistema límbico – amígdala, hipocampo, tálamo, hipotálamo, giro cingulado, septo e bulbos olfatórios (FIGURA 3.3).

O cérebro superior, encontrado nos mamíferos modernos, como os primatas, cetáceos e humanos, compreende a maior parte dos hemisférios cerebrais, formado por um córtex mais recente, o neocórtex, e por alguns grupos neuronais subcorticais (FIGURA 3.4).

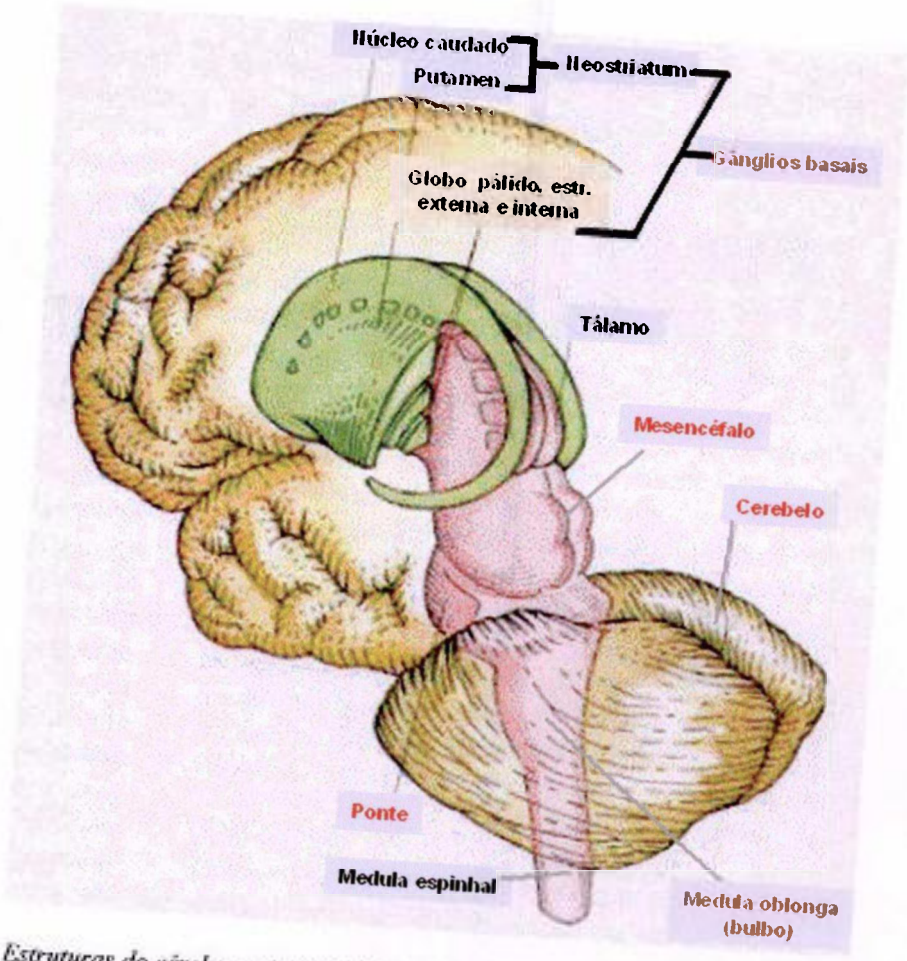


FIGURA 3.2 - Estruturas do cérebro primitivo. Por continuidade está exibido o tálamo e a medula espinhal.

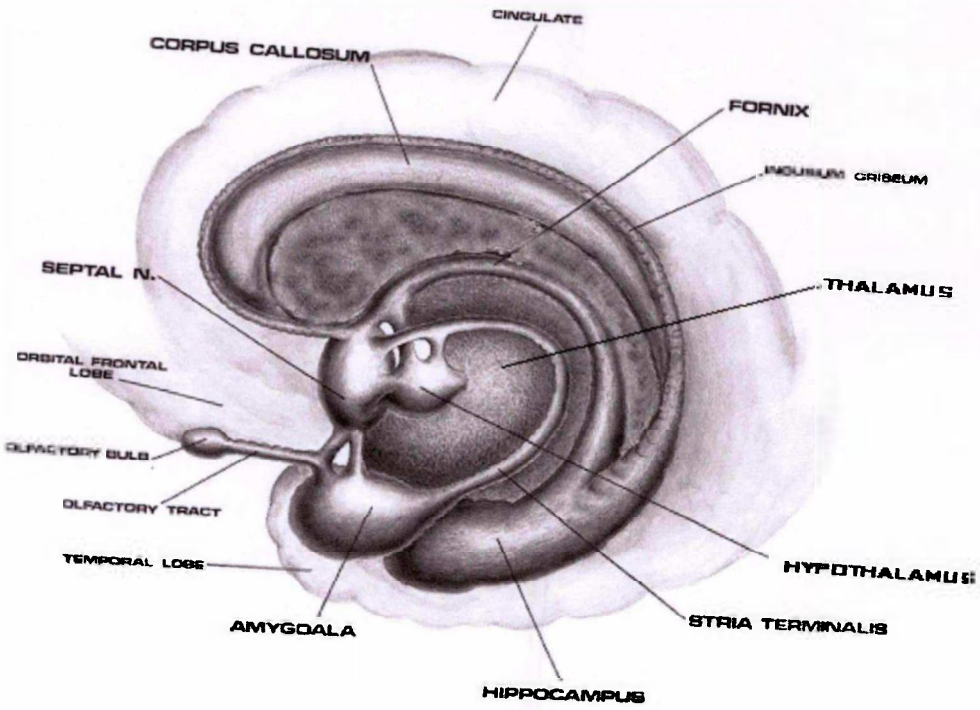


FIGURA 3.3 - Corte sagital do cérebro humano, destacando o cérebro intermediário.

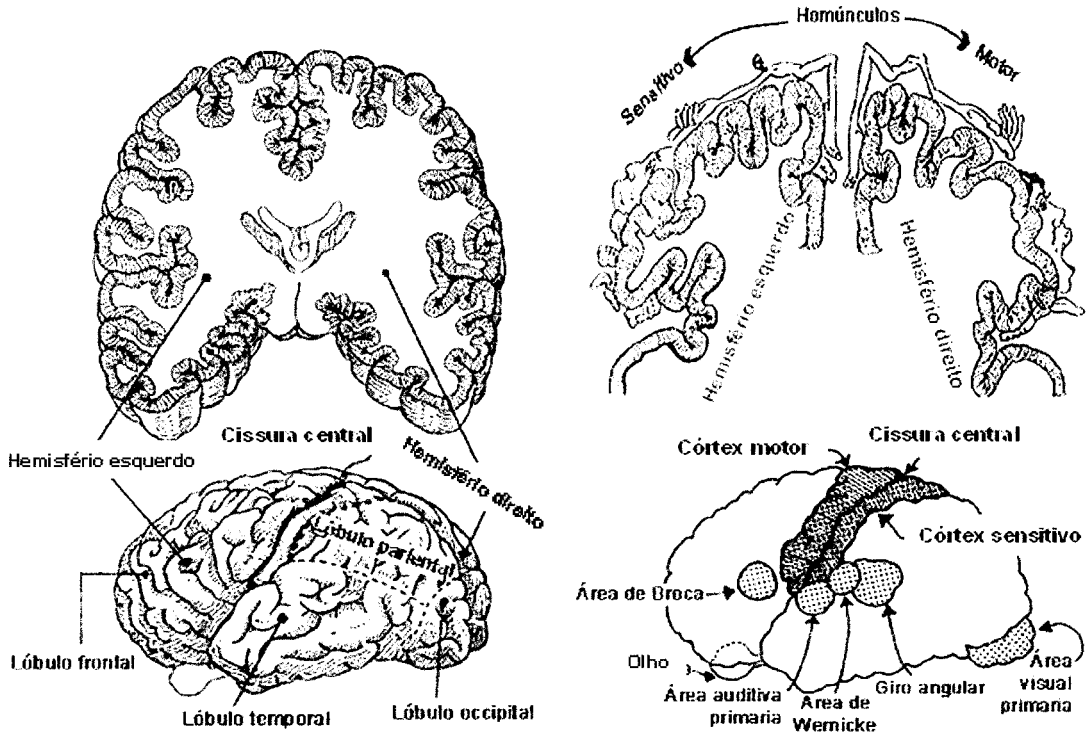


FIGURA 3.4 – Cérebro superior humano, neocórtex e estruturas subcorticais (desenho inferior direito).

O neurônio é uma célula especializada e altamente diferenciada, causa da sua baixa capacidade reprodutiva (FIGURA 3.5). Ao receber estímulo nos dendritos (filamentos sensíveis a partir do corpo celular ou soma) transmite impulsos em determinada freqüência e intensidade pelo axônio, caracterizando o sinal. A transmissão é unidirecional na célula e é feita por despolarização variável da membrana (FIGURA 3.6). Entre os neurônios, a comunicação ocorre através de ligações contíguas chamadas sinapses, via emissão de moléculas neurotransmissoras (pré-sinapse). Os neurotransmissores, dentro do neurônio, se concentram em vesículas sinápticas, que por sua vez estão nos bulbos ou botões sinápticos na ponta dos terminais axônicos (FIGURA 3.7). São capturados pelos neuroreceptores nas espinhas dendríticas (pós-sinapse). Neurotransmissores e neuroreceptores formam um modelo molecular do tipo chave-fechadura. Quando se mencionar "conexão na rede neural artificial", o análogo biológico é essa conexão sináptica.

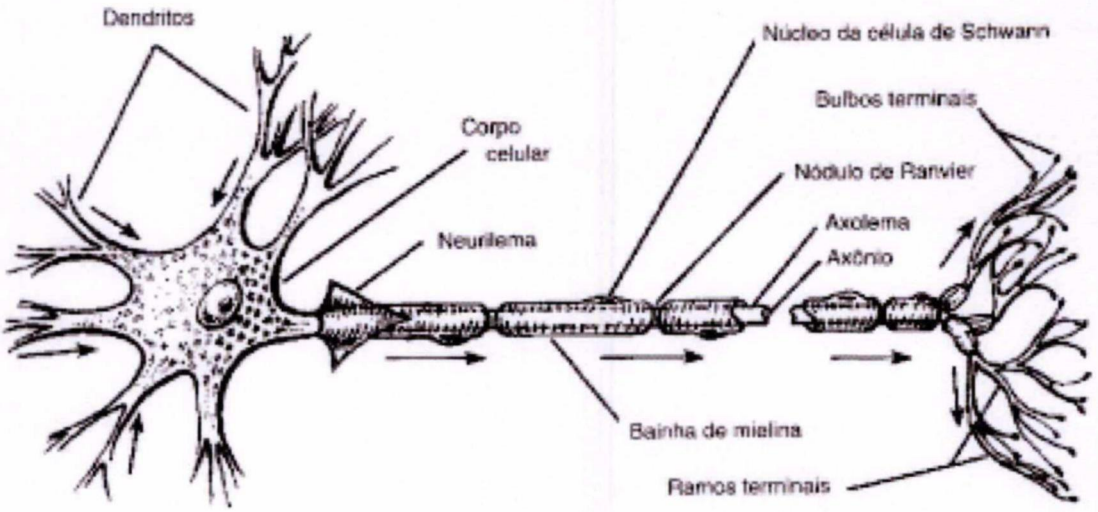


FIGURA 3.5 - Anatomia de um neurônio.

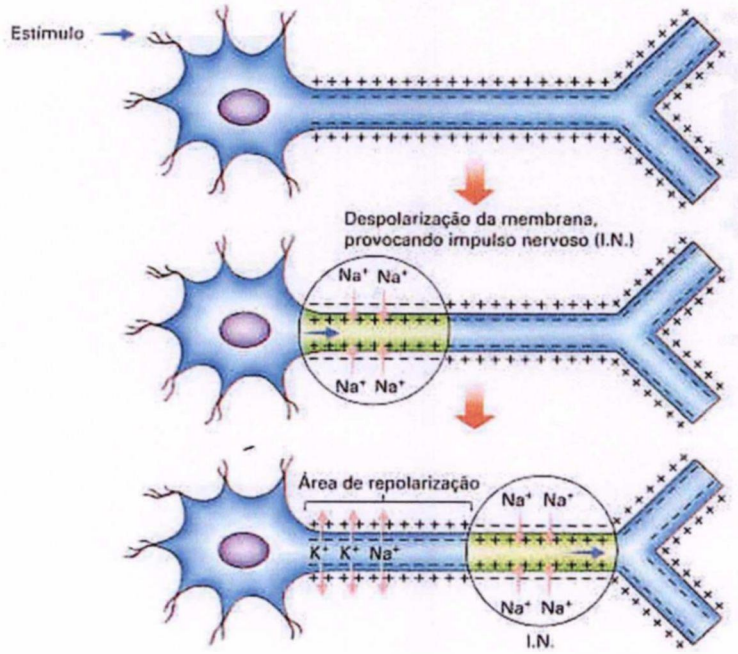


FIGURA 3.6 - Bomba de sódio (+) - potássio (-) que modifica o potencial local. Bomba que funciona à custa de ATPs.

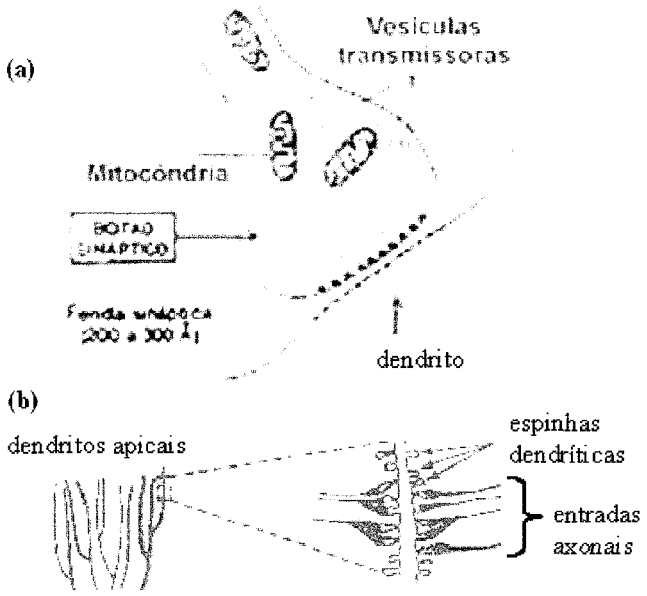


FIGURA 3.7. – Sinapse em detalhes. (a) Botão ou bulbo sináptico na ponta dos terminais axônicos. (b) Espinhas dentríticas em torno de segmento de dendrito.

O neurônio somente será estimulado se for ultrapassado um limiar de ativação (FIGURA 3.8), dado pela somatória de todas as intensidades das conexões. Cada intensidade de conexão é uma combinação da quantidade de neurotransmissor (intensidade de entrada) com a quantidade de neuroreceptor (valorização da entrada ou peso da entrada).

Segundo o postulado de Hebb (1949), quando um estímulo causa a ativação de dois neurônios simultaneamente – pré-sinapse em um e pós-sinapse no outro – a conexão sináptica entre eles se reforça, o que leva à propriedade do mecanismo local, ou seja, os neurônios não necessitam de um nível superior regulador, são unidades autônomas dentro do processo cognitivo.

A "simultaneidade", chamada hoje de processo síncrono, a que se refere Hebb pode hoje ser traduzida como estímulo de longa duração ou LTP (*long-term potentiation*). Ocorre quando a freqüência da emissão de neurotransmissores está sintonizada com a freqüência máxima com que o neurônio receptor pode se estimular (FIGURA 3.9). Essa freqüência máxima, acima da qual nenhum reforço da conexão é realizado, existe devido ao período de refração: tempo para a célula voltar ao seu potencial normal de repouso, em torno de –60 mV, após queda de uma ativação de aproximadamente +40 mV para –80 mV. No estado de vigília e no sono REM (*rapid eye movement*), presente nos mamíferos placentários e marsupiais, essa freqüência máxima é de aproximadamente 5 Hz ou período de 200 ms, é o ritmo θ.

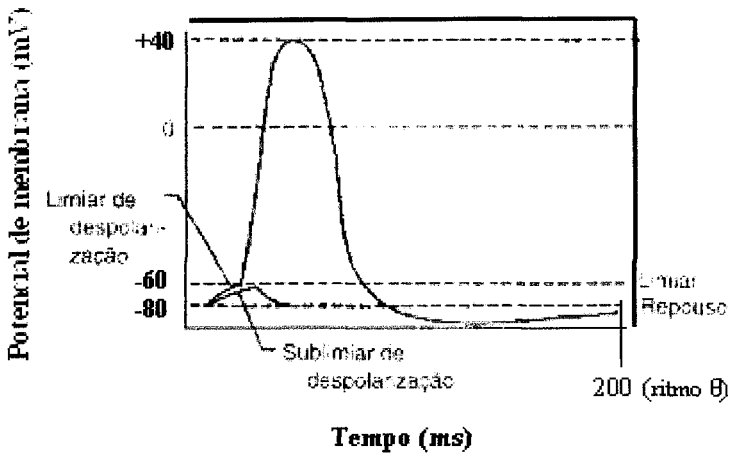


FIGURA 3.8 - Pulso: limiar de ativação e tempo de refração (extensão da curva abaixo do repouso).

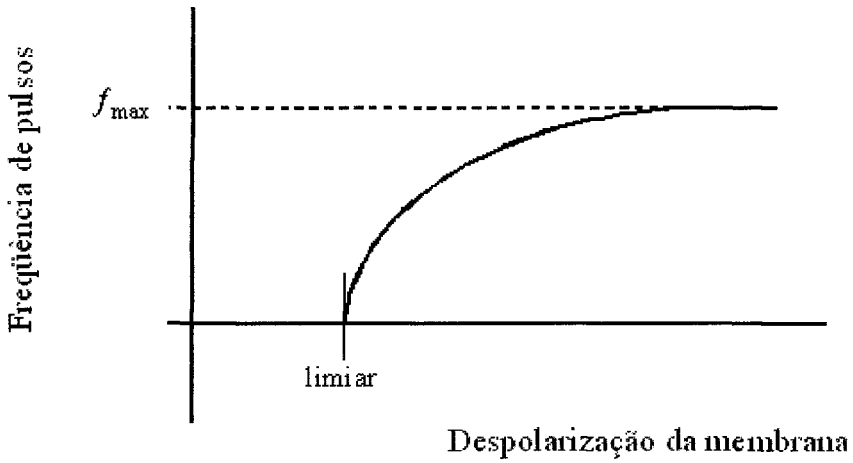


FIGURA 3.9 – Relação entre a freqüência de pulsos de um neurônio e a intensidade do estímulo de longa duração.

3.1.2 A unidade neural para redes artificiais

Seguindo o comportamento de um neurônio, especialmente o modo de ativação e o sinal produzido, as unidades neurais ou elementos de uma rede artificial, fazem a mesma combinação citada no item anterior entre a entrada (**x**) e o valor ou peso (**w**) da conexão, levando também em conta o limiar de ativação θ, para se obter a intensidade de entrada no neurônio (s). Ou seja:

$$s = \sum_{j=1}^{C} w_j x_j + \theta \quad , \tag{3.1}$$

onde N é o número de sinapses ou componentes do elemento.

A intensidade de saída do neurônio será uma função da entrada s, que pode ser linear ou não linear.

$$y = g\left(\sum_{j=1}^{C} w_j x_j + \theta \right). \tag{3.2}$$

O limiar θ pode ser encarado como mais um componente, w_0, do vetor peso se for considerado um componente "mudo", $x_0 = +1$ (ou $x_0 = -1$), do vetor de entrada.

Como exemplo, no caso de g ser uma função sinal (3.3), teremos a unidade se comportando como uma função discriminante linear, separando o espaço de entrada em duas regiões. A rede funciona assim como um classificador.

$$g(s) = \begin{cases} +1, \textit{para } s > 0 \\ \\ -1, \textit{para } s \leq 0. \end{cases} \tag{3.3}$$

Para ficar mais claro, se esta única unidade neural possuir somente duas sinapses e o limiar de ativação θ, então as duas regiões do espaço de entrada ficarão separadas pela reta

$$w_1 x_1 + w_2 x_2 + \theta = 0, \tag{3.4}$$

como mostra o exemplo representado na FIGURA 3.10.

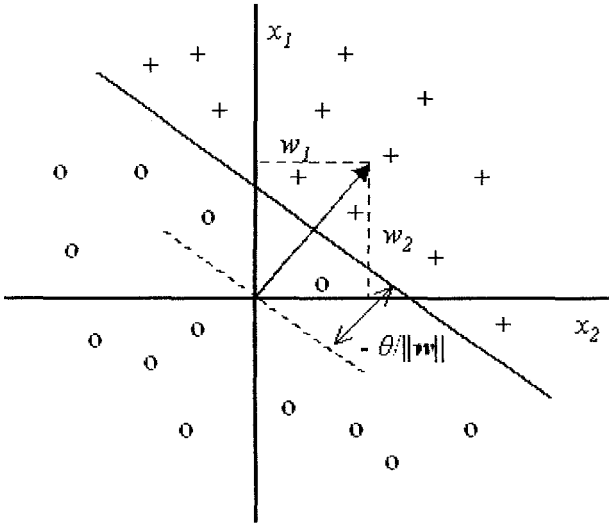


FIGURA 3.10 - Representação geométrica da função discriminante e dos pesos.

Para se chegar ao correto **w**, cada coordenada deve receber um termo corretivo Δw_i. De acordo com a sinapse hebbiana, uma conexão recebe reforço se os dois neurônios são ativados ao mesmo tempo. Dessa maneira o reforço Δw_i será proporcional à entrada x_i e à saída y_o,

$$\Delta w_{io} = \eta\, x_i\, y_o, \tag{3.5}$$

onde η representa a taxa de aprendizado.

Este modelo neural foi adotado por um dos primeiros modelos de redes neurais, o *Perceptron* (Rosenblatt, 1959).

A correção para **w** foi generalizada por Widrow e Hoff (Widrow e Hoff, 1960) e batizada de regra delta:

$$\Delta w_{io} = \eta\, x_i\, (d_o - y_o). \tag{3.6}$$

Ou seja, a cada entrada x, a saída y será comparada com um valor devido d_o, e a diferença entrará no cálculo para correção do vetor peso, juntamente com o valor da entrada e uma taxa de aprendizado, lembrando que o limiar θ pode ser encarado como uma componente w_0 do peso, com entrada $x_0 = +1$ (ou -1).

3.1.3 A regra de aprendizado do perceptron e o teorema de convergência

Seja um conjunto de amostras Ψ de aprendizado consistindo de vetores de entrada $\boldsymbol{x}$ e respectivas saídas desejadas $d_o(\boldsymbol{x})$. Vamos considerar a unidade neural sendo um classificador que atribui +1 ou –1 para cada entrada. A regra de aprendizado para o perceptron é simples e pode ser assim declarada:

1. Inicialize o processo com pesos aleatórios para as conexões;
2. Selecione (de preferência aleatoriamente) um vetor de entrada $\boldsymbol{x}$ do conjunto de amostras de treinamento;
3. Se a saída $y \neq d_o(\boldsymbol{x})$, ou seja, o perceptron forneceu uma resposta errada, modifique todas as conexões w_i de acordo com: $\Delta w_{io} = d_o(\boldsymbol{x})\, x_i$;
4. Retorne ao passo 2.

A única diferença com a regra de Hebb é que quando o neurônio responde corretamente, o peso da conexão não sofre modificações.

A regra de aprendizado do perceptron utiliza o teorema da convergência.

Teorema da convergência. Se existe um conjunto de pesos de conexão w^* apto para realizar a transformação $y = d_o(x)$, a regra de aprendizado convergirá para alguma solução (que pode ou não ser igual a w^*) num número finito de passos, para qualquer escolha inicial dos pesos.

Prova. Uma vez que o comprimento do vetor w^* não desempenha papel por causa da função sinal, considera-se $\| w^* \| = 1$. Porque w^* é a solução correta, o valor de $| w^*. x|$, produto interno, será sempre maior que 0 ou: existe $\delta > 0$ tal que $| w^*. x| > \delta$ para todas as entradas x. Define-se $\cos \alpha \equiv w \, . \, w^*/||w||$. Uma vez que de acordo com a regra de aprendizado do perceptron, os pesos das conexões são modificados para uma dada entrada x, $\Delta w = d^o(x)\, x$, e os pesos após a modificação serão $w' = w + \Delta w$. Daí segue que:

$$w' . w^* = w . w^* + d_o(x) . w^*.x$$

$$= w . w^* + sgn(w^*.x) . w^*.x$$

- $w . w^* + \delta$

$$\|w'\|^2 = \| w + d_o(x) .x\|^2$$

$$= w^2 + 2. d_o(x) . w.x + x^2$$

$$< w^2 + x^2$$

$$= w^2 + M.$$

Após t modificações se obterá:

$$w(t).w^* > w.w^* + t\,\delta$$

$$||w(t)||^2 < w^2 + tM$$

tal que

$$\cos \alpha(t) = \frac{w^*.w(t)}{\|w^*\|.\|w(t)\|},$$

$$\cos \alpha(t) > \frac{w^*.w + t\delta}{\sqrt{w^2 + tM}}.$$

Daqui segue que $lim_{t\to\infty} \cos \alpha(t) = lim_{t\to\infty} \delta\sqrt{t}/\sqrt{M} = \infty$, enquanto, pela definição $\cos \alpha \leq 1$.

Conclusão. Existe um limite máximo t_{max} para t, ou seja, o sistema modifica suas conexões um número limitado de vezes. Iniciando com conexões $w = 0$,

$$t_{max} = \frac{M}{\delta^2}. \qquad (3.7)$$

3.1.4 O Adaline e a retropropagação

Como uma generalização do algoritmo de treinamento do perceptron, foi apresentado, por Widrow e Hoff, em 1960, um procedimento de aprendizado por "mínimos quadrados" (ou LMS[3]), também conhecido por regra delta[4]. A principal diferença funcional do perceptron é o modo como a saída é usada no treinamento. O perceptrom utiliza a saída de uma função de corte (-1 ou +1) para o aprendizado. A regra delta utiliza diretamente a saída líqüida antes de possivelmente mapear para –1 ou +1.

Esta regra de aprendizado foi aplicada para o "elemento linear adaptativo" ou Adaline[5] (FIGURA 3.11). Numa implementação física simples, o aparato consiste num conjunto de resistores controláveis conectados a um circuito que pode somar correntes vindas de sinais de voltagem de entrada. Muitas vezes o bloco central, o somador, é seguido por um quantizador que tem como saída +1 ou –1, dependendo da polaridade da soma.

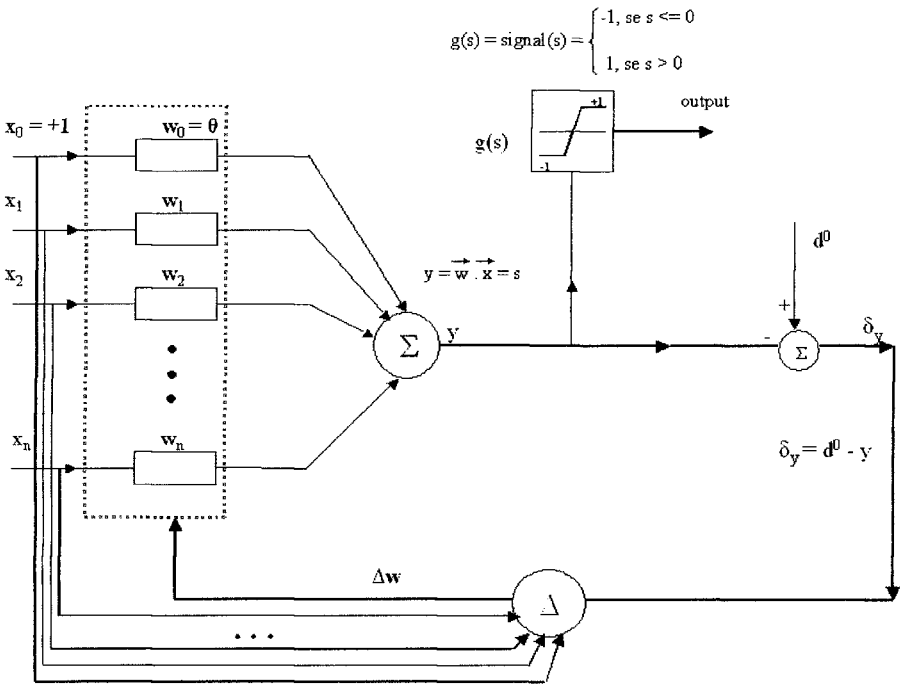


FIGURA 3.11 - O Adaline em implementação simples.

[3] LMS: *least mean square.*
[4] Delta-rule.
[5] A palavra ADALINE foi primeiro acrônimo de *ADAptative LInear NEuron.* Porém quando os neurônios artificiais se tornaram impopulares a partir do final da década de 1960, devido à obra de Minsky & Pappert e a consequente popularização da programação simbólica, ADALINE se tornou acrônimo de *ADAptative LINear Element.*

A regra delta tem origem no método do "gradiente descendente" (Kovács, 1998) que, coloquialmente, pode ser exposto como, "do alto de uma montanha, a trilha mais rápida e descendente para se chegar ao ponto mais baixo de uma montanha" (FIGURA 3.12). A função sobre a qual se fará o gradiente é a chamada "função de energia", dada pela comparação, quadrática, entre o resultado devido (*d*) e o resultado obtido pelo algoritmo de aprendizado numa determinada iteração dos dados,

$$E(\boldsymbol{w}) = \sum_p E^p(\boldsymbol{w}) = \tfrac{1}{2}\sum_p [\boldsymbol{d}^p - \boldsymbol{y}^p]^2, \tag{3.8}$$

onde o índice *p* percorre todo o conjunto de entrada. Esta função de energia é uma função de erro – erro quadrático. Aplicar o método do gradiente descendente sobre ela significa minimizar o erro quadrático, ou equivalentemente, aplicar o método do erro quadrático médio mínimo ou LMS,

$$\Delta_p w_j = -\eta \frac{\partial E^p}{\partial w_j}, \tag{3.9}$$

onde η é uma constante de proporcionalidade e o sinal negativo advém do gradiente descendente. Desenvolvendo a derivada parcial,

$$\frac{\partial E^p}{\partial w_j} = \frac{\partial E^p}{\partial y^p}\frac{\partial y^p}{\partial w_j}. \tag{3.10}$$

Como estamos considerando elementos lineares com função de ativação *g* (eq. (3.2)) identidade,

$$\frac{\partial y^p}{\partial w_j} = x_j, \tag{3.11}$$

e

$$\frac{\partial E^p}{\partial y^p} = -\left(\boldsymbol{d}^p - \boldsymbol{y}^p\right) \tag{3.12}$$

de maneira que

$$\Delta_p w_j = \eta \delta^p x_j, \tag{3.13}$$

onde $\delta^p = (\boldsymbol{d}^p - \boldsymbol{y}^p)$ é a diferença entre a saída alvo e a saída real para o padrão *p*.

A regra delta modifica os pesos apropriadamente tanto para entrada e saída contínuas como para binárias, abrindo uma gama das mais diversas aplicações.

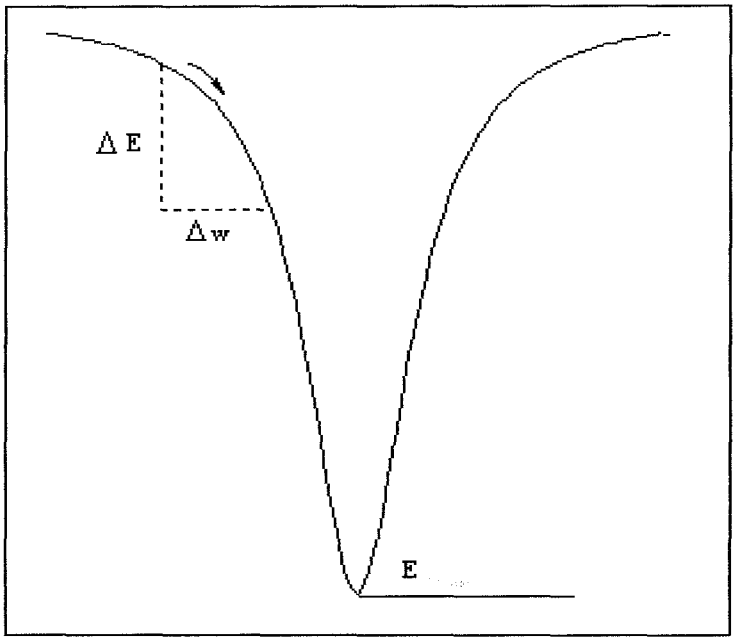


FIGURA 3.12 - Método do 'gradiente descendente'.

Redes neurais com somente a camada de entrada (***input***), ou seja, os componentes da amostragem para aprendizado ou padrões de entrada, e a camada de elementos de saída (***output***) não são suficientes para várias situações de classificação, como o caso do OU-EXCLUSIVO (FIGURA 3.13). Por isso, para muitas aplicações, é recomendada a construção de redes neurais com uma ou várias camadas intermediárias ou ocultas (***hidden***), e a regra delta ou dos mínimos quadrados vai precisar ser generalizada para cada camada oculta (FIGURA 3.14).

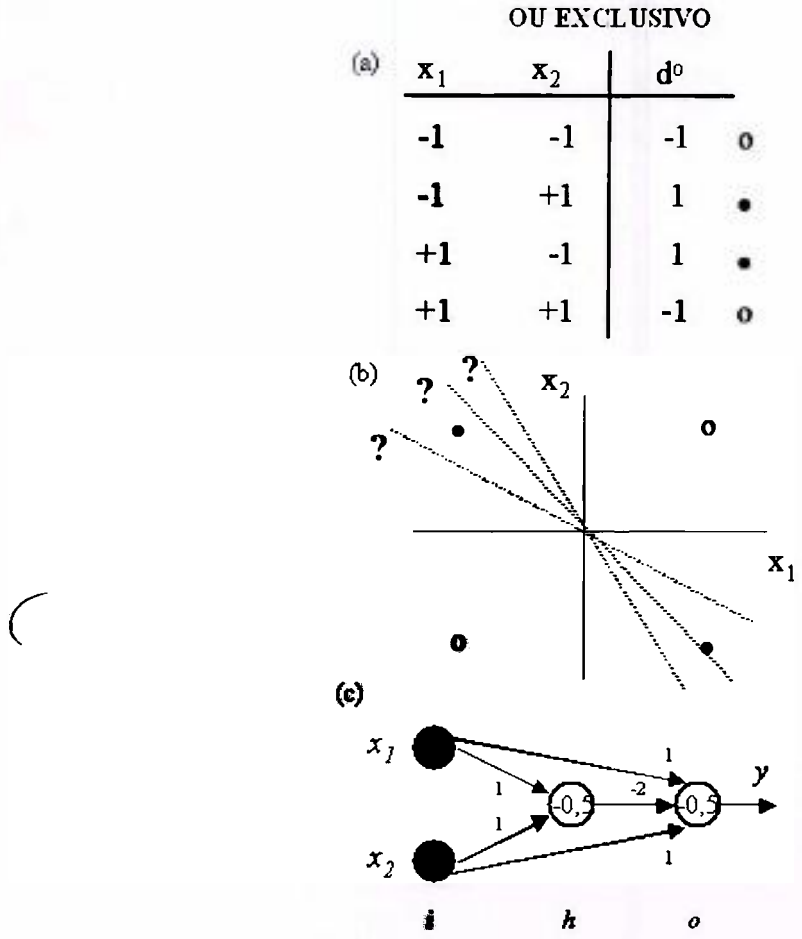


FIGURA 3.13 – O problema do OU EXCLUSIVO e sua solução. (a) Tabela verdade do ou-exclusivo. (b) Representação geométrica do espaço de entrada para ou-exclusivo. (c) Uma solução do ou-exclusivo para redes neurais. Foi preciso a inclusão de uma camada intermediária neste perceptron.Os valores dentro das unidades neurais são para o limiar e, fora, são para os componentes do peso. Ao contrário desse exemplo, uma rede neural multicamada "clássica" deve ter conexão somente entre camadas adjacentes.

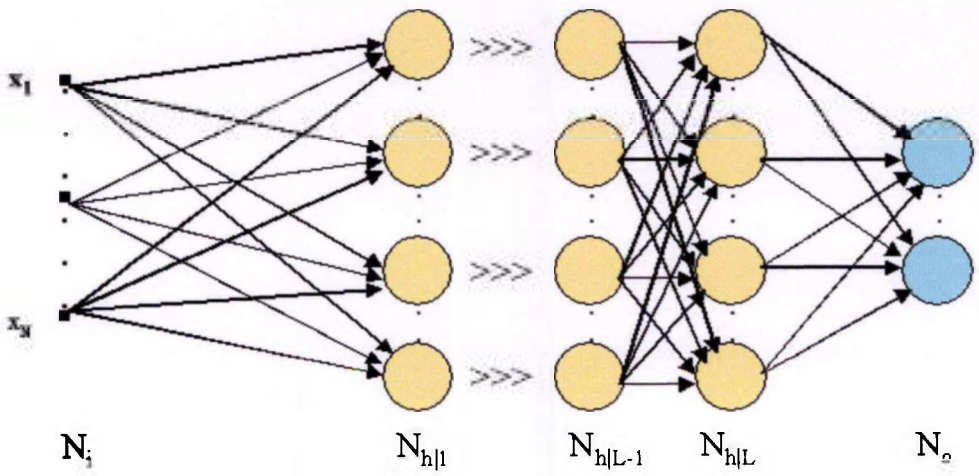


FIGURA 3.14 - Uma rede neural multicamada com L camadas intermediárias de unidades neurais.

Nessa rede neural multicamada é suposto que cada unidade neural se conecta com unidades de camadas adjacentes somente. É suposto também que a ativação de cada unidade é uma função diferenciável da entrada total, a saber,

$$y_k^p = g\left(s_k^p\right), \tag{3.14}$$

onde y_k representa a componente k da saída $\boldsymbol{y}$ de uma camada, ou ainda, o índice k representa uma unidade neural da camada em questão. Por sua vez

$$s_k^p = \sum_{j=0}^{C_j} w_{jk} y_j^p. \tag{3.15}$$

Para proceder à correta generalização da regra delta utilizando o mesmo método do gradiente descendente, deve-se, à semelhança de (3.9):

$$\Delta_p w_{jk} = -\eta \frac{\partial E^p}{\partial w_{jk}}. \tag{3.16}$$

A medida de erro E^p é definida como o erro quadrático total para o padrão p nas unidades de saída:

$$E^p = \frac{1}{2}\sum_{o=1}^{N_o}\left(d_o^p - y_o^p\right)^2, \tag{3.17}$$

onde d_o^p é a saída desejada para a unidade neural o quando é apresentado o padrão p. Desenvolvendo então (3.16):

$$\frac{\partial E^p}{\partial w_{jk}} = \frac{\partial E^p}{\partial s_k^p}\frac{\partial s_k^p}{\partial w_{jk}}. \tag{3.18}$$

O segundo fator, pela equação (3.15), fica

$$\frac{\partial s_k^p}{\partial w_{jk}} = y_j^p, \tag{3.19}$$

e o primeiro fator definimos como

$$\delta_k^p = -\frac{\partial E^p}{\partial s_k^p}. \tag{3.20}$$

Dessa maneira obtém-se um formato de atualização semelhante à regra delta (3.13),

$$\Delta_p w_{jk} = \eta\, \delta_k^p y_j^p. \tag{3.21}$$

Agora vai ser preciso desenvolver δ_k^p para cada unidade neural k. O resultado será uma computação recursiva dos δ's que pode ser implementado por propagação de sinais de erro que retorna através da rede.

Para computar δ_k^p aplica-se a regra da cadeia à essa derivada parcial, que se transforma num produto de dois fatores

$$\delta_k^p = -\frac{\partial E^p}{\partial s_k^p} = -\frac{\partial E^p}{\partial y_k^p}\frac{\partial y_k^p}{\partial s_k^p}, \tag{3.22}$$

o primeiro fator reflete a mudança de erro em função da saída da unidade e o segundo fator reflete a mudança da saída em função da entrada nesta unidade. Pela equação (3.14) o segundo fator é simplesmente a derivada da função de compressão g para a k-ésima unidade, avaliada para a entrada s_k^p.

$$\frac{\partial y_k^p}{\partial s_k^p} = g'\left(s_k^p\right). \tag{3.23}$$

Para computar o primeiro fator da equação (3.22), é preciso considerar dois casos. Primeiro, é considerado que a unidade k é uma unidade da camada de saída ($k = o$). Neste caso, pela definição de E^p,

$$\frac{\partial E^p}{\partial y_o^p} = -\left(d_o^p - y_o^p\right), \tag{3.24}$$

que é o mesmo resultado obtido pela regra delta padrão. Substituindo na (3.22), obtemos

$$\delta_o^p = \left(d_0^p - y_o^p\right) g_o'\left(s_o^p\right), \tag{3.25}$$

para qualquer unidade de saída o. Segundo, quando k não é uma unidade de saída, mas unidade de camada oculta $k=h$, será preciso aplicar a regra da cadeia para a medida de erro em função das variáveis de entrada, desde das camadas ocultas até a da camada de saída,

$$E^p = E^p\left(s_1^p, s_2^p, \ldots, s_j^p, \ldots\right), \tag{3.26}$$

$$\frac{\partial E^p}{\partial y_h^p} = \sum_{o=1}^{N_o} \frac{\partial E^p}{\partial s_o^p}\frac{\partial s_o^p}{\partial y_h^p} = \sum_{o=1}^{N_o} \frac{\partial E^p}{\partial s_o^p}\frac{\partial}{\partial y_h^p}\sum_{j=1}^{N_h} w_{ko} y_j^p = \sum_{o=1}^{N_o} \frac{\partial E^p}{\partial s_o^p} w_{ho} = -\sum_{o=1}^{N_o} \delta_o^p w_{ho}. \tag{3.27}$$

Substituindo este último resultado na equação (3.22), obtemos finalmente

$$\delta_h^p = g'\left(s_h^p\right)\sum_{o=1}^{N_o}\delta_o^p w_{ho}. \tag{3.28}$$

As equações (3.25) e (3.28) fornecem um procedimento recursivo para computar os δs para todas as unidades na rede neural, que por sua vez são usados para computar as atualizações de acordo com a equação (3.21). Este é o procedimento que constitui a regra delta generalizada para uma rede neural multicamada de unidades não-lineares.

Em resumo, quando um padrão é apresentado, os valores de ativação são propagados até as unidades de saída, e os valores de saída da rede são comparados com os valores desejados, resultando num erro para cada unidade de saída. Este erro deve ser levado a zero. Para tal os pesos devem ser mudados. Então o erro de uma unidade de saída é distribuído para as unidades ocultas com as quais tem conexão, multiplicado pelo peso da correspondente conexão (ponderação) de saída ($\sum_o \delta_o^p w_{ho}$) e pelo valor da derivada da função de ativação da unidade oculta avaliada na entrada integrada s_h^p.

3.1.5 Deficiências da retropropagação

O procedimento de aprendizado requer que a atualização do peso seja proporcional a $\frac{\partial E^p}{\partial w}$ com a constante de proporcionalidade η que é a taxa de aprendizado. A aplicação do gradiente descendente rigorosamente requer que passos infinitesimais sejam tomados. Para propósito prático, a taxa de aprendizado deve ser a maior possível até o limite que não haja oscilações, para que se vá suavemente até o ponto de mínimo da função energia ou de erro. Uma maneira de se evitar oscilações para η alto é que a atualização do peso seja dependente da atualização anterior através da adição de um termo de momento:

$$\Delta w_{jk}(t+1) = \eta\delta_k^p y_j^p + \alpha\Delta w_{jk}(t), \tag{3.29}$$

onde t representa a iteração e α é uma constante que determina o efeito da atualização anterior.

O papel do termo de momento é mostrado na FIGURA 3.15. Sem ele, leva-se muito tempo para o mínimo ser alcançado, no caso de η pequeno, ou então nunca o mínimo pode ser atingido devido às oscilações, para η grande. Com ele, o mínimo é alcançado rapidamente.

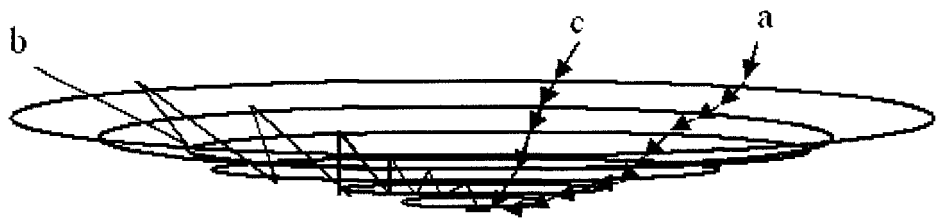


FIGURA 3.15 - A descida para a energia mínima. a) para uma pequena taxa de aprendizado; b) para uma grande taxa de aprendizado, e c) com uma taxa de aprendizado grande, mas adicionado o termo de momento.

Uma rede neural pode ser ter seus pesos atualizados após a apresentação de todas as entradas ou padrões de treinamento (batelada). Ou pode ter seus pesos atualizados após cada apresentação da entrada ou padrão (passo-a-passo): p é apresentado, E^p é calculado e os pesos são adaptados. Existe indicação empírica que o procedimento passo-a-passo converge mais rápido.

Fora os problemas que podem ocorrer com a falta de otimização da taxa de aprendizado e do momento, outros dois problemas cruciais no treinamento podem surgir: paralisia e mínimo local.

- Paralisia. Conforme a rede é treinada, os pesos podem ser ajustados para valores muito grandes e conseqüentemente a entrada total de uma unidade oculta ou de saída também fica com valor alto, positivo ou negativo, e com o uso de uma função de ativação sigmóide a unidade vai ter uma ativação próxima a zero ou próxima a um. Ao computar o erro da equação (3.28), para $g(s)=y$ sigmóide, $1/1+e^{-s}$, é necessária a sua derivada $g'(s)$ que é $g(s)(1-g(s))$. Com isso, para ativações próximas a zero ou a um o ajuste será próximo a zero, e dará a impressão que o processo de treinamento congelou.
- Mínimo local. A superfície de erro de uma rede neural complexa é cheia de montanhas e vales. Por causa do método do gradiente descendente, a rede pode estancar num mínimo local ao invés de ir a um mínimo mais profundo que provavelmente existe. Métodos probabilísticos podem evitar isso, mas o processo se torna lento. Outra possibilidade é aumentar o número de unidades das camadas intermediárias, mas até um certo limite, a partir do qual volta a ocorrer o problema do mínimo local.

Existe outra relação de compromisso com o número de unidades ocultas ou intermediárias. Um baixo número leva a uma discrepância com uma possível função desejada, porém um alto número de unidades faz com que a função obtida se encaixe

perfeitamente com os valores de entrada, reproduzindo inclusive os ruídos, é o supertreinamento, quando o que interessaria é uma função suave (FIGURA 3.16).

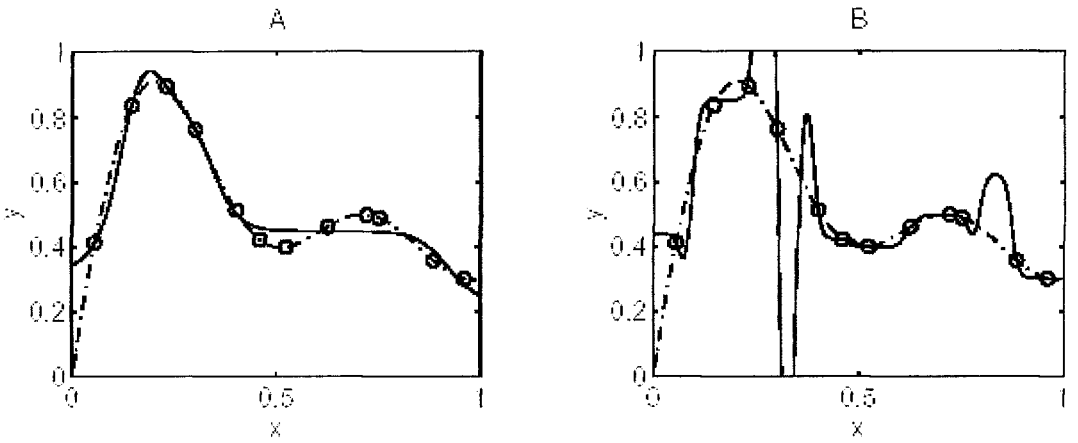


FIGURA 3.16 - Efeito do número de unidades ocultas na performance da rede neural. A curva pontilhada é a função desejada enquanto que a linha cheia denota a aproximação obtida pela rede neural. 12 amostras foram usadas. A) Com cinco unidades ocultas. B) Com 20 unidades ocultas, o supertreinamento.

Este exemplo demonstra que o aumento do número de unidades ocultas leva a um pequeno erro de treinamento $E_{aprendizado}$, mas não necessariamente leva a um pequeno erro de teste E_{teste}, isso após um certo número de unidades acrescidas. Este é o chamado efeito de pico. Um esboço das médias das taxas de erro, para treinamento e teste, em função do número de unidades ocultas é mostrado na FIGURA 3.17.

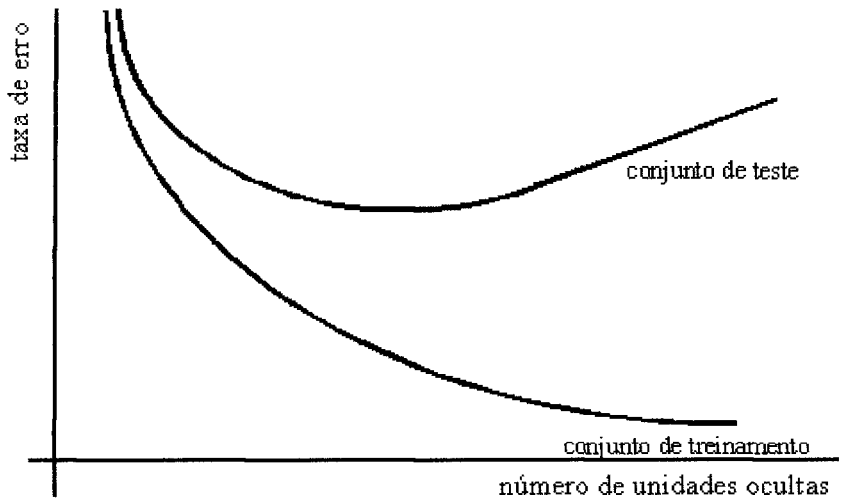


FIGURA 3.17 - As médias da taxa de erro de aprendizado e da taxa de erro de teste em função do número de unidades ocultas.

3.2 Mapas auto-organizáveis

Diferentemente do treinamento supervisionado, no treinamento não supervisionado não há um valor desejado ou devido fornecido pelo treinador ou usuário. Agora os pesos serão ajustados pela similaridade com as entradas. O mapa auto-organizável é um treinamento não-supervisionando, realizado com o intuito de extrair características de um conjunto de dados de entrada ou agrupá-los (saída discreta). No mapa auto-organizável ocorre o aprendizado competitivo, no qual as unidades, além de possuírem conexões com as entradas - conexões frontais, possuem conexões entre elas próprias - conexões laterais, como representado na FIGURA 3.18. As conexões laterais podem ser inibitórias ou excitatórias, como na FIGURA 3.19.

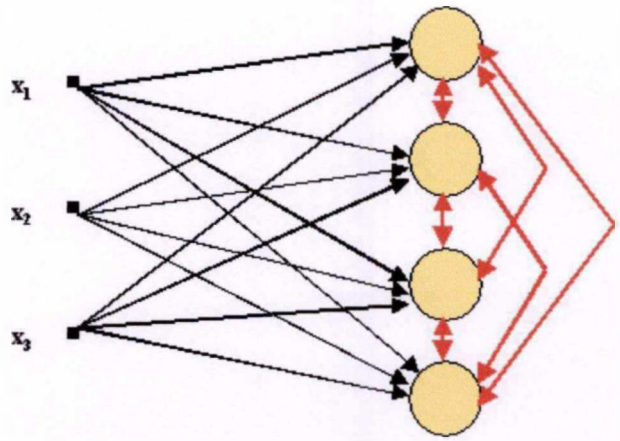


FIGURA 3.18 - Diagrama arquitetural de uma rede simples de aprendizado competitivo, com conexões frontais a partir da entrada (setas pretas), excitatórias, e conexões laterais inibitórias ou excitatórias entre os neurônios de saída (setas vermelhas), com intensidade de acordo com a proximidade do vencedor.

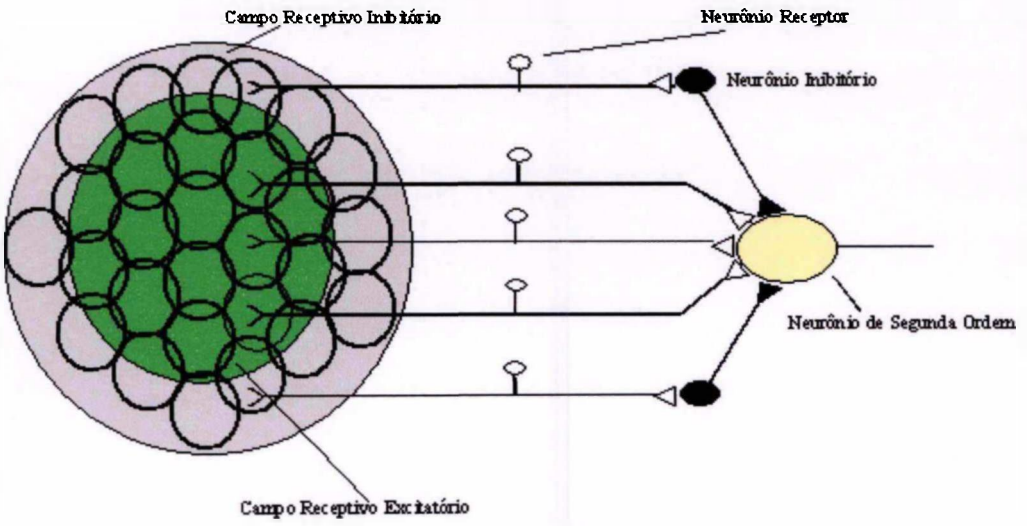


FIGURA 3.19 - Estrutura de campos receptivos.

3.2.1 Considerações gerais

Os mapas auto-organizáveis (SOM, na sigla em inglês) são uma classe especial de redes neurais artificiais (RNA) baseadas no aprendizado competitivo; os neurônios da camada de saída da rede competem uns com os outros para serem ativados (ou "queimados"), resultando que somente um neurônio de saída, ou um neurônio por grupo, fica ativo para uma entrada ou estímulo. A idéia original do aprendizado competitivo – o vencedor leva tudo ou neurônio vencedor – foi proposta por Rosenblatt (1958), mas seu modelo mais geral foi desenvolvido por Kohonen (1982).

O desenvolvimento dos mapas auto-organizáveis foi motivado por uma característica distintiva do cérebro dos mamíferos, seu cérebro está organizado em várias regiões de tal maneira que diferentes entradas sensórias ficam representadas por mapas físicos computacionais ordenados topologicamente. Exemplos dessas regiões são os córtices táctil, olfatório (Nadel *et al.*, 1989), visual e auditivo. A informação derivada dessa maneira pode ser prontamente acessada por outros processos de ordem mais elevada usando esquemas simples de conexão (Haykin, 1999).

Num mapa auto-organizável, os neurônios ou unidades neurais ficam localizados nos nós de um reticulado ou rede usualmente uni ou bidimensional. Dimensões mais altas são possíveis, mas não comuns devido à dificuldade de visualização ou de processamento do ordenamento topológico. Os neurônios ficam seletivamente sintonizados para vários padrões da entrada ou classes de padrões de entrada durante o curso do processo de aprendizado competitivo. As localizações dos neurônios assim sintonizados, ou seja, os vencedores, se tornam ordenados em relação a uns com os outros, de tal maneira que um sistema de coordenadas com significado é criado na rede para diferentes padrões de entrada (Kohonen, 1990). Um mapa auto-organizável é, portanto, caracterizado pela formação de um mapa topográfico dos padrões de entrada em que as localizações espaciais (coordenadas) dos neurônios com suas sintonias na rede são indicativas das características estatísticas intrínsecas contidas nos padrões de entrada.

O interesse aqui reside em construir mapas topográficos artificiais que aprendem através da auto-organização. Nesse contexto, um ponto importante é o princípio da formação do mapa topográfico, de inspiração neurobiológica, vindo a ser que "a localização espacial de um neurônio de saída em um mapa topográfico corresponde a um particular domínio ou característica dos dados obtidos do espaço de entrada" (Kohonen, 1990). Este princípio forneceu dois modelos de mapas de características: o modelo de Willshaw - von der Marlsburg (1976) e o de Kohonen (1982). O primeiro, com duas

camadas de neurônios (entrada e saída), utiliza para as pós-sinapses um mecanismo excitatório de curto alcance e um mecanismo inibitório de longo alcance, com um limite de ativação e um máximo para a soma dos pesos sinápticos para cada neurônio pós-sináptico. Este modelo, inspirado no mapeamento retinotópico para o córtex visual, não é interessante por não realizar compressão de dados e redução de dimensionalidade sobre a entrada. Já o modelo de Kohonen (FIGURA 3.20), pertencente à classe dos algoritmos de codificação vetorial, fornece um mapeamento topológico que otimizadamente relaciona um número fixo de vetores (palavras-código), associados a pontos num espaço discreto de saída uni ou bidimensional, para um espaço de entrada contínuo de maior dimensão, dessa maneira facilitando a compressão dos dados. A palavra-código corresponde no SOM aos pesos sinápticos de um neurônio e possui dimensão igual à da entrada.

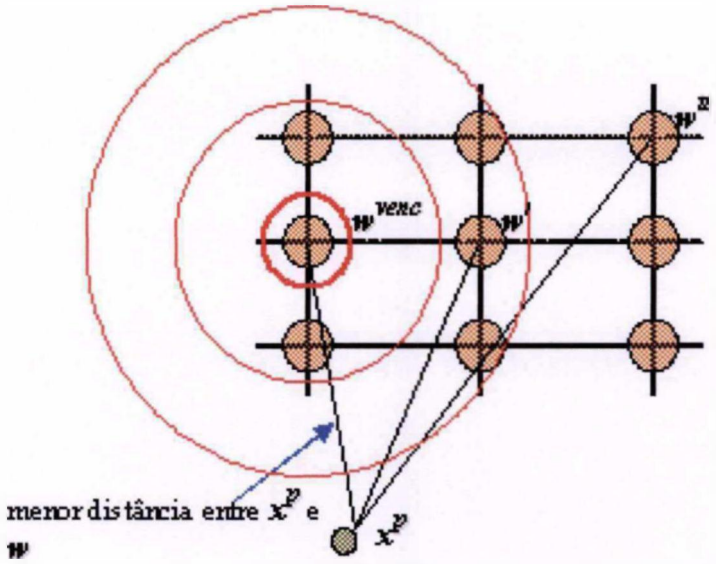


FIGURA 3.20 - Mapa auto-organizável (SOM) com grade bidimensional.

3.2.2 Algoritmo do mapa auto-organizável

O algoritmo responsável pela formação do mapa auto-organizável tem como primeiro passo a inicialização dos pesos sinápticos da rede. São atribuídos a eles valores pequenos extraídos de um gerador de números aleatórios, dessa maneira, nenhum ordenamento é previamente estabelecido. Uma vez que a rede tenha sido propriamente inicializada, há três processos essenciais envolvidos na formação do mapa auto-organizável, que são competição, cooperação e adaptação.

1. Competição. Para cada padrão de entrada, as unidades neurais computam seus respectivos valores de uma função discriminante $g(s) = g(\mathbb{w}^T \mathbb{x})$. Essa função discriminante fornece a base da competição entre os neurônios. O neurônio da rede que tiver o maior valor da função discriminante é declarado o vencedor da competição.

2. Cooperação. O neurônio vencedor determina a localização espacial de uma vizinhança de neurônios excitados fornecendo a base da cooperação entre tais vizinhos.

3. Adaptação Sináptica. Esse mecanismo habilita os neurônios excitados a incrementarem seus valores individuais da função discriminante em relação ao padrão de entrada mediante ajustes adequados nos seus pesos sinápticos. O ajuste feito é tal que a resposta do neurônio vencedor para a aplicação seguinte de um padrão similar é aumentada.

3.2.2.1 Competição

Seja m a dimensão do espaço de entrada (dados). Seja um padrão de entrada (vetor) selecionado aleatoriamente do espaço de entrada denotado por

$$\mathbb{x} = [x_1, x_2, x_3, \ldots, x_m]^T. \tag{3.30}$$

O vetor peso sináptico de cada neurônio da rede tem a mesma dimensão que o espaço de entrada. Seja o vetor peso denotado por

$$\mathbb{w}_j = [w_{j1}, w_{j2}, w_{j3}, \ldots, w_{jm}]^T, \quad j = 1, 2, \ldots, l, \tag{3.31}$$

onde l é o número total de unidades neurais na rede. Para encontrar o melhor casamento (ou a melhor sintonia) entre o vetor de entrada $\mathbb{x}$ e os vetores peso $\mathbb{w}_j$, compara-se o produto interno $\mathbb{w}_j^T \mathbb{x}$ para $j = 1, 2, \ldots, l$ e seleciona-se o maior. Considera-se o mesmo limite de ativação para todos os neurônios, que pode ser considerado zero. Portanto, ao selecionar o neurônio com o maior produto interno $\mathbb{w}_j^T \mathbb{x}$, fica determinado onde a vizinhança topológica dos neurônios excitados está centrada.

O critério da melhor sintonia, baseada na maximização do produto interno $\mathbb{w}_j^T \mathbb{x}$, é matematicamente equivalente a minimizar a distância euclidiana entre os vetores $\mathbb{x}$ e $\mathbb{w}_j$. Utilizando-se o índice $i(\mathbb{x})$ para identificar o neurônio que melhor se parceiriza com o vetor de entrada $\mathbb{x}$, pode-se determinar $i(\mathbb{x})$ aplicando a condição

$$i(\mathbb{x}) = \arg\min_j \| \mathbb{x} - \mathbb{w}_j \|, \quad j = 1, 2, \ldots, l, \tag{3.32}$$

o que sumariza a essência do processo de competição entre os neurônios, a saber, um espaço de entrada contínuo de padrões de ativação é mapeado para um espaço discreto de saída de neurônios por um processo de competição entre os neurônios na rede (FIGURA 3.21).

Dependendo da aplicação em interesse, a resposta da rede pode ser tanto o índice do neurônio vencedor, mais especificamente, sua posição na rede, ou o vetor peso sináptico que está mais próximo topologicamente do vetor de entrada.

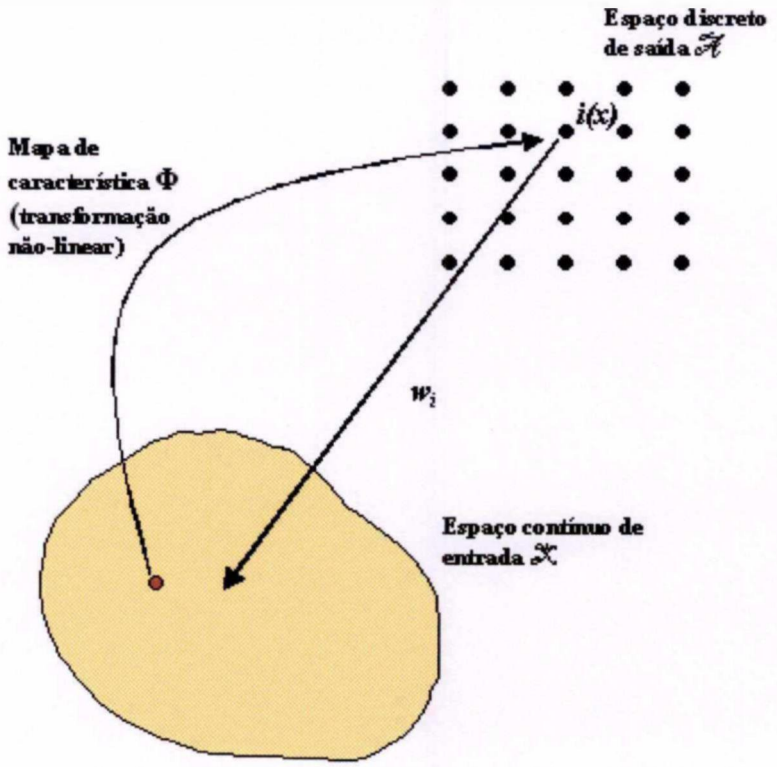


FIGURA 3.21 - Relacionamento entre o mapa de característica Φ e o vetor peso w_i *do neurônio vencedor* i.

3.2.2.2 Cooperação

O neurônio vencedor se localiza no centro de uma vizinhança topológica de neurônios cooperativos. A definição de como é essa vizinhança topológica é inspirada pela evidência neurobiológica da interação lateral entre um grupo de neurônios excitados. Um neurônio que é ativado tende a excitar os neurônios em sua vizinhança imediata mais que outros longe dele. Tal comportamento leva a definir uma vizinhança topológica em torno

da unidade neural vencedora i, que decai suavemente com a distância lateral (Ritter *et al.*, 1992). Especificamente, seja $h_{j,i}$ a vizinhança topológica centrada no neurônio vencedor i, encerrando um grupo de neurônios excitados (cooperativos), um deles indicado por j. A distância lateral entre o neurônio vencedor i e o neurônio excitado j será representado por $d_{i,j}$. Ao considerar a vizinhança topológica $h_{j,i}$ uma função unimodal da distância lateral $d_{j,i}$, dois requisitos distintos serão satisfeitos:

1) A vizinhança topológica $h_{j,i}$ é simétrica em relação ao máximo em $d_{i,j}$=0; em outras palavras, ela atinge seu valor máximo sobre o neurônio vencedor i para o qual a distância $d_{i,j}$ é zero.

2) A amplitude da vizinhança topológica $h_{j,i}$ decresce monotonicamente com a distância lateral $d_{j,i}$, caindo a zero para $d_{j,i} \rightarrow \infty$; que é uma condição necessária para a convergência.

A escolha mais típica de $h_{j,i}$ que satisfaz esses requisitos é a função gaussiana

$$h_{j,i(\mathbb{x})} = \exp(- d_{j,i}^{\,2} / 2\sigma^2), \tag{3.33}$$

que é invariante sob translação, isto é, independe da localização do neurônio vencedor. O parâmetro σ é a "largura efetiva" da vizinhança topológica (FIGURA 3.22) que mede o grau em que os neurônios excitados nas cercanias do neurônio vencedor participarão no processo de aprendizado. Em termos qualitativos, a vizinhança topológica gaussiana é mais apropriada biologicamente que a retangular, além de fazer o algoritmo do SOM convergir mais rapidamente (Lo *et al.*, 1991, 1993).

A distância lateral $d_{j,i}$ é definida por

$$d_{j,i}^{\,2} = ||\mathbb{r}_j - \mathbb{r}_i||^2, \tag{3.34}$$

onde o vetor discreto $\mathbb{r}_j$ define a posição do neurônio excitado j e $\mathbb{r}_i$ define a posição discreta da unidade vencedora i, ambos medidos no espaço de saída discreto.

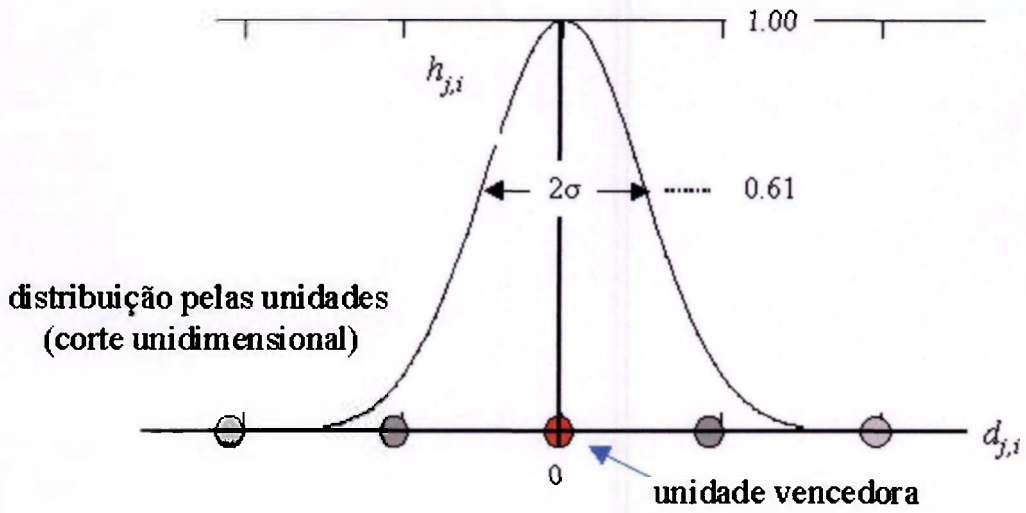


FIGURA 3.22 - Função de vizinhança gaussiana.

Outra característica única do algoritmo do SOM é o encolhimento da vizinhança topológica com o tempo. Este requisito é satisfeito fazendo a largura σ da função de vizinhança topológica $h_{j,i}$ decrescer com o tempo. Aqui "tempo" significa o número de passagens ou iterações (n) do conjunto de vetores de entrada para o aprendizado. A escolha mais popular para a dependência de σ com o tempo discreto n é o decaimento exponencial, usado neste trabalho, ou a função inversa (Baptista, 2002)

$$\sigma(n) = \sigma_0 \exp (-n/\tau_1), \qquad n = 0, 1, 2, 3, ..., \qquad (3.35a)$$

$$\text{ou} \quad \sigma(n) = \sigma_0 /(1+ b_1 n), \qquad n = 0, 1, 2, 3, ..., \qquad (3.35b)$$

onde σ_0 é o valor de inicialização do algoritmo SOM, e τ_1, $1/b_1$ são constantes de tempo. Uma boa escolha para τ_1 é número total de iterações/ln(σ_0) (FIGURA 3.23).

O argumento da exponencial da gaussiana pode ter um fator de ajuste C_R fazendo com que a ativação dos neurônios da vizinhança topológica seja mais forte ou mais fraca durante todo o decorrer do decaimento. Com isso a função de vizinhança topológica fica com a seguinte forma dependente do tempo

$$h_{j,i(x)}(n) = \exp [- C_R (d_{j,i}^2 / 2\sigma^2(n))]. \qquad (3.36)$$

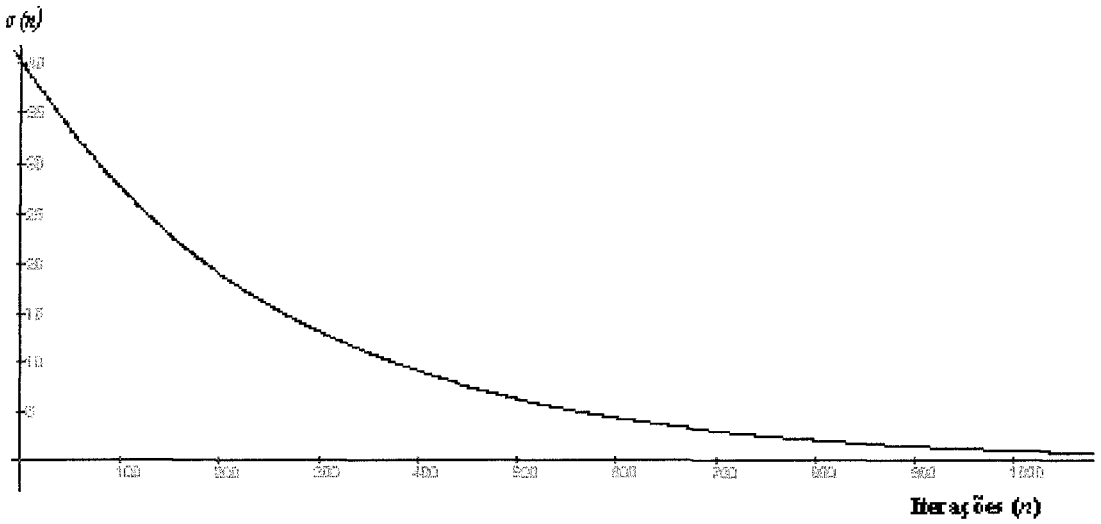


FIGURA 3.23 - Decaimento exponencial do parâmetro σ(n) *da função de vizinhança.*

O propósito da função de vizinhança pode ser visto como a correlação das direções nas atualizações dos vetores peso de um grande número de neurônios excitados na rede (Luttrell, 1989). Com a dependência do tempo temos a renormalização da função de vizinhança para trabalhar com número cada vez menor de graus de liberdade normalizados. Isso garante uma adaptação sináptica suave para a rede neural.

Uma outra opção de função de vizinhança que faz uso de comunicação lateral excitatória (fortemente para região próxima, fracamente para longe) e inibitória (fracamente para intermediária) é o "chapéu mexicano" unimodal (FIGURA 3.24).

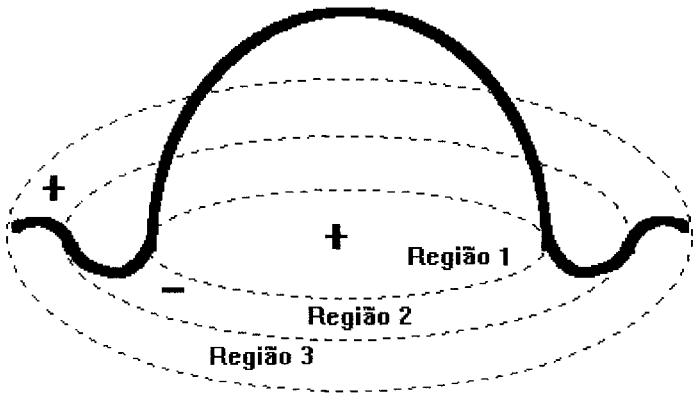


FIGURA 3.24 - Função vizinhança "chapéu mexicano".

Simulações com o "chapéu mexicano" não mostraram eficiência maior do que com a gaussiana de vizinhança.

3.2.2.3 Adaptação

A adaptação sináptica é o último processo na formação auto-organizada de um mapa de características. Para a rede se auto-organizar, o vetor peso sináptico $\mathbb{w}_j$ do neurônio j na rede precisa se modificar em relação ao vetor de entrada $\mathbb{x}$. A questão reside em como fazer essa modificação. No postulado de Hebb do aprendizado, um peso sináptico aumenta com uma ocorrência simultânea de atividades pré e pós-sináptica. O uso de tal regra cabe bem para um aprendizado associativo. Para o tipo de aprendizado não-supervisionado considerado aqui, a hipótese de Hebb na sua forma básica é insatisfatória porque as mudanças nas conexões ocorrem em uma só direção, o que no final levaria todos os pesos sinápticos para a saturação. Para superar este problema, a hipótese de Hebb é modificada para incluir um termo de esquecimento $e(y_j)\mathbb{w}_j$, onde $\mathbb{w}_j$ é o vetor peso sináptico do neurônio j e $e(y_j)$ é uma função escalar positiva da resposta y_j. O único requisito imposto a $e(y_j)$ é que na expansão em série de Taylor o seu termo constante seja zero

$$e(y_j) = 0 \quad para \quad y_j = 0. \tag{3.37}$$

De posse dessa função, a mudança no vetor peso do neurônio j na rede pode ser expressa como

$$\Delta \mathbb{w}_j = \eta\, y_j\, \mathbb{x} - e(y_j)\mathbb{w}_j, \tag{3.38}$$

onde η é o parâmetro taxa de aprendizado do algoritmo. O primeiro termo do membro direito da equação (3.38) é o hebbiano e o segundo termo é o de esquecimento. Para satisfazer o requisito da equação (3.37), é escolhida uma função linear para $e(y_j)$ como a seguinte

$$e(y_j) = \eta\, y_j. \tag{3.39}$$

Escolhendo a resposta y_j como

$$y_j = h_{j,i(\mathbb{x})}, \tag{3.40}$$

e usando as equações (3.39) e (3.40) em (3.38), obtém-se

$$\Delta \mathbb{w}_j = \eta\, h_{j,i(\mathbb{x})}\, (\mathbb{x} - \mathbb{w}_j). \tag{3.41}$$

Finalmente, utilizando o formalismo de tempo discreto, dado o vetor peso sináptico $\mathbb{w}_j(n)$ do neurônio j no tempo n, o vetor peso atualizado $\mathbb{w}_j(n + 1)$ no tempo $n + 1$ é definido por (Kohonen, 1982):

$$\mathbb{w}_j(n + 1) = \mathbb{w}_j(n) + \eta(n)\, h_{j,i(\mathbb{x})}(n)\, (\mathbb{x} - \mathbb{w}_j(n)), \tag{3.42}$$

que é aplicado a todos os neurônios da rede que estão dentro da vizinhança topológica do neurônio vencedor i. A equação (3.42) produz o efeito de mover o vetor peso sináptico $\mathbb{w}_i$ do neurônio vencedor i em direção ao vetor de entrada $\mathbb{x}$. Com as repetidas apresentações dos dados de treinamento, os vetores pesos sinápticos tendem a seguir a distribuição dos vetores de entrada devido à atualização da vizinhança. O algoritmo, portanto, leva a um ordenamento topológico do mapa de características para o espaço de entrada uma vez que neurônios adjacentes na rede tenderão a ter pesos sinápticos similares.

É preciso notar que para obter a equação de atualização (3.42) dos vetores pesos sinápticos, são necessárias a heurística da equação (3.36) ao selecionar a função de vizinhança $h_{j,i(\mathbb{x})}(n)$ e a heurística de selecionar qual o parâmetro taxa de aprendizado $\eta(n)$.

Esta dedução pode ser também obtida pelo método do "gradiente descendente" (Kovács, 1998) que, coloquialmente, pode ser exposto como, "do alto de uma montanha, a trilha mais rápida e descendente para se chegar ao ponto mais baixo de uma montanha". A função sobre a qual se fará o gradiente é a chamada "função de energia", dada pela comparação, quadrática, entre o resultado devido (d) e o resultado obtido pelo algoritmo de aprendizado numa determinada iteração dos dados

$$E(w) = \tfrac{1}{2} \sum_k \sum_l [y_k^d - y_l]^2 . \tag{3.43}$$

No caso do algoritmo do SOM, tendo a regra do vizinho-mais-próximo, a saída devida é a própria entrada $\boldsymbol{x}$ que deve ser comparada com o peso $\mathbb{w}_j$. Além disso, cada comparação tem a sua taxa de contribuição dada por uma função densidade de probabilidade (pdf), cabendo bem a função de vizinhança gaussiana $h_{j,i(\mathbb{x})}$.

$$E(w) = \tfrac{1}{2} \sum_k \sum_l h_{l,k(\boldsymbol{x})}[x_k^d - w_l]^2 . \tag{3.44}$$

Pelo método do gradiente descendente, a atualização (Δ) de $\mathbb{w}_j$ será proporcional ao negativo do componente do gradiente (∇) de $E(\mathbb{w})$

$$\Delta w = -\eta \frac{\partial E(w)}{\partial w}, \tag{3.45}$$

levando a

$$\Delta w_j = \eta\, h_{j,i(x)}(x - w_j). \tag{3.46}$$

O parâmetro taxa de aprendizado $\eta(n)$ varia com o tempo discreto como acontece numa abordagem estocástica, devendo começar com um valor inicial η_0 e decrescer gradualmente com o aumento do tempo n. Este requisito pode ser satisfeito por um decaimento exponencial, usado neste trabalho, ou por função inversa (Baptista, 2002),

$$\eta(n) = \eta_0 \exp(-n/\tau_2), \qquad n = 0, 1, 2, 3, ..., \tag{3.47a}$$

$$\text{ou} \quad \eta(n) = h_{j,i(x)}\,[\eta_0/(1 + b_2 n)], \qquad n = 0, 1, 2, 3, ..., \tag{3.47b}$$

onde τ_2, $1/b_2$ são constantes de tempo diferentes das equações 3.35a e 3.35b. Uma boa escolha de τ_2 é: total de iterações / (η_0 * 10). Por exemplo, se são escolhidas 1000 apresentações do conjunto de dados de treinamento, e se a taxa de aprendizado inicial é 0,2, então $\tau_2 = 500$.

De uma completa desordem, para se atingir gradualmente a representação organizada dos padrões de ativação, o processo adaptativo necessita possuir duas fases que são ordenamento e convergência explicitados a seguir.

1) Fase de ordenamento ou de auto-organização. É durante esta fase do processo adaptativo que a ordem topológica dos vetores peso é realizada. O número de iterações (apresentações do conjunto de dados de treinamento) deve ser da ordem de 1000 e a taxa de aprendizado $\eta(n)$ deve iniciar com um valor da ordem de 0,1 e não ficar mais baixa que 0,01. As simulações mostraram eficiência para valores iniciais da taxa de aprendizado entre 0,2 e 0,4. A função de vizinhança topológica $h_{j,i(x)}(n)$ deve inicialmente incluir todos os neurônios da rede centralizados no neurônio vencedor i, e encolher vagarosamente para somente os neurônios vizinhos de i (FIGURA 3.25), ou para somente i, conforme foi discutido sobre os parâmetros da equação (3.33).

2) Fase de convergência. Esta segunda e última fase do processo adaptativo é um refinamento do mapa de características providenciando uma quantificação estatística mais acurada do espaço de entrada. Como regra geral o número de iterações para esta fase deve ser pelo menos 500 vezes o número de neurônios na rede (Haykin, 1999), podendo atingir dezenas de milhares de iterações. Nesta fase a taxa de aprendizado $\eta(n)$ deve se manter constante e baixa, da ordem de 0,01. Em qualquer evento ou fase, a taxa de aprendizado

não pode chegar a zero, senão é possível que a rede neural fique "emperrada" num estado metaestável, um defeito topológico de mínimos locais. A função de vizinhança $h_{j,i(x)}$ nesta fase deve conter somente os neurônios mais próximos, podendo se reduzir para somente o vencedor. Esta fase, por exigir muito recurso de CPU devido ao número de iterações, foi omitida ou muito reduzida quando somente interessava a classificação do neurônio pelo vetor de entrada mais próximo.

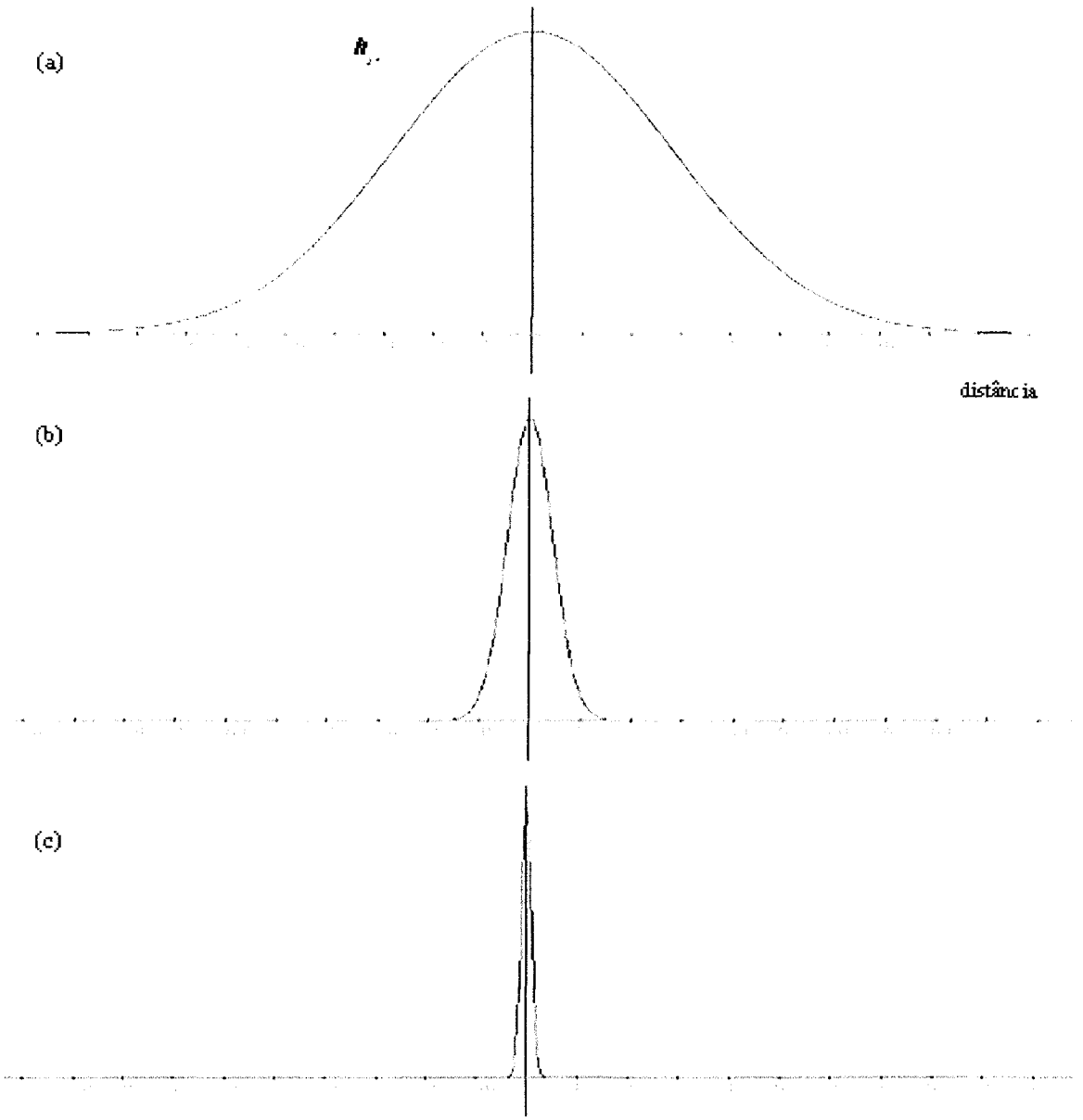


FIGURA 3.25 - Perfil inicial (a) da função de vizinhança unimodal $h_{j,i}(x)(n)$ gaussiana. Com 500 iterações (b) do conjunto de entrada X (n=500). Após 950 iterações.

A FIGURA 3.26 apresenta um exemplo de convergência de um mapeamento para a topologia original das entradas.

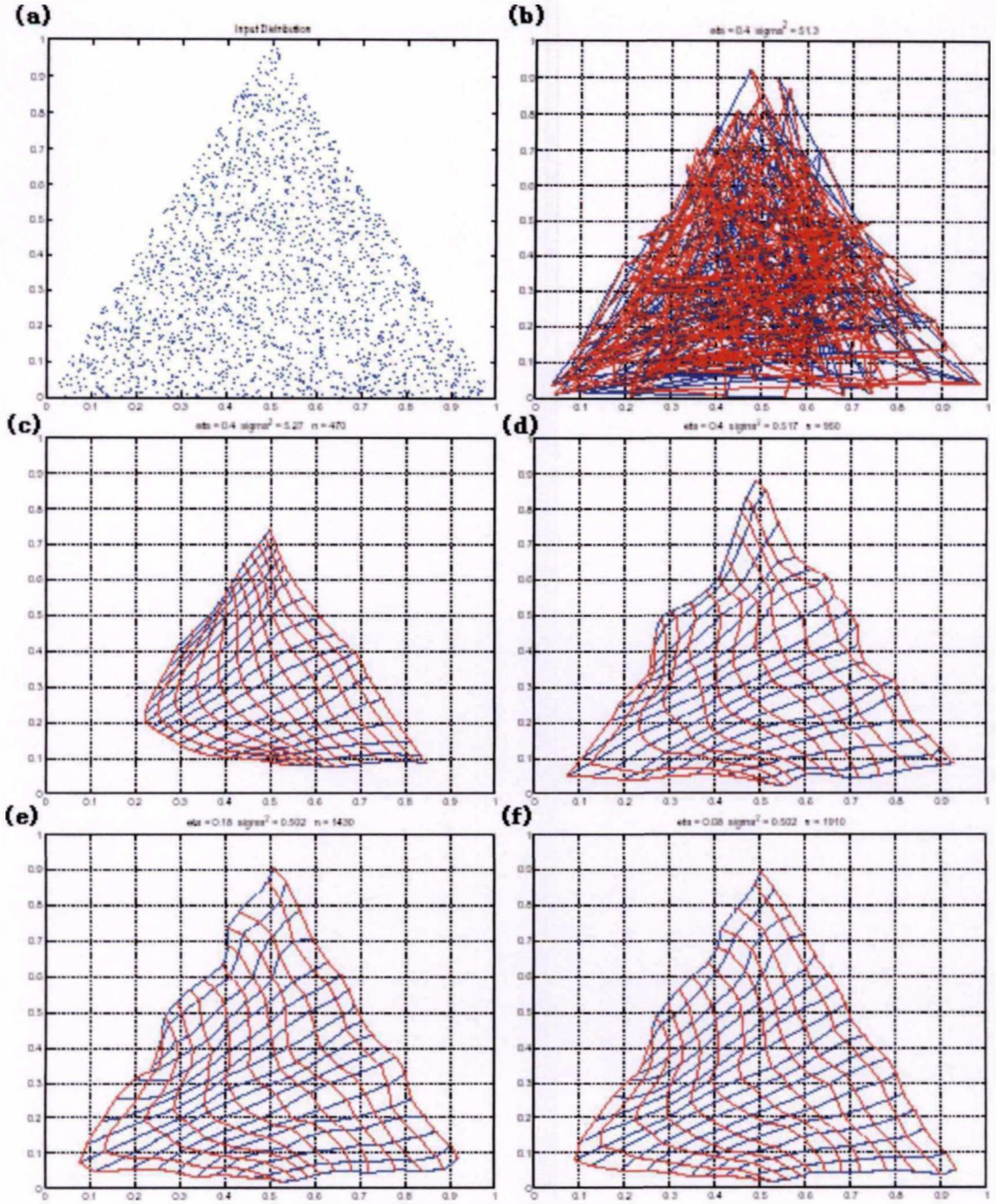


FIGURA 3.26 - A convergência de um mapeamento para a conservação de topologia. No gráfico (a) está a distribuição dos vetores de entrada. Nos gráficos seguintes, são plotadas as coordenadas dos vetores pesos de uma rede bidimensional retangular plana 12X18 unidades. Uma linha cheia em cada figura conecta um peso de uma unidade neural com os pesos das unidades neurais vizinhas. Em (b) estão plotadas as coordenadas dos pesos iniciais (geralmente escolha aleatória). Em (f) o final da ordenação após 1910 iterações. Extraído de Papliński, 2002, onde estes gráficos foram realizados com a ajuda da ferramenta (tool) sofm.m para o MATLAB.

3.2.3 Propriedades do mapa de características

Uma vez que o algoritmo do SOM convergiu, o mapa de características computado pelo algoritmo exibirá características importantes do espaço de entrada. Para descrevê-las, considere X um espaço de entrada (dados) contínuo, a topologia de tal é definida pelo relacionamento métrico dos vetores $\boldsymbol{x} \in X$. Seja A um espaço de saída discreto, cuja topologia é definida por um arranjo de unidades neurais localizadas sobre os nós de um reticulado ou rede. Seja $\boldsymbol{\Phi}$ uma transformação não-linear chamada mapa de características, que mapeia o espaço de entrada X sobre o espaço de saída A

$$\boldsymbol{\Phi}: X \rightarrow A. \tag{3.48}$$

A equação (3.48) pode ser vista como uma abstração da equação (3.32) que define o local de um neurônio vencedor $i(\boldsymbol{x})$ obtido como resposta a um vetor de entrada $\boldsymbol{x}$. Por exemplo, em termos neurobiológicos, o espaço de entrada X pode representar todas as coordenadas dos receptores somato-sensórios distribuídas densamente sobre o corpo inteiro. Em correspondência, o espaço de saída A representa todos os neurônios localizados naquela camada do córtex cerebral em que os receptores somato-sensórios estão confinados.

Propriedade 1 - Aproximação do Espaço de Entrada. "O mapa de características $\boldsymbol{\Phi}$, representado por um conjunto vetores peso sináptico $\{ w_j \}$ no espaço de saída A, é uma boa aproximação do espaço de entrada X."

A meta básica do algoritmo SOM é guardar um grande número de vetores de entrada $\boldsymbol{x} \in X$ num pequeno conjunto de protótipos $w_j \in A$, de tal maneira que realiza uma boa aproximação do espaço de entrada original X. A base teórica desta idéia tem raízes na teoria de quantização vetorial, que leva a resultados como a redução de dimensionalidade ou a compressão de dados (Gersho et Gray, 1992).

Propriedade 2 - Ordenamento Topológico. "O mapa de características $\boldsymbol{\Phi}$ computado pelo algoritmo SOM é topologicamente ordenado no aspecto de que a localização espacial de um neurônio na rede corresponde a um particular domínio ou característica dos padrões de entrada."

A propriedade de ordenamento topológico é uma conseqüência direta da equação de atualização (eq. (3.42)) que força o vetor peso sináptico $\mathbb{w}_i$ do neurônio vencedor $i(\boldsymbol{x})$ a se movimentar em direção ao vetor de entrada $\boldsymbol{x}$. Isso também afeta o movimento dos vetores peso sináptico $\mathbb{w}_j$ dos neurônios mais próximos j ao redor do neurônio vencedor $i(\boldsymbol{x})$. Dessa maneira pode-se visualizar o mapa de características $\boldsymbol{\Phi}$ como uma rede elástica com topologia de rede uni ou bidimensional como prescrita no espaço de saída A, e cujos nós possuem pesos como coordenadas no espaço de entrada X (Ritter, 1995). A meta geral do algoritmo pode ser declarada como: "Aproximar o espaço de entrada X por apontadores ou protótipos na forma de vetores peso sináptico $\mathbb{w}_i$, de tal maneira que o espaço de características $\boldsymbol{\Phi}$ fornece uma representação fiel das propriedades que caracterizam os vetores de entrada $\mathbf{x} \in X$ em termos de um critério claro."

Propriedade 3 - Proporcionalidade entre Densidades. "O mapa de características $\boldsymbol{\Phi}$ reflete as variações na estatística da distribuição de entrada: regiões do espaço de entrada X do qual vetores de amostra $\mathbf{x}$ são extraídos, vindos portanto com alta probabilidade de ocorrência, são mapeados em domínios mais amplos do espaço de entrada A, e portanto com melhor resolução que regiões em X, em que há vetores $\mathbf{x}$ com baixa probabilidade de ocorrência."

Propriedade 4 - Seleção de Características. "Fornecidos dados de um espaço de entrada com uma distribuição não-linear, o mapa auto-organizável é apto a selecionar um grupo das principais características por aproximar a distribuição subjacente."

Esta propriedade é a culminação das propriedades 1 a 3 e traz à tona a idéia da análise de componentes principais (*principal component analysis*, PCA). Mas com uma importante diferença. Na FIGURA 27a está exibida uma distribuição bidimensional de dados com média zero resultante de um mapeamento de processo entrada-saída linear corrompida por ruído. Nesta situação, a análise de componentes principais trabalha bem: fornece-nos a melhor descrição da distribuição "linear" por meio de uma linha reta, ou melhor, de um hiperplano unidimensional que passa pela origem e corre paralelo ao autovetor associado com o maior autovalor da matriz correlação dos dados. Porém na situação da FIGURA 27b, que é resultado do mapeamento de um processo entrada-saída não-linear corrompido por ruído de média zero, é impossível uma aproximação por linha reta computada por PCA de tal maneira que se torne aceitável como representação dos dados. Por outro lado, o SOM, construído sobre uma rede unidimensional, é apto a lidar com esta situação em virtude da sua propriedade de ordenamento topológico, como ilustrado na FIGURA 27b.

Em termos precisos, o mapa de características auto-organizável fornece uma aproximação discreta das curvas principais ou superfícies principais (no exemplo da FIGURA 27b, hipercurva unidimensional), e portando o SOM pode ser visto como uma generalização não-linear da análise de componentes principais.

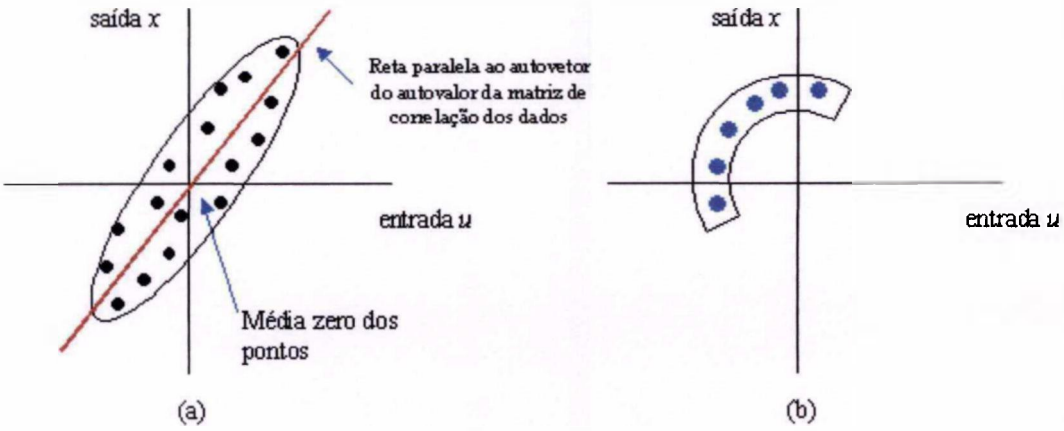


FIGURA 3.27 - (a) Distribuição bidimensional produzida por um mapeamento de processo linear entrada-saída com ruído. (b) Distribuição bidimensional produzida por um mapeamento de processo não-linear entrada-saída com ruído (pontos omitidos dentro do contorno em arco). Os círculos azuis representam os componentes dos vetores pesos w_j.

4 METODOLOGIA

4.1 Desenvolvimento da ferramenta de *software* para o SOM

4.1.1 Descrição da ferramenta

A plataforma de desenvolvimento escolhida foi a Java, por ser 100% orientada a objetos, por sua versatilidade, pela escalabilidade e pela gratuidade de suas licenças (java.sun.com), além de rodar em qualquer plataforma que tenha o JVM (*Java Virtual Machine*) instalado. O critério de arquitetura para o desenvolvimento foi para utilização na Web. Especificamente, trata-se de páginas HTML (*Hypertext Mark up Language*) com formulários de envio/retorno (FIGURA 4.1) e um *applet* (FIGURA 4.2). O *applet* (*application* + *pellet*) é um tipo de aplicação própria para páginas Web que se caracteriza por utilizar recursos da máquina cliente, não sobrecarregando o servidor, porém seguro contra executáveis e vírus. Na necessidade de uma aplicação *on-demand*, sem dependência da máquina do cliente, o código do *applet* pode ser transportado para um servlet (*server* + *pellet*), controlador de páginas JSP (*Java Server Pages*).

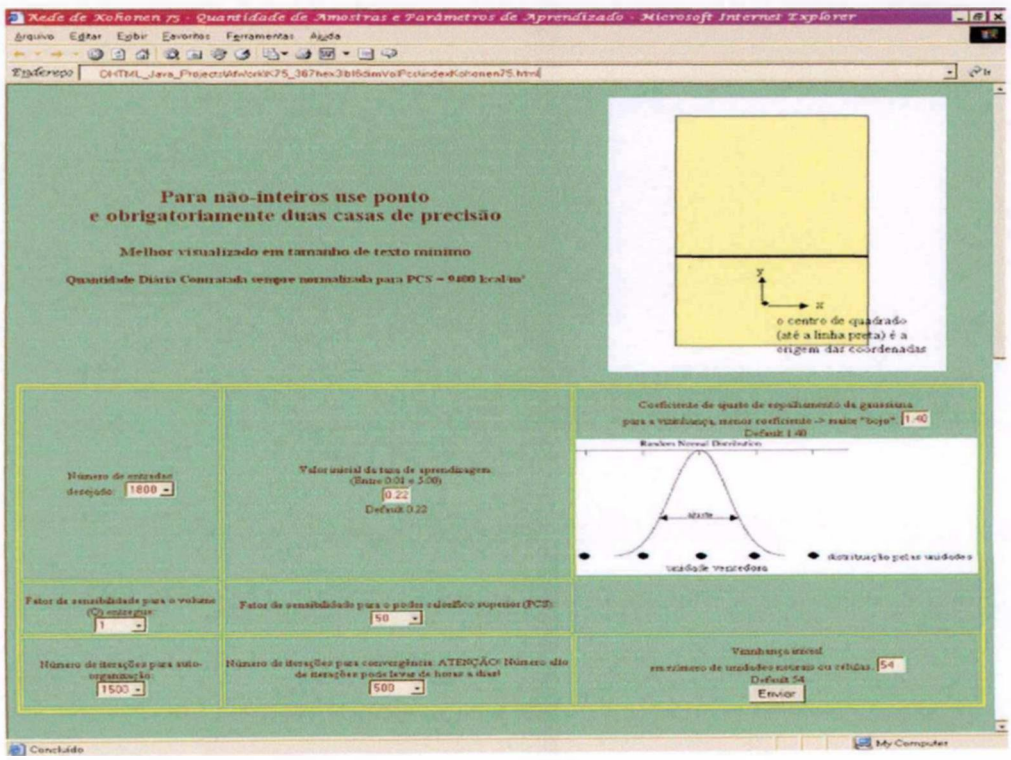


FIGURA 4.1 - Página Web em HTML com formulário para preenchimento de parâmetros para o SOM.

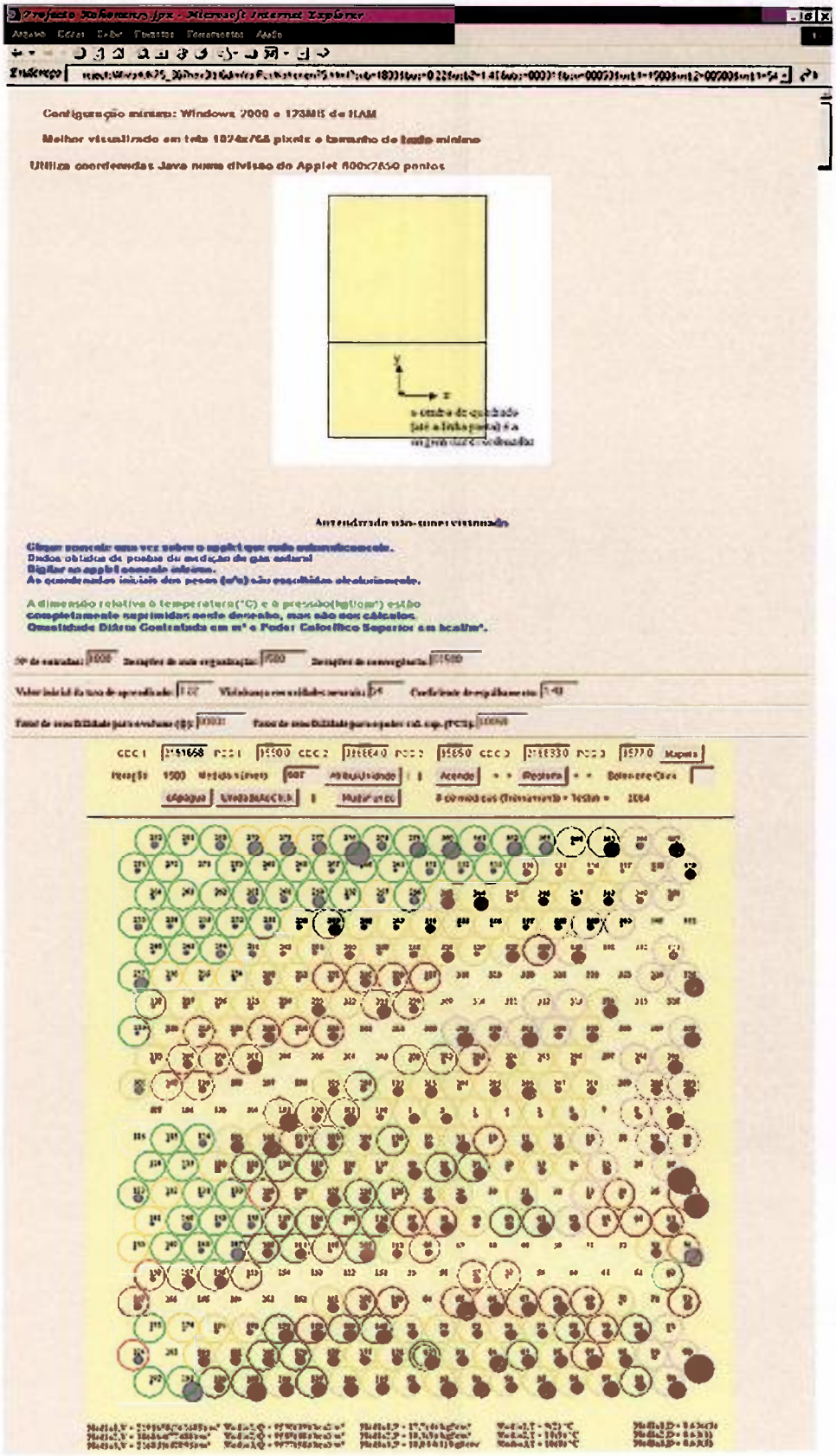


FIGURA 4.2 – Página Web contendo o applet *do SOM (em amarelo). Está exibida a parte superior e central do* applet*, que possui grande comprimento, para as diversas visões necessárias ao usuário.*

O design do *applet* para esta adaptação do SOM possui na sua parte mais inferior (FIGURA 4.3) a plotagem dos vetores dados de entrada, símbolo "x + número seqüencial da entrada". Esta plotagem considera até quatro dimensões (x, y, a e b) do vetor de entrada – as mais relevantes para o negócio – sendo as coordenadas x, y no plano do *applet* e, sobre estas coordenadas, é construído o desenho de uma elipse com parâmetros a e b - semi-eixo maior e semi-eixo menor. Por exemplo, no caso de transporte de gás natural, como se verá em detalhes mais adiante, a coordenada x será o *label* (índice de distância geográfica) do ponto de medição, a coordenada y será o volume entregue deste ponto de medição, o semi-eixo maior, o poder calorífico e o semi-eixo menor, a densidade relativa. Os pesos, simbolizados por "@ + número da unidade", seguem estas mesmas coordenadas. Os pesos iniciais, gerados aleatoriamente, são plotados utilizando-se o símbolo "O + número da unidade", permanecendo durante toda a simulação.

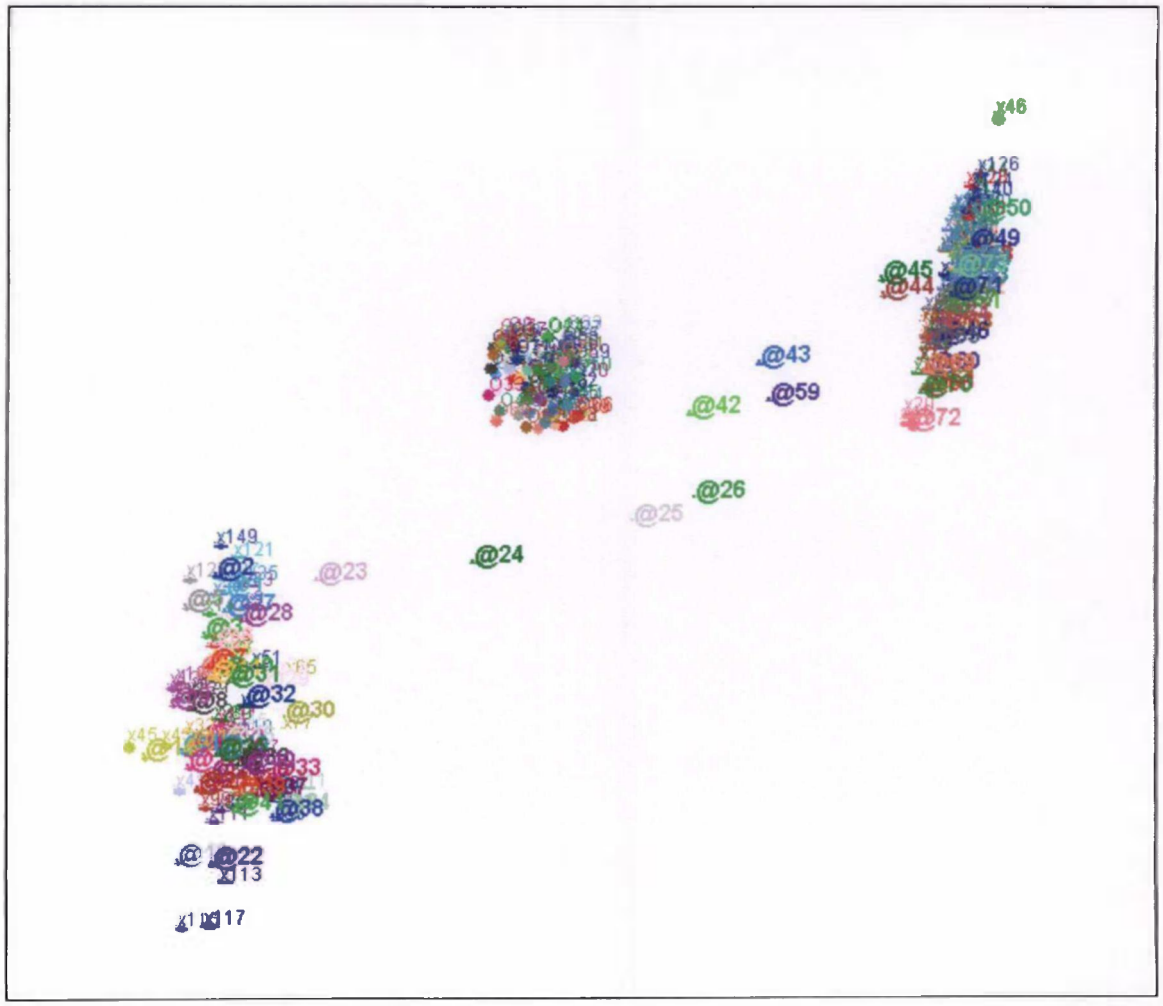


FIGURA 4.3 - A parte mais inferior do applet, *que realiza a plotagem das entradas (x) e dos pesos (@). Os pesos iniciais também ficam registrados (O). A cor de fundo (background) é escolhida pelo usuário.*

Na parte da tela imediatamente acima (FIGURA 4.4) é exibida uma relação com as coordenadas dos pesos, atualizadas dinamicamente. Na FIGURA 4.4, os valores das coordenadas estão normalizados entre –35 e +35, facilitando a visualização pelo usuário de suas posições relativas. Logo ao lado das coordenadas, é impressa a proporção do número de entradas atribuída à unidade neural correspondente, ou seja, para quantas entradas a unidade é vencedora.

Seguindo para cima no *applet*, aparecem as linhas de média e desvio padrão de cada componente do vetor de entrada para as amostras de treinamento (FIGURA 4.5), excetuando a componente *label*, que será representada por um índice indicando o ponto ou local de medição. Para uma quantidade muito grande de componentes, como no caso de transientes de reatores nucleares, somente a média e o desvio padrão dos cinco primeiros componentes são impressos. O valor do desvio padrão é colocado entre parênteses e se refere ao último algarismo da média.

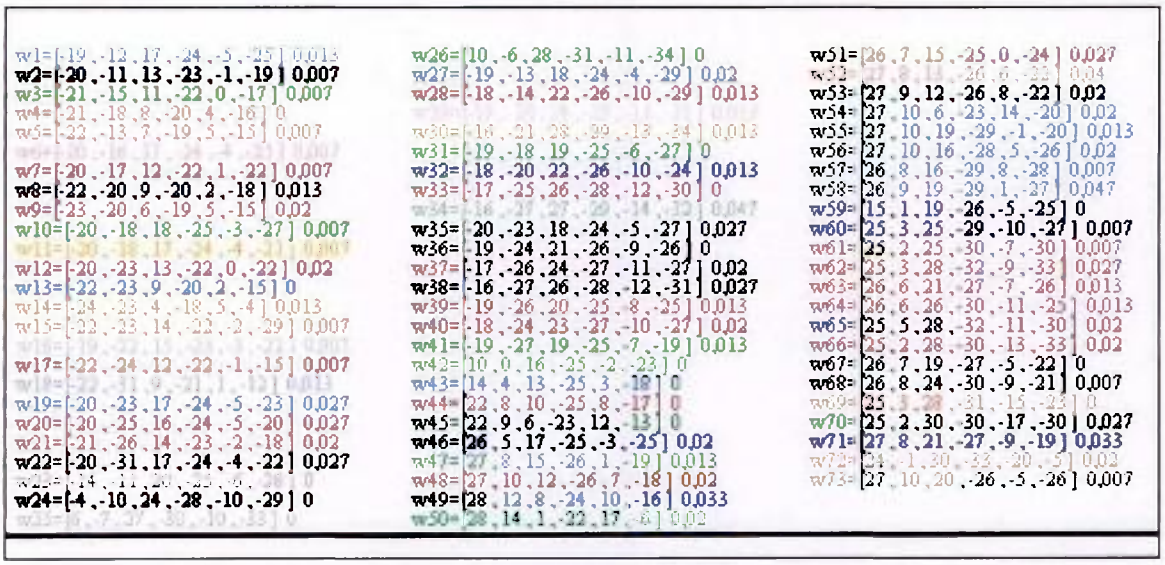


FIGURA 4.4 - Coordenadas normalizadas de cada peso da rede neural. Ao lado de cada conjunto de coordenada, há a proporção do número de entradas atribuído à unidade neural.

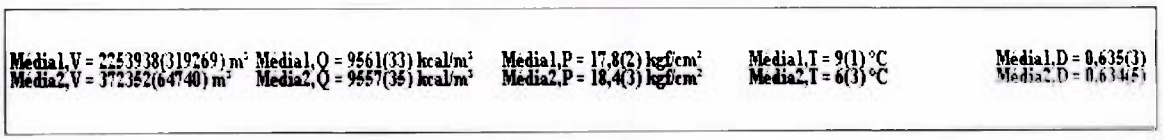


FIGURA 4.5 - Impressão no applet *das médias e desvios padrão dos componentes do vetor entrada, para o conjunto de entradas de treinamento. Os índices 1 e 2 referem-se a diferentes locais de medição.*

A seguir vem a topologia da rede neural. A FIGURA 4.6 mostra o resultado final do processamento. Cada unidade neural está representada por um círculo vazio. Os círculos cheios no centro de cada unidade dão a noção de quantidade de entradas atribuída, uma alternativa visual às proporções mencionadas na FIGURA 4.4. As cores dos círculos vazios correspondem à classificação da unidade baseado na classificação da entrada que está mais próxima do peso desta unidade. A classificação da entrada pode ser feita *a priori* como no caso de transientes, ou seja, um vetor vem indicado com um *label* conforme a operação for estacionária, *step*, degrau, ou anomalia. Ou a classificação da entrada pode ser feita *a posteriori* como no caso da logística, em que a classificação é baseada nas distâncias às médias de alguns componentes, ou, no lugar das médias, são inseridos valores de contrato ou ainda valores referenciais de segurança. Estes valores são inseridos, para este sistema de *software*, em campos mencionados na FIGURA 4.7.

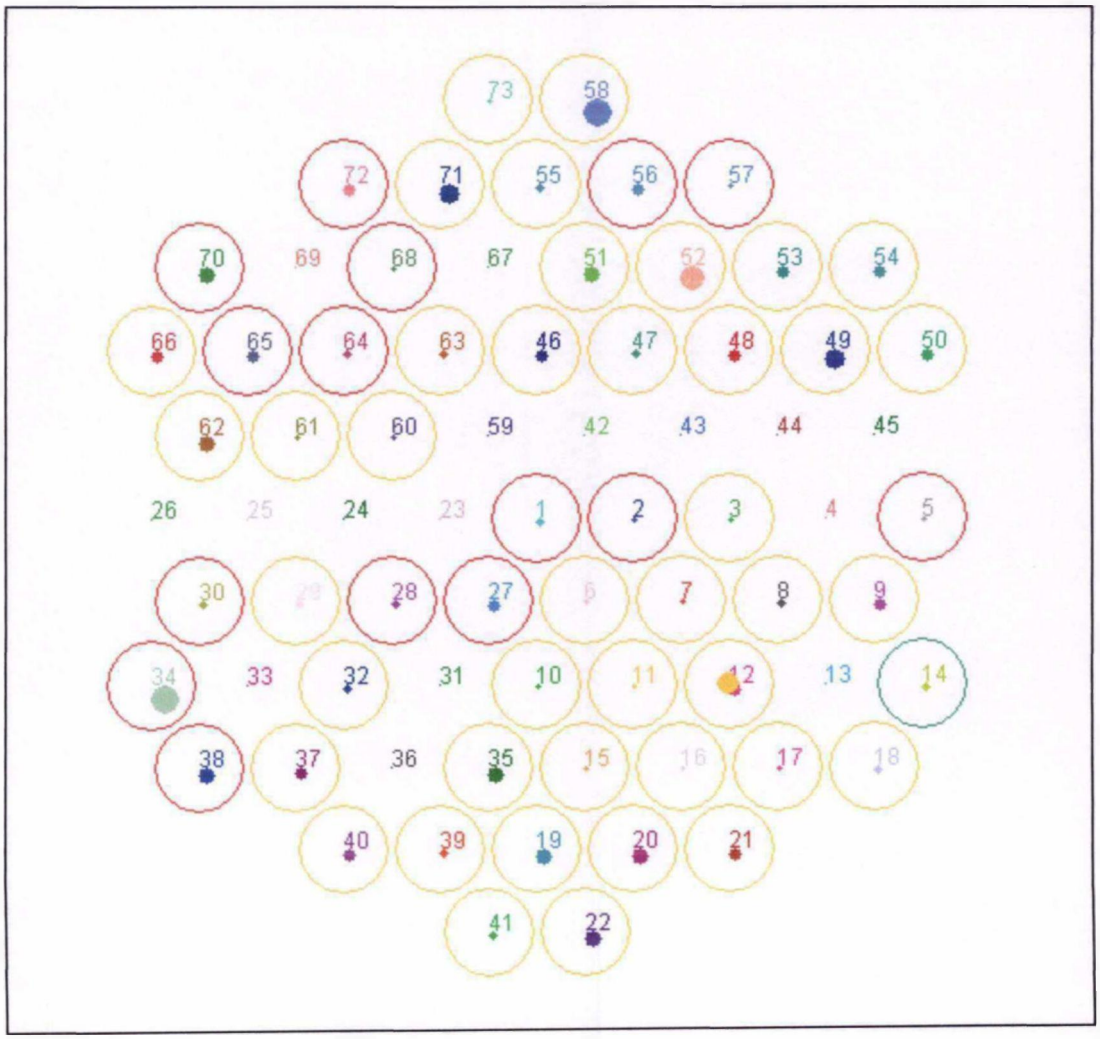


FIGURA 4.6- Exemplo de resultado de processamento de um mapa auto-organizável para camada de saída com topologia bidimensional plana hexagonal. As cores dos círculos vazios correspondem à classificação.

Círculo branco significa que a unidade não foi atribuída a nenhuma entrada. Círculo vazio verde significa qualidade acima das expectativas ou operação boa. Círculo vazio amarelo é classificação para qualidade normal ou operação regular. Círculo vermelho é para qualidade abaixo de patamares exigidos ou operação irregular. "Qualidade" aqui se refere à logística e significa qualidade de entrega e qualidade do produto. No exemplo da FIGURA 4.6, o círculo cheio amarelo sobre a unidade 12 é piscante, significando que uma entrada de teste foi atribuída a esta unidade neural, no caso, a entrada número 151, que está indicada na FIGURA 4.7.

A aplicação *applet* possui alguns botões. Entre os principais está o botão "Mapeia" que classifica cada unidade tomando como referência os valores inseridos nos campos antecedentes. Por *default*, ao acionar o treinamento com um clique sobre o *applet*, são inseridas as médias dos componentes especificados no nome de cada campo. No exemplo da FIGURA 4.7, os componentes de referência para classificação são quantidade diária contratada (QDC) e poder calorífico superior (PCS), isso para cada ponto de medição (índices 1 e 2, correspondente a *label* de menor valor e *label* de maior valor respectivamente). O botão "AtribuiUnidade" atribui uma unidade neural ao número seqüencial de entrada, que pode ser tanto uma entrada de treinamento ou uma entrada de teste. A unidade atribuída fica então piscando. Esta seqüência de entradas é obtida de banco de dados corporativo mediante aplicativo auxiliar em Java, com o qual é possível inserir o intervalo de tempo e os pontos de medição desejáveis. Apertando mais uma vez o botão "AtribuiUnidade", a unidade pára de piscar. O botão "Acende" preenche os círculos vazios com a mesma cor do contorno, para melhor visualização da classificação. Apertando mais uma vez este botão os círculos voltam a ficarem vazios. O botão "Restarta" esvazia o *applet* dos dados e resultados, deixando pronto para um novo treinamento, utilizando os mesmos parâmetros do formulário da página inicial (FIGURA 4.1). O botão "MudaFundo" troca a cor de fundo do *applet*, a cada acionamento a cor é cíclica: branca, preta e amarela.

No início da página em que está o *applet*, são impressos os parâmetros escolhidos na página anterior de abertura, os principais são:

1) quantidade de entradas de treinamento;
2) valor inicial da taxa de aprendizado; e
3) coeficiente de espalhamento da função de vizinhança (FIGURA 4.7).

O denominador de decréscimo caiu em desuso em versões posteriores por uso dos denominadores de decréscimo τ_1 e τ_2 descritos na seção 3.2.2.3.

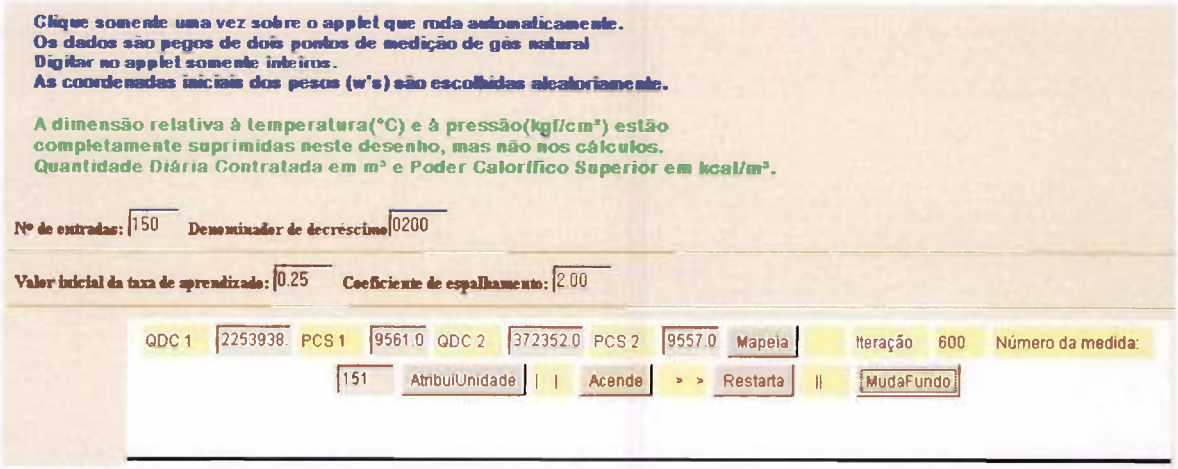


FIGURA 4.7 - Parte superior da página e do applet para o SOM.

Junto aos botões é exibido qual o número corrente da iteração ou época, ou seja, o número de quantas vezes se passou pelo conjunto de treinamento. O treinamento está terminado quando se atinge o número de iteração escolhido na página de início. Outra maneira de saber que o treinamento chegou ao fim é quando o *applet* adquire um suave piscar de freqüência constante.

4.1.2 Arquitetura do *software*

A arquitetura das aplicações Web (FIGURA 4.8) é uma variação da antiga arquitetura cliente-servidor, introduzindo-se mais uma camada. A camada cliente é constituída pelo navegador Web (*browser*), a camada intermediária (*middleware*) se subdivide entre a camada Web (camada na qual residem páginas Web que fazem solicitações via HTTP) e a camada de aplicação. Para o sistema deste trabalho, reside na camada de aplicação uma rotina Java de consulta que fornece um arquivo texto dos dados de entrada para treinamento e teste do mapa auto-organizável. Tal rotina pode ser colocada como serviço em um servidor de aplicações, rodando em intervalos programados. Como toda aplicação multicamadas, cada camada pode ser distribuída ou desdobrada (*deployment*) em servidores separados ou mesmo funcionar tudo num único servidor (FIGURA 4.9). Esta arquitetura multicamadas, juntamente com a plataforma Java, facilita a distribuição e utilização da ferramenta, não necessitando de instalação máquina a máquina do usuário e tornando-a virtualmente independente de sistema operacional e *hardware*.

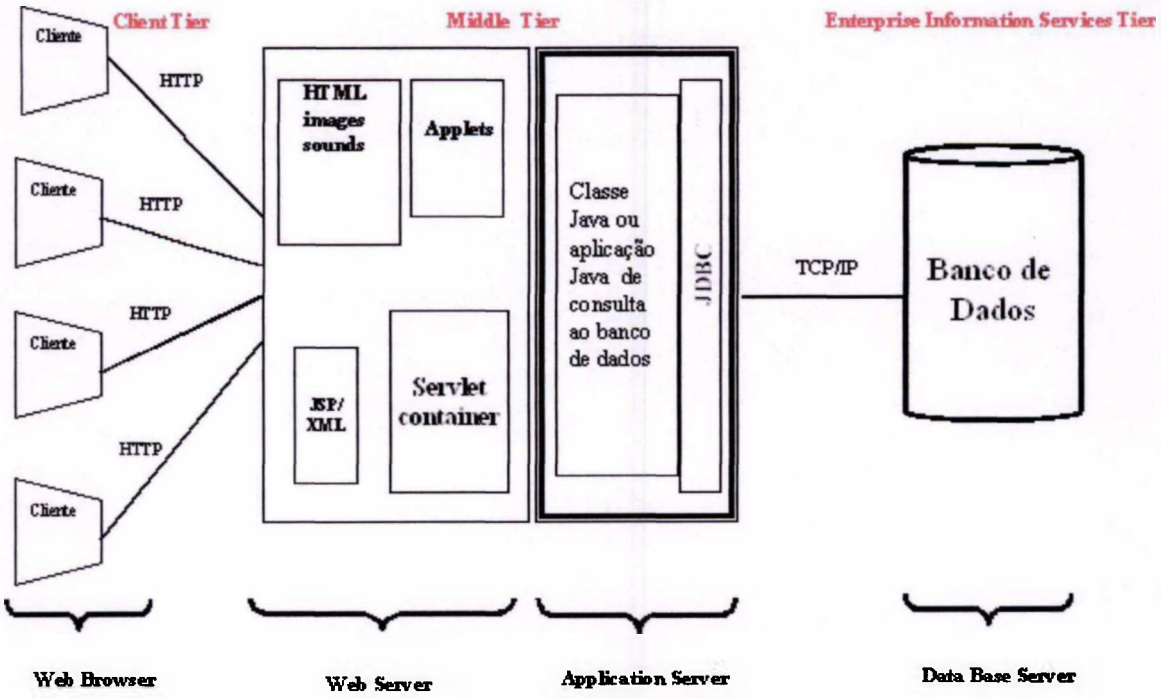


FIGURA 4.8 - Arquitetura do sistema.

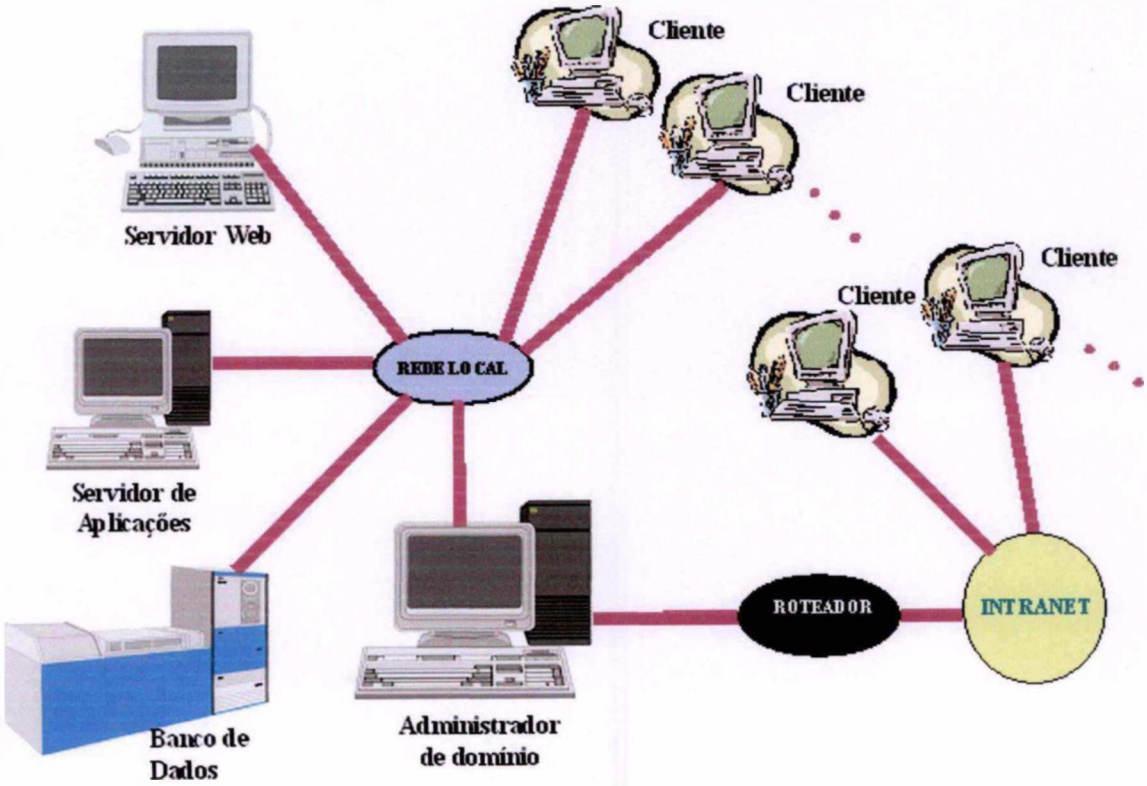


FIGURA 4.9 - Distribuição (deployment) de uma arquitetura de sistema multicamadas.

4.1.3 Recursos

O *hardware* para desenvolvimento do *software* foi um microcomputador com sistema operacional Microsoft Windows 2000, 512 MB de RAM e CPU da família x86 Intel com 1GHz de processamento.

O ambiente para desenvolvimento de aplicações Java 2 foi o J2SE SDK versão 1.4.1_06 (http://java.sun.com/j2se/1.4.1/download.html), que inclui o JRE, máquina virtual para executar aplicações Java.

O navegador (*browser*) utilizado foi preponderantemente o Internet Explorer versão 6.0, porém o Netscape 6 e o FireFox 1.0 também foram utilizados.

4.2 Procedimentos de aquisição e tratamento dos dados de entrada

4.2.1 Dados das condições de operação de reatores nucleares

4.2.1.1 Reator IRIS

IRIS (*International Reactor Innovative and Secure*) é um esforço de cooperação internacional para projetar um sistema de energia nuclear para atingir metas desafiadoras de segurança e de vantagens econômicas (Carelli *et al.*, 2001). Consiste em módulos de reator a água leve integrado com o circuito primário. Uma planta nuclear seria capaz de possuir até três módulos, cada um projetado para gerar 335 MWe. Detalhes técnicos mais apurados estão em Carelli *et al.*, 2000 e em Petrovic *et al.*, 2000. A sua configuração integral facilita a abordagem "segurança intrínseca ao projeto" (*safety-by-design*) que essencialmente é a eliminação física da ocorrência de alguns acidentes e a diminuição da probabilidade da ocorrência de outros acidentes, e caso ocorram, diminui as conseqüências.

A segurança intrínseca ao projeto e as barreiras de segurança recomendadas pelo princípio da defesa em profundidade podem vir a eliminar acidentes e atenuar muitas conseqüências de falhas mecânicas, elétricas e humanas, mas, para produzir uma planta realmente confiável, são necessárias ferramentas modernas que ofereçam informações relevantes com usabilidade para a equipe de operação.

O vaso integrado do IRIS possui uma geometria cilíndrica com altura de 21 m e diâmetro interno de 6,8 m, fechado na parte inferior por uma calota soldada e na parte superior por um flange removível em forma de hemisfério. Este vaso abriga o núcleo do reator, os mecanismos de controle, as barras de controle, as estruturas de suporte (para o

núcleo do reator e para as varetas de combustível e barras de controle), o pressurizador, oito geradores de vapor, blindagens internas, aquecedores e oito bombas para resfriamento do reator (FIGURA 4.10). Em operação, a temperatura e a pressão médias são de aproximadamente 305°C e 15,5 MPa respectivamente. Entre o flange e o núcleo, localizam-se as aberturas de entrada e saída do fluido refrigerante (água leve pressurizada). O vaso é construído em aço-carbono (AS 533 *grade* B *plate* ou AS 508 *class* 2 ou *class* 3), revestido com uma camada de aço inoxidável. A estrutura que sustenta o vaso é constituída de aço montada sob uma segunda estrutura de concreto.

O núcleo do reator se localiza na parte inferior do vaso. A sustentação interna do núcleo é constituída por duas partes fundamentais: suporte inferior (inclui a blindagem térmica) e o suporte para instrumentação *incore*. Conforme mostrado na FIGURA 4.10, existe uma abertura em forma de anel entre a parede interna do vaso e o núcleo do reator – o canal externo – por onde a água do circuito primário desce através de bombeamento, cedendo calor para os tubos helicoidais do circuito secundário. A água do circuito primário após esta troca possui uma temperatura de 290°C e, após passar pelo canal de descida, segue caminho subindo através do núcleo do reator, atingindo na parte superior temperatura de 328°C. A água é mantida a uma pressão entre 14,2 – 16,3 MPa.

Alguns parâmetros típicos do núcleo do reator do IRIS são:

- Núcleo: h = 4,3 m e d = 2,4 m; superfície de transferência de calor = 2992 m²;
- Combustível: 48,5 tU, UO_2;
- 89 a 264 varetas combustíveis numa grade 17x17 (4,95% U^{235} – duração ~48 meses);
- *Burn up*[6] ≈ 55 – 60 MW*day(thermal)*/kgU;
- Encamisamento: Zirlo[7], espessura = 0,57 mm;
- Vareta: d_{ext} = 9,5 mm;
- Veneno queimável: IFBA[8] e Er;
- Absorvedor: Ag-In-Cd; e
- Controlador químico H_3BO_3.

[6] Taxa de irradiação (ou "queima") do combustível nuclear, expressa em energia térmica produzida por unidade de massa inicial do combustível.
[7] ZIRLO™, liga metálica cuja referência é Zr – 1%Nb – 1%Sn – 0,1%Fe.
[8] IFBA, *Zirconium Diboride* (ZrB_2, enriquecido com ^{10}B) *Integral Fuel Burnable Absorber*.

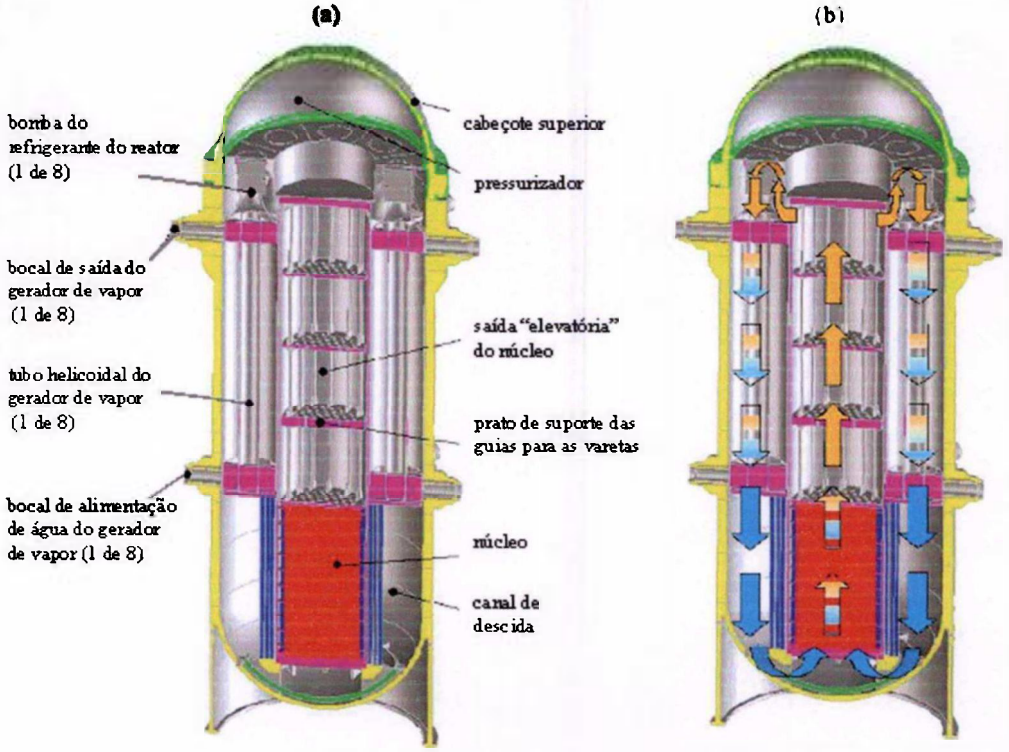


*FIGURA 4.10 - (a) Descrição do vaso integrado do IRIS. (b) Fluxo do refrigerante (*coolant*).*

Um reator nuclear está sujeito a diferentes tipos de transientes. Alguns deles serão aqui considerados:

a. Subida constante da potência a 5% da potência nominal por minuto;
b. Descida constante da potência a 5% da potência nominal por minuto;
c. Descida da potência em degraus de 10% da potência nominal;
d. Subida da potência em degraus de 10% da potência nominal;
e. Desarme da turbina;
f. SCRAM[9] ;
g. Abertura inadvertida da válvula de alívio;
h. Descida da potência em degrau negativo de 70% a partir da plena potência; e
i. Acidente de pequena perda de refrigerante (*small* LOCA[10]).

[9] *Safety Control Rod Axe Man*, termo advindo do Chicago Pile, primeiro reator nuclear do mundo, em suma é o desarme do reator.
[10] *Loss of Coolant Accident*.

A FIGURA 4.11 ilustra a resposta de temperatura durante um SCRAM a partir da plena potência. Junto com estes transientes, o conjunto de dados de treinamento contém muitos padrões de condições estacionárias, de 20 a 110% da potência nominal.

É importante observar que a resposta do reator para cada um destes transientes depende do sistema de controle. Quando a arquitetura de controle e os parâmetros não são otimizados, modelos simplificados são suficientes para prover dados básicos a fim de caracterizar o comportamento do IRIS. Em Barroso *et al.*, 2003, há a descrição das ferramentas simplificadas que produziram os dados utilizados neste trabalho.

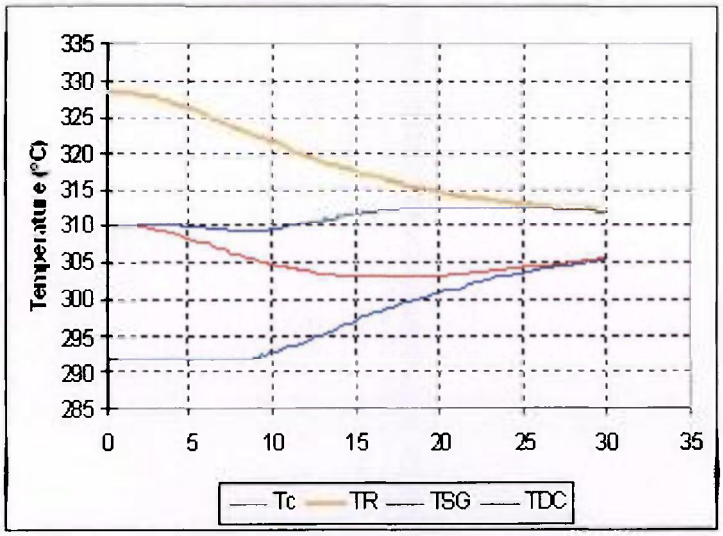
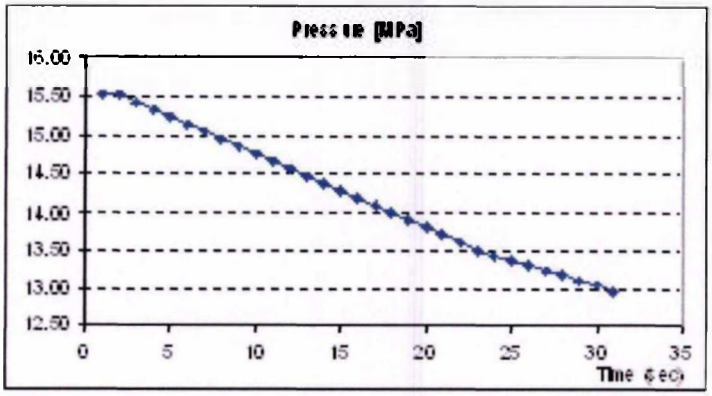
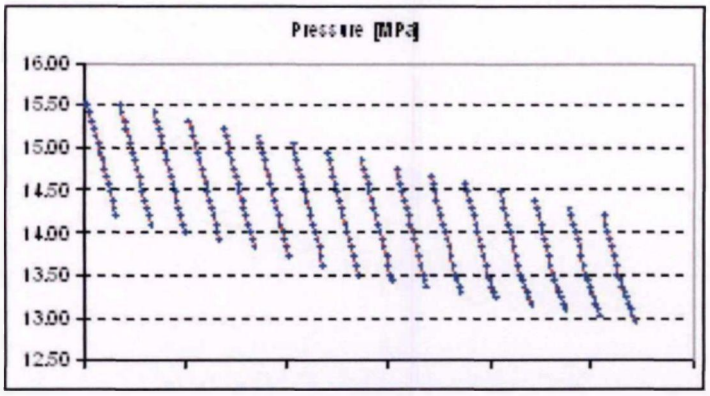


FIGURA 4.11 - Resposta de temperatura para SCRAM. Onde T_C é temperatura do refrigerante, T_R é temperatura do reator, T_{SG} é temperatura no gerador de vapor e T_{DC} temperatura no canal de descida.

O sistema para identificação de transientes do IRIS trabalha com a idéia de que algumas variáveis são capazes de caracterizar um tipo de transiente. Essas variáveis são: a potência do reator, a potência do gerador de vapor, a pressão do pressurizador, o nível de água e a temperatura em diferentes pontos como citado na FIGURA 4.11. São adquiridos os valores máximos e mínimos de cada variável em cada condição, resultando numa entrada de 16 dimensões, ou equivalentemente, em 16 unidades na camada de entrada para o SOM. Por outro lado, o simulador tem um buffer para coletar dados por alguns segundos de cada uma das principais variáveis, que formarão um espectro que será a entrada para o SOM. Por exemplo, a variação de pressão durante um acidente com pequena perda de refrigerante é traduzida para pacotes de 16 segundos. Cada pacote contém informação de 16 segundos de tempo transiente e a diferença de chegada entre pacotes é de 1 segundo (FIGURA 4.12).



(a)



(b)

FIGURA 4.12 - Variação de pressão (a) e conteúdo para um buffer de 16 s (b) para LOCA.

Em trabalho anterior de Baptista e Barroso, 2003, os dados eram normalizados somente com respeito a cada valor de variável. Na nova versão, a normalização é do conteúdo do buffer. Trabalhou-se principalmente com uma camada de saída de rede hexagonal, no modo não-supervisionado, por 1000 épocas ou iterações para ordenamento e mais 8000 épocas para convergência. O objetivo foi verificar a habilidade da ferramenta em fazer a distinção clara entre quatro tipos de condição de operação: estado estacionário, transiente em rampa, transiente em degrau e evento anormal ou acidental. A TABELA 4.1, de transientes normais, e a TABELA 4.2, de transientes anormais, listam os pacotes de dados para diversos transientes usados no treinamento da rede.

TABELA 4.1 - Lista de transientes - transientes normais. As percentagens se referem à potência nominal.

Seqüencial	Operação
1	20% em estado estacionário
2	25% em estado estacionário
3	30% em estado estacionário
4	35% em estado estacionário
5	40% em estado estacionário
6	45% em estado estacionário
7	50% em estado estacionário
8	55% em estado estacionário
9	60% em estado estacionário
10	65% em estado estacionário
11	70% em estado estacionário
12	75% em estado estacionário
13	80% em estado estacionário
14	85% em estado estacionário
15	90% em estado estacionário
16	95% em estado estacionário
17	100% em estado estacionário
18	105% em estado estacionário
19	110% em estado estacionário
20	25% em rampa +5%/min
21	30% em rampa +5%/min
22	40% em rampa +5%/min
23	50% em rampa +5%/min
24	60% em rampa +5%/min
25	70% em rampa +5%/min
26	80% em rampa +5%/min
27	90% em rampa +5%/min
28	100% em rampa –5%/min
29	90% em rampa –5%/min
30	80% em rampa –5%/min
31	70% em rampa –5%/min
32	60% em rampa –5%/min
33	50% em rampa –5%/min
34	40% em rampa –5%/min
35	30% em rampa –5%/min
36	100%→ degrau –10%/min
37	90% → degrau –10%
38	80% → degrau –10%
39	70% → degrau –10%
40	60% → degrau –10%
41	50% → degrau –10%
42	40% → degrau –10%
43	30% → degrau –10%
44	20% → degrau +10%
45	30% → degrau +10%
46	40% → degrau +10%
47	50% → degrau +10%
48	60% → degrau +10%
49	70% → degrau +10%
50	80% → degrau +10%
51	90% → degrau +10%

TABELA 4.2 - Lista de transientes - transientes anormais. As percentagens se referem à potência nominal.

Seqüencial	Operação
52	100% → desarme da turbina
53	100% → SCRAM
54	100% → abertura inadvertida da válvula alívio
55	100% → degrau –70%
56	100% → *small* LOCA

4.2.1.2 Reator de Angra 2

Os dados de treinamento para identificação de transientes do reator de Angra 2 foram adquiridos de Baptista, 2002, artigo de redes neurais artificiais aplicadas à engenharia nuclear. Os resultados de classificação para Angra 2 que serão apresentados no próximo capítulo também foram comparados com os obtidos de Baptista, 2002. Na TABELA 4.3 estão listadas as condições de operação analisadas e na TABELA 4.4 as medições para cada operação.

TABELA 4.3 - Lista de condições de operação para a análise de transientes de Angra com os respectivos índices numéricos.

	Condição de operação	Índice
NORMAIS	Variação de potência, degraus de +10%	1
	Variação de potência, degraus de –10%	2
	Rampa, 100 a 40%, 10%/min	3
	Rampa, 40 a 100%, 10%/min	4
ANORMAIS	Desligamento do reator (trip)	5
	Desligamento da turbina (trip)	6
	Fechamento inadvertido de 1 válvula de vapor	7
	Fechamento inadvertido de 4 válvulas de vapor	8
	Perda parcial de vazão (referente a 1 bomba)	9
EMERGÊNCIA	Falha de todas as bombas	10
	Trip da turbina sem desvio de vapor	11
	Falha de tubos do gerador de vapor	12
	Pequena LOCA	13
	Pequena ruptura da tubulação de vapor	14
FALHA	Grande LOCA	15

TABELA 4.4 - Lista das variáveis de Angra e seus valores para dados de treinamento. Os índices correspondem às condições de operação explicitadas na TABELA 4.3.

Índice	$P_{PR}(+)$	$P_{PR}(-)$	$L_{PZ}(+)$	$L_{PZ}(-)$	$T_R(+)$	$T_R(-)$	$Q_R(+)$	$Q_R(-)$	$W_S(+)$	$W_S(-)$	$P_S(+)$	$P_S(-)$	$W_{PR}(+)$	$W_{PR}(-)$	R_C	P_C	Label
1	0.000	0.020	0.013	0.040	0.006	0.003	0.144	0.000	0.111	0.000	0.000	0.059	0.000	0.000	0.000	0.000	1
2	0.020	0.010	0.040	0.013	0.003	0.006	0.000	0.130	0.000	0.100	0.046	0.000	0.000	0.000	0.000	0.000	1
3	0.020	0.020	0.013	0.013	0.000	0.043	0.000	0.600	0.000	0.600	0.177	0.000	0.000	0.000	0.000	0.000	1
4	0.020	0.020	0.013	0.013	0.045	0.000	1.500	0.000	1.500	0.000	0.000	0.200	0.000	0.000	0.000	0.000	1
5	0.000	0.063	0.000	0.280	0.000	0.086	0.000	0.950	0.000	0.950	0.250	0.000	0.000	0.000	0.000	0.000	2
6	0.025	0.032	0.013	0.120	0.000	0.049	0.000	0.700	0.000	0.700	0.300	0.000	0.000	0.000	0.000	0.000	2
7	0.013	0.050	0.013	0.400	0.000	0.091	0.000	0.950	0.020	1.000	0.250	0.000	0.000	0.000	0.000	0.000	2
8	0.063	0.100	0.093	0.410	0.000	0.101	0.000	0.950	0.100	0.980	0.375	0.000	0.000	0.000	0.000	0.000	2
9	0.025	0.038	0.000	0.160	0.006	0.028	0.000	0.500	0.000	0.450	0.170	0.000	0.000	0.340	0.000	0.000	2
10	0.025	0.032	0.013	0.120	0.006	0.049	0.000	0.950	0.000	1.000	0.375	0.000	0.000	0.950	0.000	0.000	3
11	0.070	0.101	0.093	0.410	0.000	0.101	0.000	0.950	0.100	0.980	0.375	0.000	0.000	0.000	0.000	0.000	3
12	0.000	0.519	0.650	0.493	0.000	0.117	0.000	0.950	0.100	1.000	0.203	0.047	0.000	0.000	0.000	0.000	3
13	0.000	0.524	0.000	0.900	0.000	0.210	0.000	0.950	0.000	1.000	0.000	0.250	0.000	0.950	0.100	0.100	3
14	0.000	0.175	0.000	0.450	0.000	0.160	0.000	0.950	0.000	1.000	0.000	0.280	0.000	0.000	0.000	0.100	3
15	0.000	0.974	0.000	1.000	0.000	0.828	0.000	0.950	0.000	1.000	0.000	0.747	0.000	1.000	1.000	4.000	4

4.2.2 Dados da logística do gás natural

No Brasil, o negócio gás natural ainda enfrenta dificuldades na escolha e no uso de métodos nas centrais de supervisão da logística (controle do fluxo para recebimento/entrega e qualidade do produto). Em parte a causa é a falta de automação total ou parcial para a coleta remota de dados. Progressivamente esta deficiência está sendo sanada, mas enquanto em alguns sistemas ou subsistemas de gasodutos a automação está instalada e homologada, em outros, nem sequer há previsão, como muitos trechos do Nordeste e a malha de dutos do Espírito Santo.

Outra causa da dificuldade na supervisão é a falta de ferramentas eficientes e rápidas para diagnóstico de sistema e auxílio na tomada de decisão, que tenham boa usabilidade e escalabilidade.

O uso do gás natural já tem participação importante na matriz energética brasileira (FIGURA 4.13) e se busca a sua utilização principalmente nas indústrias e termelétricas para substituição gradual do óleo combustível, uma vez que o gás natural é menos poluente. Também existe o incentivo de uso para o consumidor menor, como padarias e residências interioranas, para que se substitua o carvão vegetal, diminuindo a necessidade de desmatamento ou o plantio de árvores não nativas. Para essas e muitas outras utilizações, é necessário que a malha de gasodutos brasileira cresça muito mais.

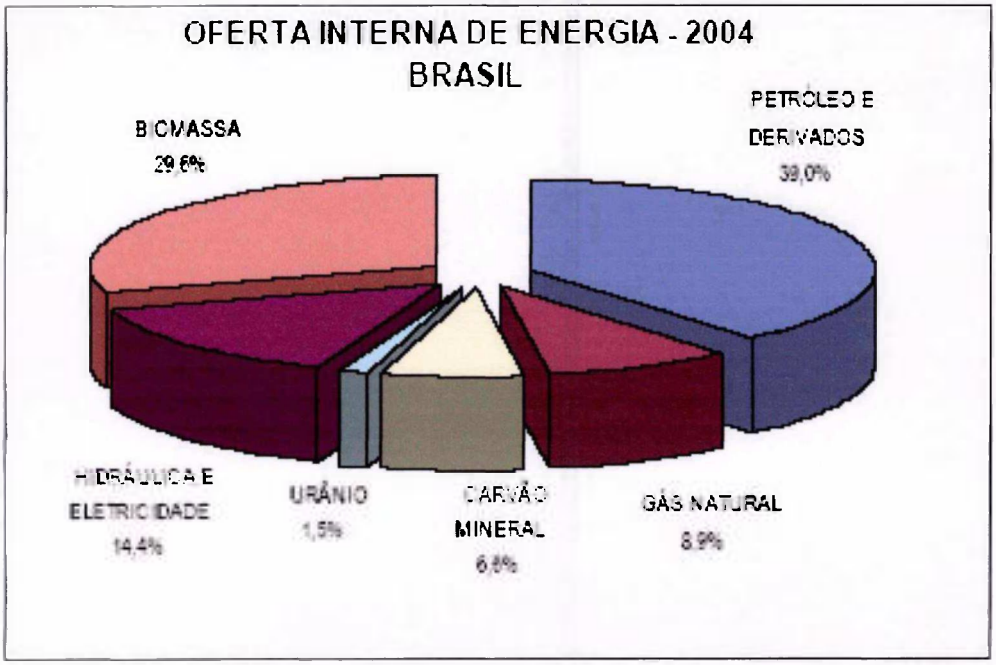


FIGURA 4.13 – Oferta Interna de Energia no Brasil, ano-base 2004. A Oferta Interna de Energia – OIE representa a energia que se disponibiliza para ser transformada (refinarias, carvoarias, termelétricas etc.), distribuída e consumida nos processos produtivos do País. O total do consumo final nos setores econômicos mais o que foi perdido na distribuição, armazenagem e processos de transformação é igual à OIE.Fonte: MME.

Grandes investimentos em construção e expansão de sistemas de gasodutos estão sendo realizados, que acrescentará mais 4160 km de gasodutos aos 5407 km existentes, como o Sistema GASENE (Gasoduto Sudeste-Nordeste), que fará a interligação entre os Sistemas Sudeste, Espiríto Santo e Nordeste. Na região norte, o Gasoduto Coari-Manaus levará o gás de Urucu à capital do Amazonas, visando principalmente a geração de energia elétrica e consumo veicular. A malha Sudeste ganhará reforços com mais três gasodutos, que serão o Gasoduto Campinas-Rio, uma ramificação do GASBEL para São Carlos, que liga Rio de Janeiro a Belo Horizonte, e uma ramificação do GASAN para Itu, que liga Mauá a Santos. A malha Nordeste será ampliada priorizando o interior dos estados de Alagoas, Pernambuco, Paraíba e Rio Grande do Norte. A região Sul terá a construção do Gasoduto Cruzeiro do Sul, ligando Porto Alegre a Montevidéu, se interligando com o GASBOL (Gasoduto Brasil-Bolívia), sem mencionar o trecho Porto Alegre – Uruguaiana já em construção. Atualmente, o consumo nacional de gás natural é de 49 milhões m^3/dia, sendo 23 milhões m^3/dia importado (FIGURA 4.14).

A projeção de consumo do gás natural para 2010 é de 100 milhões de m³/dia, implicando na participação de 15% na matriz energética brasileira. As origens do gás natural são a importação da Bolívia, as imensas reservas litorâneas brasileiras (FIGURA 4.15) e as reservas em terras brasileiras (43% da origem nacional), principalmente nos estados do Amazonas (quase 50% da produção terrestre), Bahia, Alagoas e Espírito Santo.

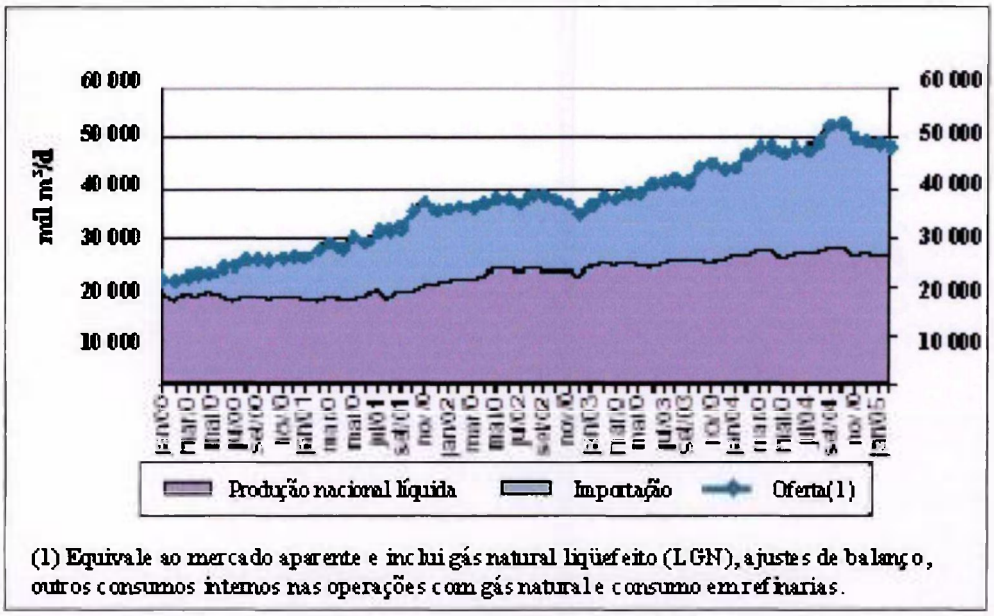


FIGURA 4.14 - Composição de oferta de gás natural – jan/2000 a fev/2005. Fontes: ANP/SCM; ANP/SDP.

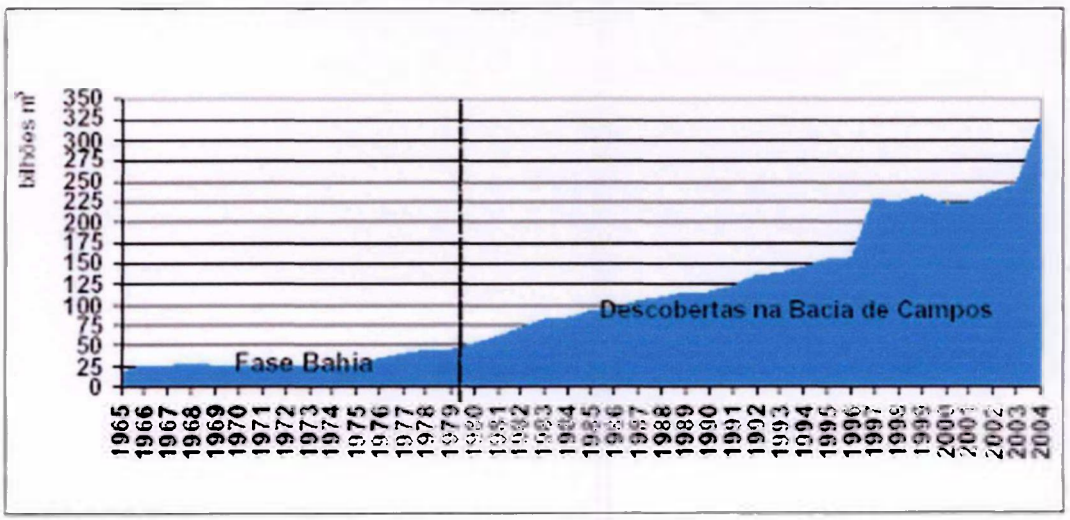


FIGURA 4.15 - Reservas provadas de gás natural no Brasil – 1965-2004. Fontes: ANP/SDP; MME.

4.2.2.1 Os sistemas de gasodutos

FIGURA 4.16 - Infra-estrutura brasileira de transporte de gás natural. Fonte: ANP.

Um gasoduto possui estações de entrega ou transferência de gás genericamente chamadas de "pontos de medição" cujos atributos principais são seus medidores, seu correspondente local de coleta de amostras, o destino do gás, a origem do gás, a natureza da operação (transferência, gás natural veicular, unidade de produção de hidrogênio etc.) e o órgão responsável pelas medições. Um ponto de medição pode possuir ramais (FIGURA 4.17), cada ramal com seu conjunto de medidores. Podem existir até 5 ramais por ponto. Cada conjunto medidor possui sensores de fluxo, de volume acumulado, de pressão e de temperatura. Mesmo hoje não são muitos os pontos de medição que possuem cromatografia automatizada em linha, por isso o local da coleta de amostragem pode ficar razoavelmente distante do ponto de medição. Um local de amostragem pode servir a um trecho inteiro de gasoduto, com vários pontos de medição. Se o sentido de fluxo do gás natural inverte, invariavelmente o local de amostragem é outro, refletindo nova origem.

Uma origem ou destino do gás natural são as unidades de processamento. O processamento do gás natural se faz necessário para dessulfurização, e também para a retirada de olefinas e elementos pesados que neste caso seguem para a indústria petroquímica.

Convém observar que, com a regra *open access* para os gasodutos, ou seja, utilização aberta a qualquer empresa de gás natural, não faz mais sentido o controle de estoque, que é a quantidade que permanece nos trechos de gasodutos, uma vez que, por sua baixa densidade, não há tancagem para o gás natural.

FIGURA 4.17 - Vistas diversas do ponto de medição City-Gate Capuava (Mauá-SP) e seus 3 ramais.

Os dados de entrada apresentados neste trabalho foram recolhidos de dois trechos de maior consumo no país, GASAN (Gasoduto Mauá - Santos) e GASPAL (Gasoduto Mauá - Paulínia), abrangendo percurso da Refinaria de Cubatão - SP até Pindamonhangaba-SP (FIGURA 4.18).

Para o Faturamento e para a Central de Supervisão da Logística do Gás Natural – Petrobras, é importante, e ao mesmo tempo satisfaz ao mínimo a supervisão, os dados consolidados uma vez ao dia. Estes dados consolidados possuem o volume totalizado e as médias de temperatura e pressão do dia, fechados à meia-noite.

4.2.2.2 Preparação dos dados

Para a classificação de entregas pelo SOM customizado especialmente para o negócio, foram consideradas entradas, ao mesmo tempo, de um a três pontos de medição da operação genérica vendas (entrega ao consumidor final ou às distribuidoras de combustível). Para a operação de grandes vendas, os pontos também são chamados de city-gates. Restringiu-se o estudo para pontos de vendas do GASPAL e do GASAN, conforme TABELA 4.5 retirada de parte de uma entidade do banco de dados corporativo para o gás natural.

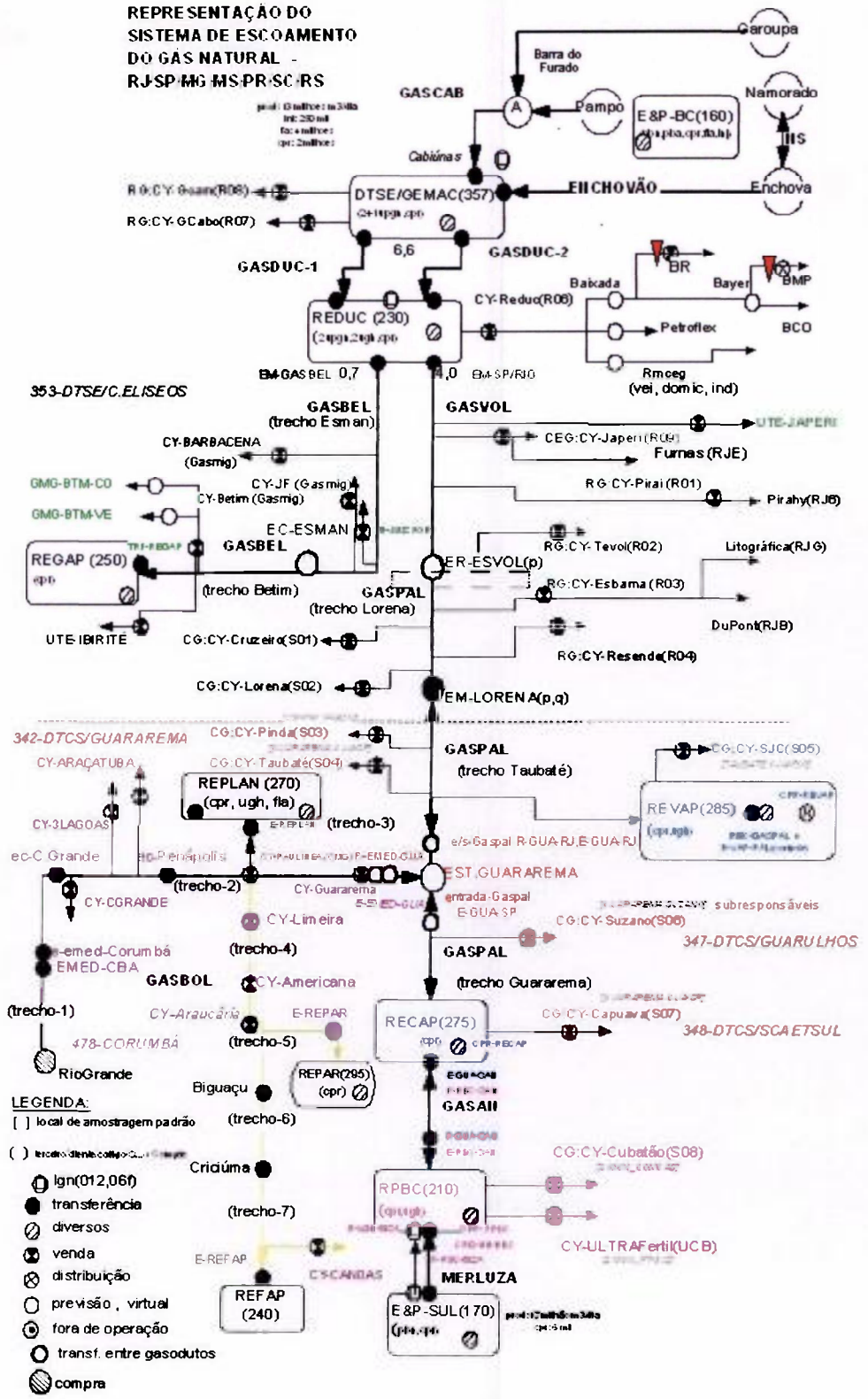


FIGURA 4.18 - Diagrama logístico dos gasodutos com seus trechos dos Sistemas Sudeste e Gasbol (Gasoduto Brasil-Bolívia). As proprietárias são Transpetro (da holding Petrobras) e TBG (empresa com 51% de capital da Petrobras) respectivamente.

TABELA 4.5 - Dados georreferenciados dos pontos de medição, no grupo vendas do GASPAL e GASAN. Os valores de ordenação serão usados como label *para o vetor de dados de entrada.*

Grupo	Código do ponto de medição	Nome	Ordenação no gasoduto	Distância da origem (km)	Latitude (°)	Longitude (°)
GASPAL/GASAN VENDAS	CY-ULTRAF	RPBC	-4	-69,8	-23,93056	313,57472
GASPAL/GASAN VENDAS	CY-CUBATAO	CUBATAO	-3	-66,6	-23,89500	313,57472
GASPAL/GASAN VENDAS	CY-CAPUAVA	CAPUAVA	-2	-52,8	-23,65639	313,51944
GASPAL/GASAN VENDAS	CY-SUZANO	SUZANO	-1	-31,5	-23,54250	313,68917
GASPAL/GASAN VENDAS	E-GUA-SP	ORIGEM	0	0,0	-23,41500	313,96500
GASPAL/GASAN VENDAS	CY-SJC	S. J. CAMPOS	1	30,3	-23,17944	314,11306
GASPAL/GASAN VENDAS	CY-TAUBATE	TAUBATE	2	65,4	-23,02639	314,44472
GASPAL/GASAN VENDAS	CY-PINDA	PINDAMONHANGABA	3	80,2	-22,92389	314,53833

Importante componente do vetor de entrada, o poder calorífico superior (PCS) é a quantidade de calor libertada pela combustão completa de uma quantidade de combustível com a correspondente quantidade estequiométrica de ar seco, ambos a 15,6 °C (60 °F), e até que os produtos atinjam 15,6 °C também, considerando, ao final da combustão, o vapor de água resultante condensado, ou seja, o calor do vapor recuperado. Para complemento, o poder calorífico inferior (PCI) é equivalente ao PCS a não ser pelo fato de que o vapor produzido pela reação com o hidrogênio presente no combustível não se condensa, concluindo que PCI < PCS pela diferença de entalpia de condensação da água. Ao se medir o poder calorífico, a pressão de 1 atm deve se manter constante.

Quando a grandeza para medir o combustível é volume, este deve estar normalizado para alguma condição de pressão e temperatura. Aqui se usará, a menos de nota em contrário, a condição de 1atm[11] e 20°C para a normalização dos volumes.

Os volumes consolidados diários além de estarem normalizados para a condição de 1 atm e 20 °C, ao serem usados para a classificação passam por mais uma normalização para o valor energético de PCS=9400 kcal/m³[12]. Assim cada metro cúbico de gás natural, a 1 atm e 20 °C que consta para a classificação é capaz de produzir uma energia de 9400 kcal

[11] 1 atmosfera ≡ 1,033 kgf/cm² ≡ 1013,25 hPa ≡ 1,013 bar ≡ 14,70 psi
[12] V_{PCS} = V . (PCS medido/9400).

na combustão. Em geral, os contratos de envio e recebimento de gás natural são fechados para volumes normalizados para energia, assegurando que o valor trocado é em última instância energia, não a quantidade de combustível. Isso também assegura que os volumes que entram para a classificação não sofrem dependência do PCS, e nem da pressão e da temperatura pela normalização anterior. Assim o PCS, e em menor grau, a densidade são exclusivamente indicadores da qualidade[13]. Na logística, a pressão, e em menor grau, a temperatura são exclusivos para a detecção de problemas no duto ou para inferir se um ponto possuirá condições de atender a demanda, uma vez que queda de pressão pode vir a comprometer a entrega do dia. Uma vez detectada uma queda de pressão abaixo dos valores comuns, devem ser tomadas ações como acionar responsáveis por estações compressoras ou bloquear fluxos para outros pontos.

Como a unidade caloria (cal) é na verdade um indicador de energia, e não a medição direta de energia, há controvérsias de qual o valor da constante de conversão para joules. Com contratos fechados com a Petrobras, é adotada a caloria termoquímica[14] que vale exatos 1,484 J. A densidade mencionada para o gás natural é a densidade relativa ao ar seco que é o quociente entre a massa do gás contida em um volume arbitrário e a massa de ar seco com composição padronizada pela ISO 6976 que deve ocupar o mesmo volume sob condições normais de temperatura e pressão. As pressões apresentadas aqui, a menos de menção especial, são pressões absolutas.

O gás natural é encontrado em reservatórios subterrâneos em muitos lugares do planeta, tanto em terra quanto no mar, tal qual o petróleo, sendo considerável o número de reservatórios que contém gás natural associado ao petróleo. Nestes casos, o gás recebe a designação de gás natural associado. Quando o reservatório contém pouca ou nenhuma quantidade de petróleo, o gás natural é dito não-associado.

A composição do gás natural bruto é função de uma série de fatores naturais que determina o seu processo de formação e as condições de acumulação do seu reservatório de origem. Bruto ou processado, a TABELA 4.6 fornece as especificações mínimas para um gás ser natural poder ser comercializado.

---

[13] Uma outra medida de qualidade é a combinação destas duas grandezas chamada de índice de Wobbe, W = PCS/densidade relativa$^{1/2}$. É uma medida da quantidade de energia disponibilizada em um sistema de combustão através de um orifício injetor com a mesma pressão a montante e a jusante deste orifício. A quantidade de energia disponibilizada é uma função linear do índice de Wobbe.

[14] A caloria do Sistema Internacional vale aproximadamente 1,4855 J que é a energia necessária para elevar de 14,5 °C para 15,5 °C 1 g de água destilada a pressão de 1 atm.

TABELA 4.6 - Especificações para o gás natural segundo Portaria n° 128, 28-AGO-2001, Regulamento Técnico ANP n° 3/2001.

Característica[15]	Unidade	Mínimo	Máximo
PCS	kcal/m³	9100	10990
Índice de Wobbe	kcal/m³	11850	13378
Metano ou C1	% em volume	86,0	-
Etano ou C2	% em volume	-	10,0
Propano ou C3	% em volume	-	3,0
Butano e mais pesados ($C4^+$)	% em volume	-	1,5
Oxigênio (O_2)	% em volume	-	0,5
Nitrogênio (N_2)	% em volume	-	2,0
Inertes ($N_2 + CO_2$)	% em volume	-	4,0
Gás sulfídrico	% em volume	-	0,0007 (10,0 mg/m³)
Ponto de orvalho	°C	-	-45 °C

O PCS do gás natural é calculado a partir da composição de amostras coletadas diariamente e inseridas em banco de dados corporativo de qualidade, como o iLab. À disposição dos operadores e supervisores responsáveis, existem aplicativos, como o iGás (FIGURA 4.19) com rotinas que calculam o PCS, PCI e densidade a partir dos componentes químicos do gás natural assim que se dá entrada de um volume e, portanto, o PCS está sempre disponível junto com o volume corrigido por outra rotina para pressão de 1 atm e 20°C, correção esta dependente do tipo de medidor de vazão (placa de orifício, turbina, deslocamento positivo etc.). Para classificar uma operação realizada, leva-se em conta o volume diário contratado (QDC, normalizado para energia) e o PCS especificado em contrato. Por exemplo, uma operação saiu das especificações se o volume diário entregue ficou abaixo de 80% do contrato, pois em geral as multas de contrato se aplicam a partir daí. Ainda, como exemplo, uma operação teve um produto de má qualidade se o PCS ficou abaixo de 0,5 % do desejado. Uma operação é ideal se o volume entregue ficou acima de 5% da quantidade diária contratada, uma folga que pode vir a ser negociada no caso de um dia com volume abaixo da QDC. Demais operações que fiquem dentro das faixas citadas são consideradas normais.

[15] Os símbolos C2, $C4^+$..., significam que podem incluir olefinas (insaturados: eteno, propeno, buteno, butino...), mas, no gás natural, as olefinas são traços.
Quando porventura aparecerem os símbolos "n" e "i" como em nC10, iC10, significam cadeia normal ou reta e cadeia ramificada (parafina) respectivamente.

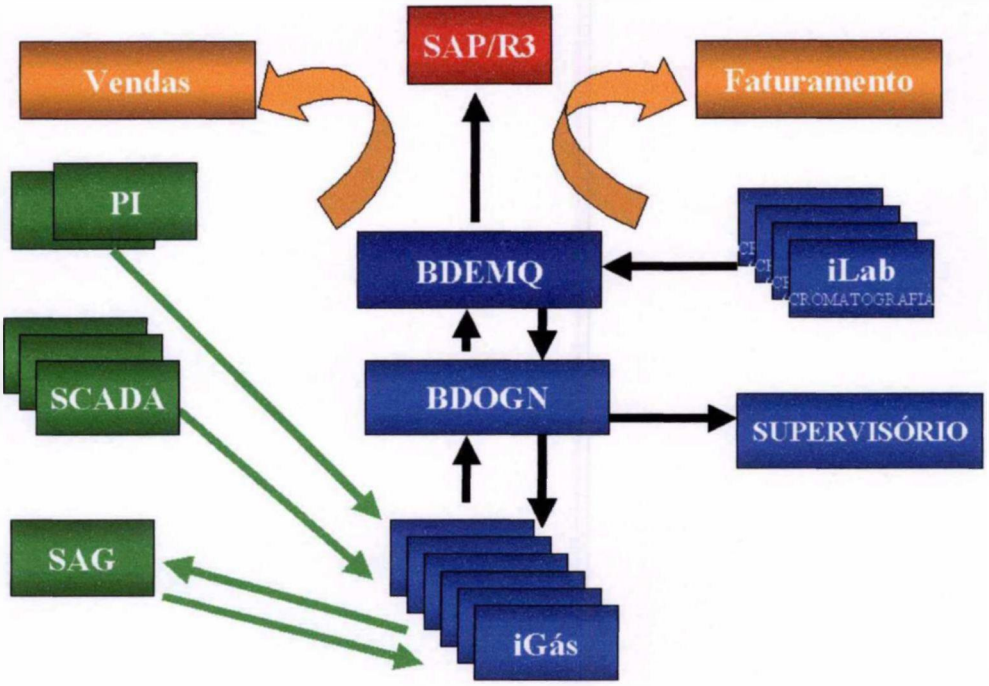


FIGURA 4.19 - Fluxograma dos dados da logística do gás natural. Os sistemas em quadros verdes, à esquerda, são de medição automatizada de campo. Para os sistemas restantes, quanto mais superior na figura, mais corporativos.

O vetor de dados de entrada para o SOM fornece a visão do gasoduto como um sistema coeso de componentes, sem fortemente classificar por cada ponto de entrega. A distinção entre os pontos se dá por uma componente que será um *label*, um índice para o ponto. Outras dimensões são: pressão, temperatura e densidade, ou seja, junto com volume consolidado e PCS, o vetor de dados terá seis dimensões. Neste trabalho, foram utilizados intervalos de tempo regulares para os dados dos pontos de entrega; na verdade, um dado consolidado por dia, que corresponde ao volume de fechamento do dia, de interesse do Faturamento. Assim a pressão e a temperatura trazidas correspondem à média diária.

A FIGURA 4.20 ilustra um arquivo-texto com dados brutos obtidos por uma *query* no banco do gás natural, por questão de espaço estão somente exibidas medições de uma semana (7 dias). Os dados são gravados em arquivo-texto antes de ser processado pelo SOM, o que é uma vantagem para a distribuição e autonomia entre as camadas do sistema de *software*. Cada linha possui os seguintes campos (separados por ";"): índice seqüencial da linha de dado; *label* do ponto de medição; data e hora do volume consolidado; volume consolidado no dia, em m^3, e normalizado para 1 atm e 20 °C; PCS em $kcal/m^3$; distância (em km) da origem do gasoduto; pressão em kgf/m^2; temperatura em °C, densidade relativa ao ar seco; última data de atualização de cadastro de medidor para o ponto – para que a

query não adquira características desatualizadas de medição, como unidades de medida erradas; e índice seqüencial para a data – no exemplo da FIGURA 4.20, como são pedidos dados de três pontos, então aparece um mesmo índice em três linhas de dados.

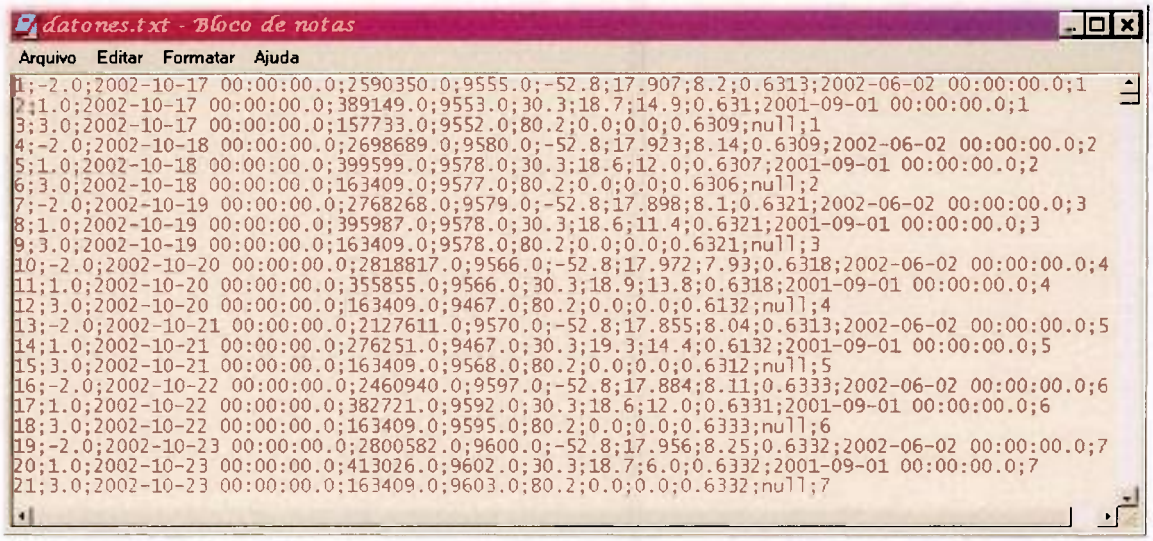


FIGURA 4.20 - Arquivo-texto com dados gravados da aplicação-interface com o banco de dados. As colunas estão separadas por ponto e vírgula e são, na ordem, número seqüencial de medição, data do volume consolidado, label que indica posição em relação à origem, volume consolidado (m³), PCS (kcal/m³), distância em relação à origem do gasoduto (km), pressão (kgf/cm²), temperatura (°C), data da última atualização dos parâmetros dos medidores; número seqüencial de dia.

O *applet* selecionará do arquivo texto da consulta somente o necessário para compor o vetor dados de entrada, e gravará, para efeito de registro e análise, os componentes do vetor de dados em outro arquivo texto conforme o exemplo da FIGURA 4.21. No primeiro e último campos está o índice seqüencial, para facilitar uma análise; o segundo campo é do *label* multiplicado por 1/10 ; o terceiro campo corresponde ao volume consolidado em m³ e normalizado para energia de 9400 kcal/m³; em seguida vem o PCS em kcal/m³; a pressão em kgf/cm²; a temperatura em °C; e a densidade relativa ao ar seco.

No programa, há um método que trata dos dados omissos ou faltantes (*faults*) fazendo uma média dos valores que estão presentes em todo o conjunto de entrada. E, quando não é possível fazer a média pela falta de qualquer dado para um componente, são colocados no lugar valores típicos de medição. No exemplo da FIGURA 4.21, os valores de 9,89 (°C) para a temperatura e 18,026 (kgf/cm²) foram obtidos realizando-se a média de todas as temperaturas e pressões, respectivamente, do conjunto de amostras.

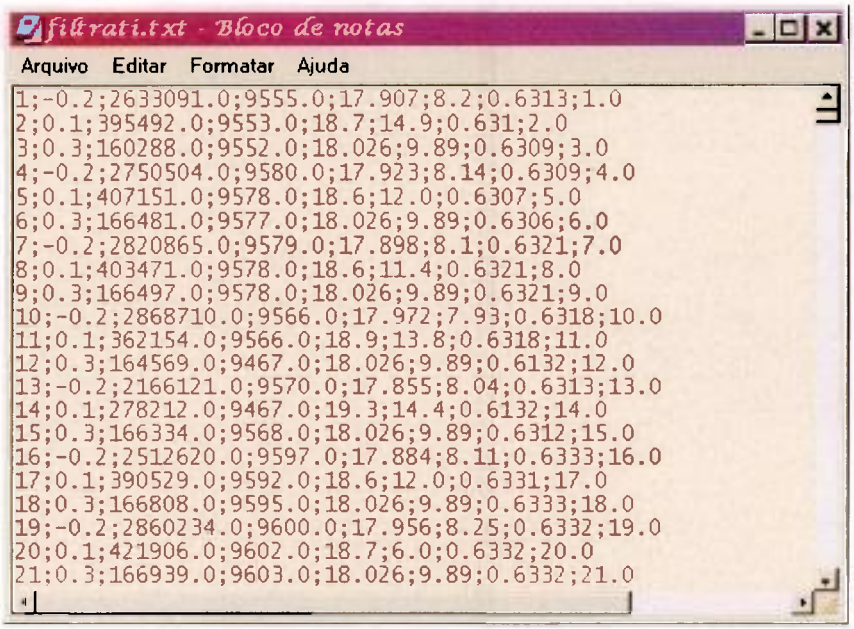


FIGURA 4.21 - Arquivo texto com os campos realmente necessários para o vetor de entrada e o volume normalizado para PCS de 9400 kcal/m3.

Como cada componente possui magnitude diferente, até milhões de unidades para volume e menor que 1 para densidade, existe um outro método que normaliza todas as componentes de medição para uma mesma magnitude. Primeiramente cada valor original de uma grandeza é substituído pela diferença entre o máximo que pode assumir no processo e o valor original dividido pela diferença entre os valores máximo e mínimo que pode assumir no processo. O que resulta passa por uma segunda transformação que é a multiplicação por um fator de sensibilidade ζ. A razão deste fator de sensibilidade é que a variação entre os volumes chega a 163% e a variação entre os PCSs não ultrapassa 3%. Como pequenas variações do PCS são críticas, é preciso uma distensão na escala. A equação que segue sumariza estas duas transformações,

$$x^{j}_{c_{normal}} = \left(\frac{x^{j}_{c_{max}} - x^{j}_{c}}{x^{j}_{c_{valor}} - x^{j}_{c_{min}}} \right) \cdot \zeta_{c} \qquad (4.1)$$

onde ζ é o fator de sensibilidade, c indica um componente do vetor de entrada e j o vetor de entrada. Nos treinamentos foram usados fatores de distensão de 10^{-2} a 1 para o volume, e de 10^{1} a 10^{2} para o PCS. Para os demais componentes ζ foi 1. Ao invés de multiplicação, podia-se trabalhar com logaritmo ou potenciação, porém a variação deixaria de ser linear.

Após esta etapa de tratamento, mais uma vez o resultado será modificado dividindo-o pela soma de todos os componentes tratados pela equação (3.49), por vetor de entrada, de modo que a soma dos componentes dê 1.

$$x^{j}_{c_{nova}} = \frac{x^{j}_{c_{velha}}}{\sum_{i} x^{j}_{c_{velha}}}. \qquad (4.2)$$

Agora cada componente será normalizada para seu valor ficar entre –1 e +1, para todo o conjunto de entradas selecionado na consulta ao banco de dados. O resultado de todo este tratamento é também registrado em um arquivo texto conforme exemplificado na FIGURA 4.22. Nesta relação, a segunda e última colunas possuem o mesmo valor para facilitar o estudo do mapeamento pelo SOM, que são os *labels* da classificação da medição (1, volume ideal; 2, volume regular, 3, volume abaixo do contrato; e 4, PCS abaixo do especificado em contrato).

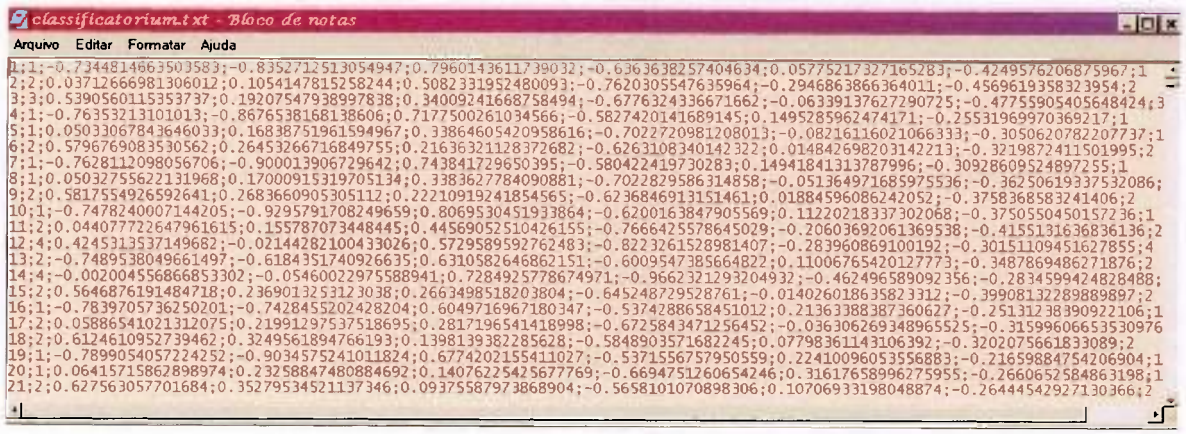


FIGURA 4.22 - Entradas normalizadas para a classificação. Este arquivo-texto é gerado entre processamentos, por isso o formato double *dos dados derivados das medições.*

Alternativamente, as listas geradas relativas à FIGURA 4.20 e à FIGURA 4.22 são também exibidas no console Java (FIGURA 4.23 e FIGURA 4.24), podendo ser ativado pelo navegador Web em Exibir > Console Java ou em Ferramentas > Opções > Avançadas > Console Java ativado ou ainda Ferramentas > Sun Java Console, conforme a versão do navegador. Alternativamente o console Java pode ser ativado por Configurações

> Painel de Controle > Java Plug-in, abrir o Java Plug-in e configurá-lo para exibir o console Java, isso caso o JRE que inclui a máquina virtual seja instalado separadamente do navegador e do ambiente de desenvolvimento j2sdk.

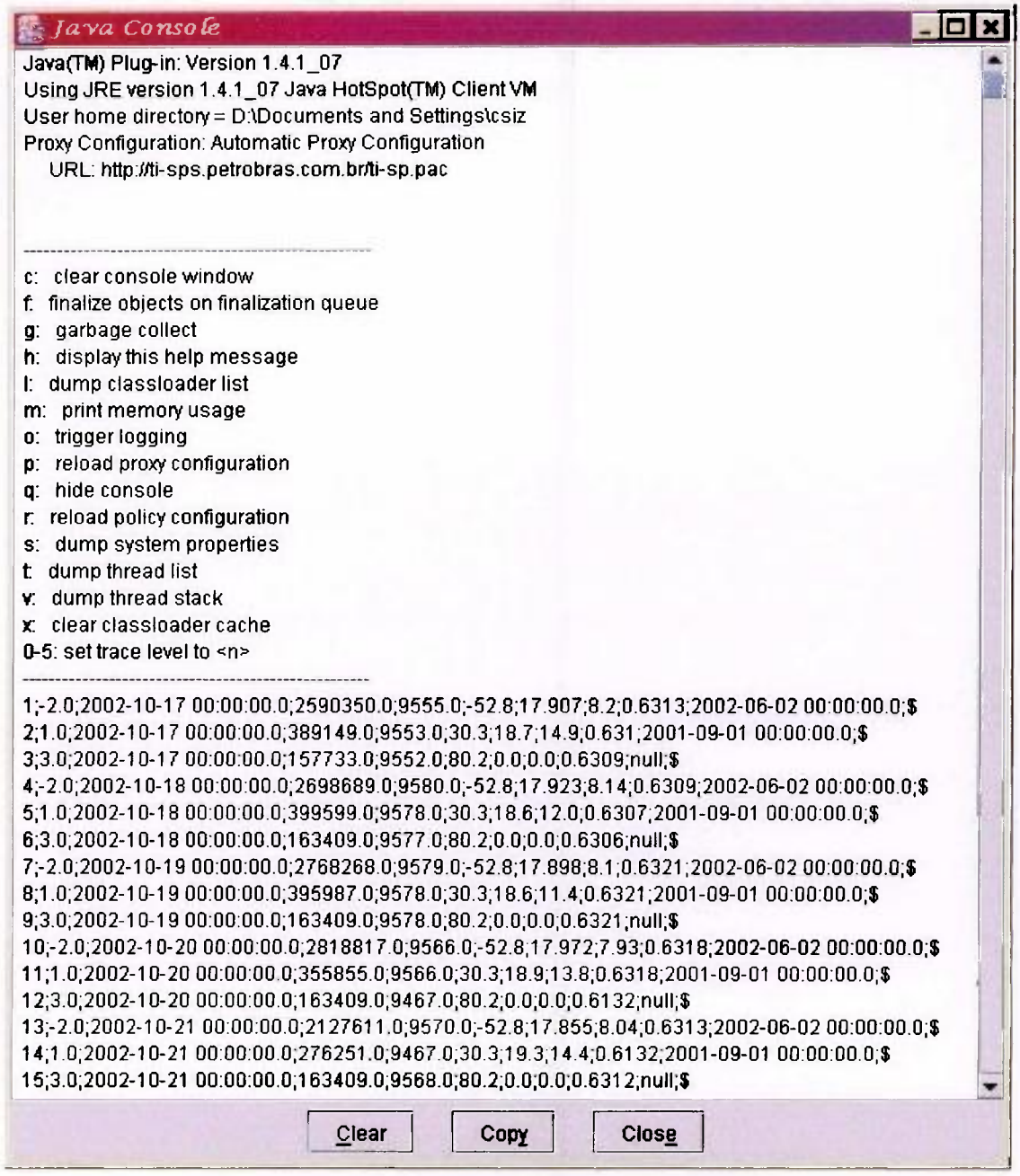


FIGURA 4.23 - Console Java, com listagem semelhante à exibida na FIGURA 4.20.

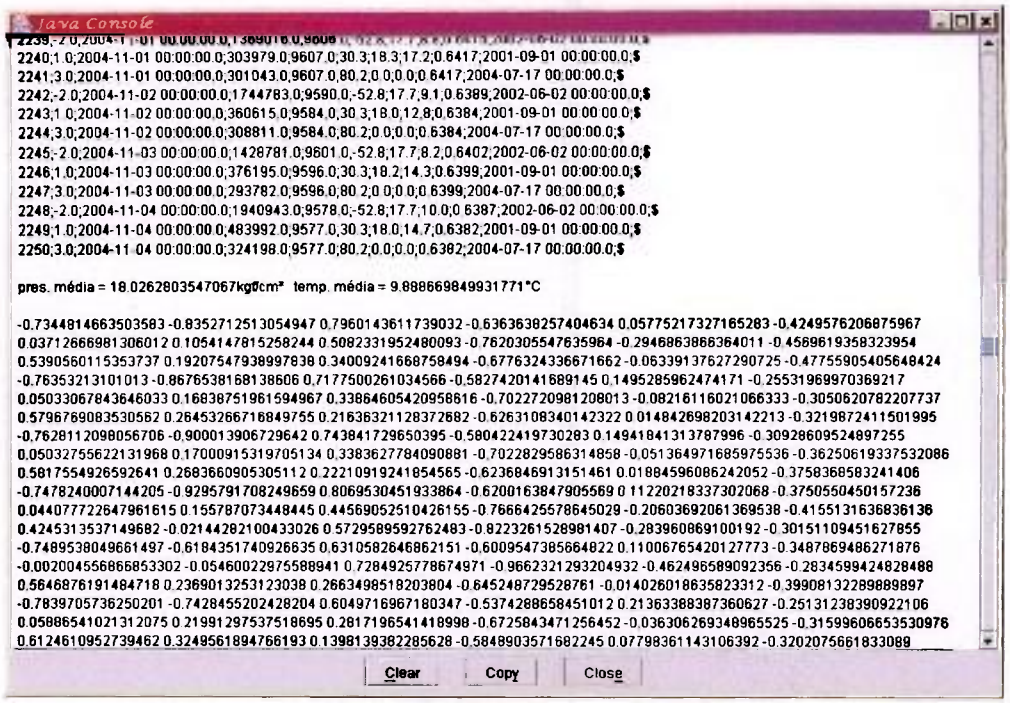


FIGURA 4.24 - Console Java, com listagem semelhante à exibida na FIGURA 4.21.

No console Java outros avisos e resultados são exibidos, como a impressão da unidade vencedora para uma entrada de teste escolhida pelo usuário, corroborando o sinal piscante da tela. Ao término do treinamento serão também impressos os índices seqüenciais das entradas mais próximas (por distância euclidiana) de cada vetor peso (FIGURA 4.25). Há uma outra opção no *applet* que são os testes feitos não com entradas advindas do banco de dados, mas inserindo uma entrada através de clique sobre o campo mencionado na FIGURA 4.3. Neste caso os valores das duas dimensões mais relevantes corresponderão ao local em que o usuário clicou, os valores das demais dimensões serão gerados aleatoriamente (FIGURA 4.26).

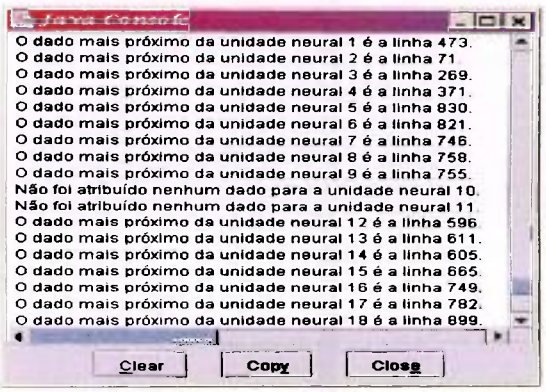


FIGURA 4.25 - Listagem exibindo as entradas mais próximas do vetor peso de cada unidade neural. São as classes destas entradas que fornecem, por sua vez, a classe das unidades neurais.

Em suma, o sistema fornece diversas informações das etapas de seu processamento dos dados e dos resultados do treinamento e do teste, tanto por gravação de arquivos[16], por exibição em interface gráfica colorida e dinâmica e por impressão no console Java.

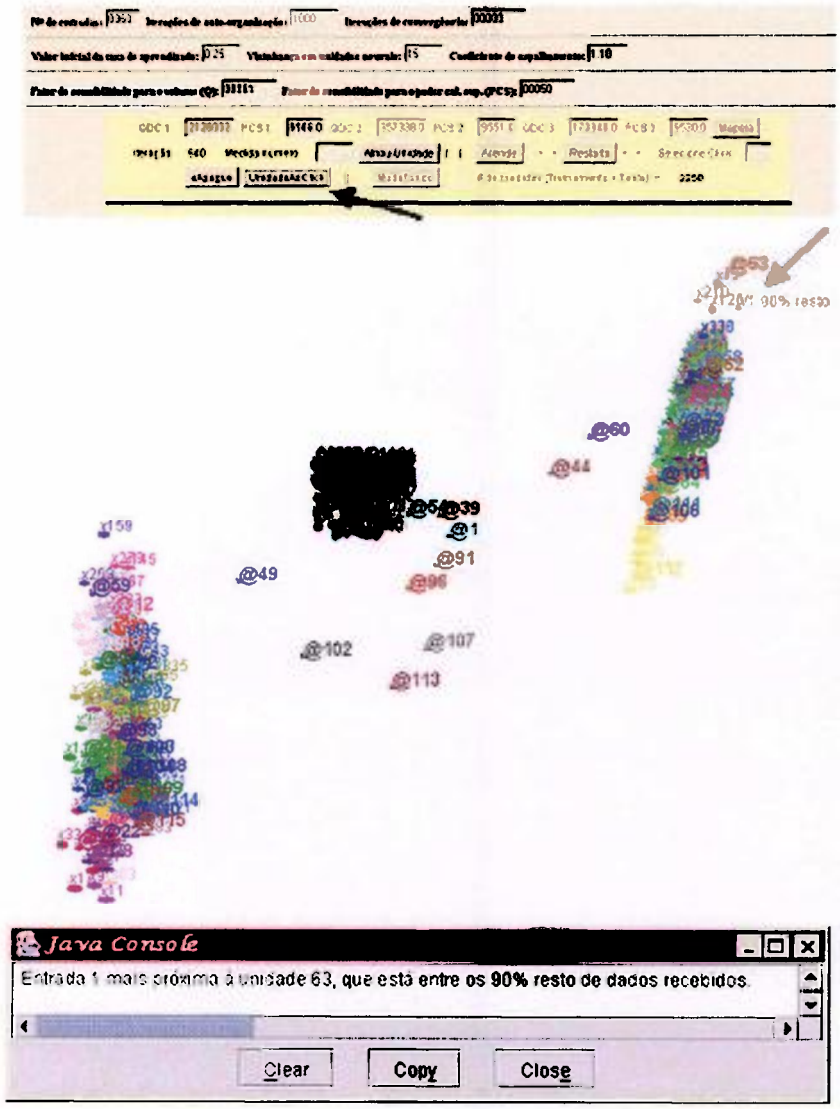


FIGURA 4.26 - Teste utilizando a interface gráfica para a plotagem das entradas.

[16] Por segurança, um applet (arquivos .class), que se assemelha a um executável (arquivos .exe), não tem permissão de gravar qualquer arquivo na máquina do usuário, somente de ler (*read*). Porém para applets conhecidos e específicos, o usuário pode fornecer permissão de exclusão (*delete*) e gravação (*write*) no arquivo .java.policy que deve ser criado, se já não existir, na pasta em que se localiza as configurações (*settings*) do usuário. Em geral é uma pasta que tem o username como nome em *Documents and Settings*, ou, para sistemas operacionais de versões antigas, o .java.policy deve ficar na pasta WINDOWS ou WINNT.

O seguinte texto deve ser editado em .java.policy:

grant codeBase "file:/<caminho absoluto da pasta em que o applet vai criar ou excluir um arquivo> /" {
permission java.io.FilePermission "<<ALL FILES>>", "read,write,delete";};.

5 RESULTADOS

O sistema construído apresentou desvantagens típicas do uso de *applet*, principalmente quando este apresenta animação, tanto é que os arquivos Flash (.swf) substituíram com grande sucesso os *applets* no que se refere a conteúdo animado. O *applet* consome razoável recurso de memória RAM, memória de tela e processamento, dependendo que a máquina *client* tenha memória RAM disponível e espaço no disco principal para memória virtual. Ocorre então que a execução de um *applet* fica sensível a outros programas que estão rodando ao mesmo tempo no *client*.

Em rapidez de cálculos, Java não consegue se rivalizar com Fortran. Atinge performance parecida a C/C++, porém a versatilidade da plataforma Java leva a equilibrar tal desvantagem.

Outro inconveniente que pode ocorrer é a necessidade de instalação ou atualização de *plug-in* Java para rodar *applets*, mesmo sendo este gratuito.

A quantidade de entradas, desde que acima de 1000, ou de unidades neurais, desde que acima de 50, não afeta fortemente o tempo de treinamento. O próprio tamanho físico do *applet* é um fator para se levar em conta por causa do uso da memória para a tela.

Outro gargalo na performance são as gravações e leituras de arquivos-texto feitos pelo *applet*, o I/O (*input/output*) costuma ser lento. No caso de gravação, para arquivos com mais de 1000 linhas, a demora pode atingir 3 minutos. Isso leva o usuário ou as próprias regras de contenção do sistema a considerar apenas o número suficiente de entradas para o objetivo que se pretende.

Dessa maneira, para uma melhor performance deste sistema, uma memória RAM de 512 MB a 1 GB seria necessária para o computador, mesmo assim com o mínimo de *softwares* rodando simultaneamente.

Na TABELA 5.1, os tempos coletados foram para um computador pessoal somente com processos em *background*, sem reinicialização do computador a cada treinamento. Está incluído o tempo de processamento para I/O (*input/output*) de leitura e gravação de arquivos-texto.

TABELA 5.1 - Tempos decorridos para 500 iterações e 1200 entradas.

Versão	Especificação	Tempo
73	367 unidades, precisão dupla, com animação	14m10s
74	367 unidades, precisão simples, com animação	10m27s
75	367 unidades, precisão simples, sem animação	11m47s
68	115 unidades, precisão dupla, com animação	05m10s
65	115 unidades, precisão dupla, com animação	04m28s
54	73 unidades, precisão dupla, com animação	06m05s

Somente para comparar, na TABELA 5.2 é apresentado o tempo de treinamento quando se varia o número de entradas. A partir do terceiro treinamento, o tempo se elevou muito, chegando à casa dos 20 minutos, e praticamente independia do número de entradas, pois o navegador ainda "guardava" em RAM os processos anteriores. Fechar e novamente abrir o navegador alivia este "acúmulo". Porém para voltar à melhor performance somente com o reinício do computador. Foi constatado que aplicativos grandes, como ambientes de desenvolvimento, e arquivos .doc e .xls muito extensos podem causar o mesmo problema.

TABELA 5.2 - Tempos de treinamento decorridos para 367 unidades neurais e 1500 iterações.

# entradas	Tempo
1200	13m28s
900	10m22s
600	10m23s

Com estes dados percebe-se que o tempo de treinamento é afetado por processos em *background* do computador pessoal (antivírus, "*agents*", "*doctors*", "*monitorings*" etc.). Uma solução para melhorar este tempo seria o transporte do código para execução em servidor de alta performance, porém neste caso o tempo sofreria dependência do número de usuários requisitando a aplicação simultaneamente, e mesmo do número total de usuários simultâneos conectados com o servidor para quaisquer outras aplicações implantadas.

Os testes realizados após o treinamento para a logística do gás natural apresentaram resultados exibidos na TABELA 5.3.

TABELA 5.3 - Testes para o mapeamento de características para 115 unidades, 1200 entradas, 2000 iterações e 2 pontos de medição: Capuava e São José dos Campos.

Treinamento	Acertos	Parâmetros
1	70%	η_0=0.25
		ζ_{QDC}=0,1 ζ_{PCS}=10
2	82%	η_0=0.22
		ζ_{QDC}=1 ζ_{PCS}=50
3	83%	η_0=0.20
		ζ_{QDC}=1 ζ_{PCS}=50

Estes resultados demonstram a viabilidade da classificação feita por este SOM para dados de logística, e que os fatores de sensibilidade de fato produzem efeitos na taxa de acertos, e podem ser considerados parâmetros de treinamento adicionais para a rede neural.

6 DISCUSSÃO

6.1 Identificação de transientes

6.1.1 IRIS

Após o treinamento, as unidades do SOM foram testadas na sua resposta e atribuídas "ao tipo de transiente mais afim" que teve como critério a distância euclidiana mais curta entre o vetor peso da unidade e os vetores de entrada associados à unidade, ou seja, a classe do vetor de entrada mais próximo foi atribuída também à unidade. Este método de medida de sensibilidade pode causar resposta não adequada a padrões não presentes no conjunto de treinamento. E a classificação inadequada de transientes pode ser responsável por ações incorretas. Um refinamento através da quantização vetorial poderá melhorar a precisão do sistema, porém esta etapa será abordada em trabalho futuro. Os tipos de condição de operação foram indicados por código numérico e por código de cores no caso da interface gráfica do usuário: 1 e verde para estado estacionário; +2 e laranja-escuro para rampa positiva, -2 e laranja-claro para rampa negativa; +3 e vermelho para degrau positivo, -3 e rosa para degrau negativo e 4 e roxo para eventos anormais (FIGURA 6.1).

O algoritmo fez uma distinção bem clara entre estados estacionários (faixa central e oblíqua na FIGURA 6.1), transientes para potências ascendentes (esquerda na FIGURA 6.1), transientes para potências descendentes (direita na FIGURA 6.1) e transientes anormais (bordas na FIGURA 6.1). Entre transientes para potências ascendentes e descendentes, ocorreu também a separação nítida entre rampas e degraus.

Utilizou-se taxa inicial de aprendizagem η igual a 0,20, e o tempo de processamento numa CPU de 1 GHz e 512 MB de RAM foi de aproximadamente 3 min., para ordenamento e convergência, chegando a 5000 iterações.

As 56 amostras de operação do reator foram apresentadas em ordem aleatória a cada iteração para que o aprendizado não corresse o risco de focalizar nas primeiras entradas. A atualização dos pesos também se deu a cada iteração. A vizinhança inicial foi de 20 unidades neurais para uma topologia plana hexagonal de 115 unidades da camada de saída (aproximadamente 11x11 unidades para uma plana retangular).

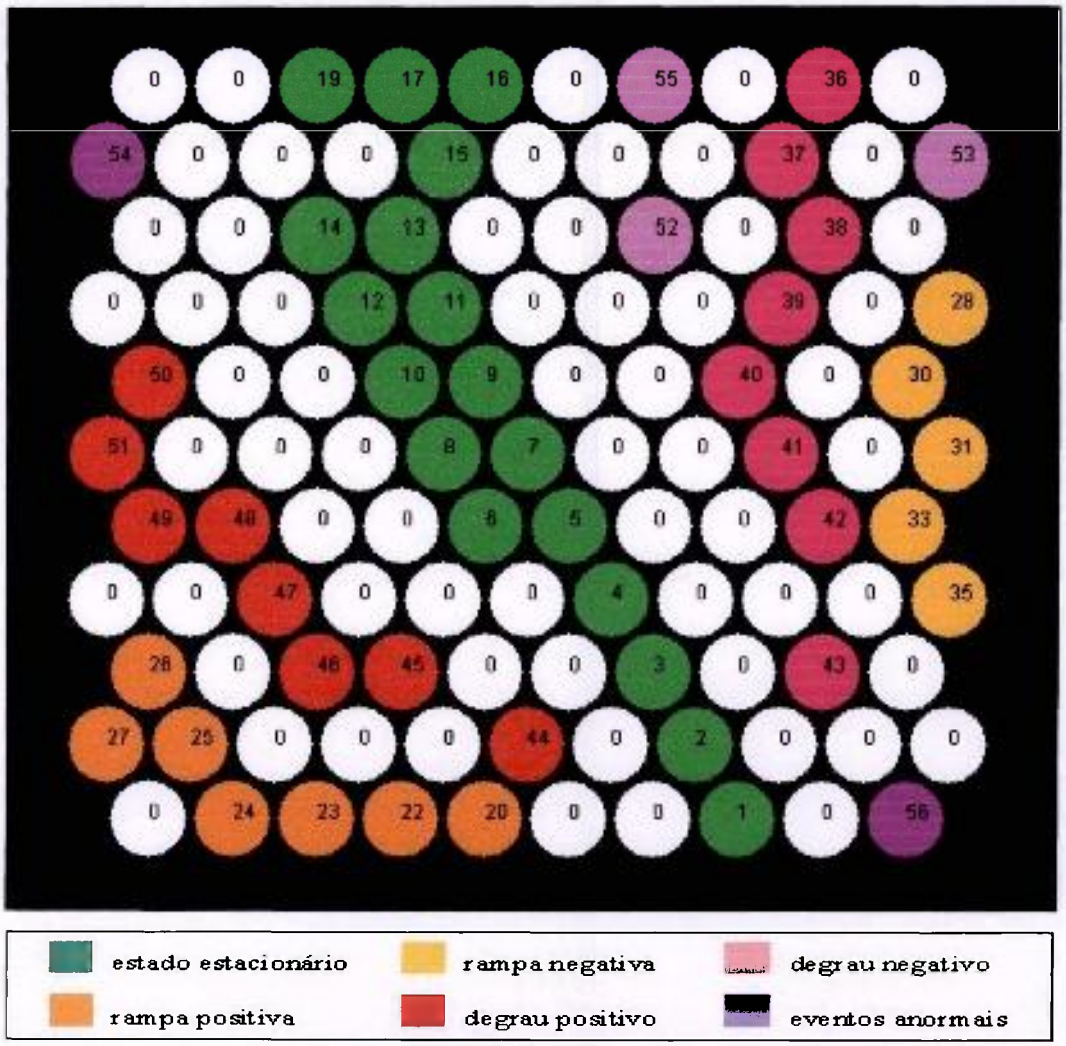


FIGURA 6.1 - Camada de saída do SOM exibindo o resultado do monitoramento da operação do IRIS sob diversas condições. Nesta visão, os números sobre as unidades neurais correspondem aos dados da TABELA 4.1 e da TABELA 4.2. Observar a seqüência sobre o mapeamento.

A FIGURA 6.2 mostra os resultados obtidos para outro sistema (Baptista e Barroso, 2003, 2004), com os dados de entrada gerados pelo mesmo simulador de regimes de operação do IRIS, e para a mesma quantidade de amostras (Barroso, 2003). Também os autores fizeram uso da camada de saída plana hexagonal para o SOM. Ao comparar com a FIGURA 6.2, verifica-se grande semelhança nos resultados.

Verificou-se que não há qualquer diferença de eficiência entre utilizar uma camada neural em rede bidimensional plana retangular ou hexagonal. Porém a rede hexagonal, por ter seus elementos mais compactados, apresenta uma visualização mais agradável para o usuário quando este necessita verificar a separação das classes. Um exemplo equivalente ao da FIGURA 6.2, mas para topologia plana retangular é exibido na FIGURA 6.3.

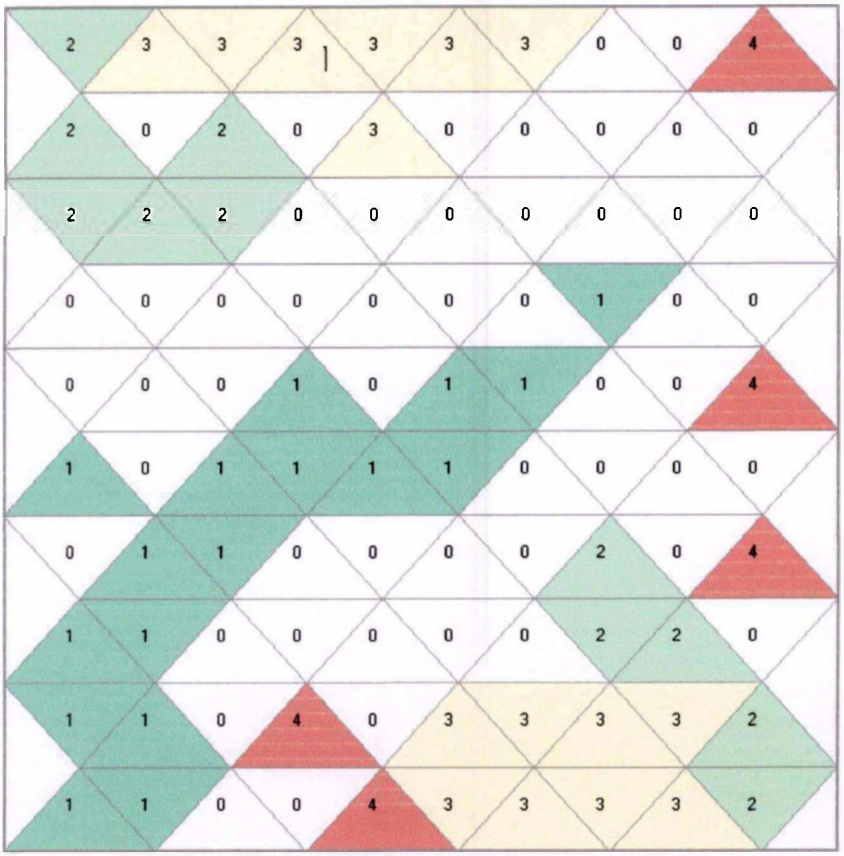


FIGURA 6.2 - Mapa de características para transientes do IRIS por Baptista e Barroso, 2003 e 2004. Label *1 ou verde-céu é para estado estacionário, 2 ou verde-claro é para rampa, 3 ou amarelo é para degrau e 4 ou vermelho é para transientes anormais.* η_0=0,7; σ_0=18; τ_1=700; τ_2=1000 para 4000 iterações.

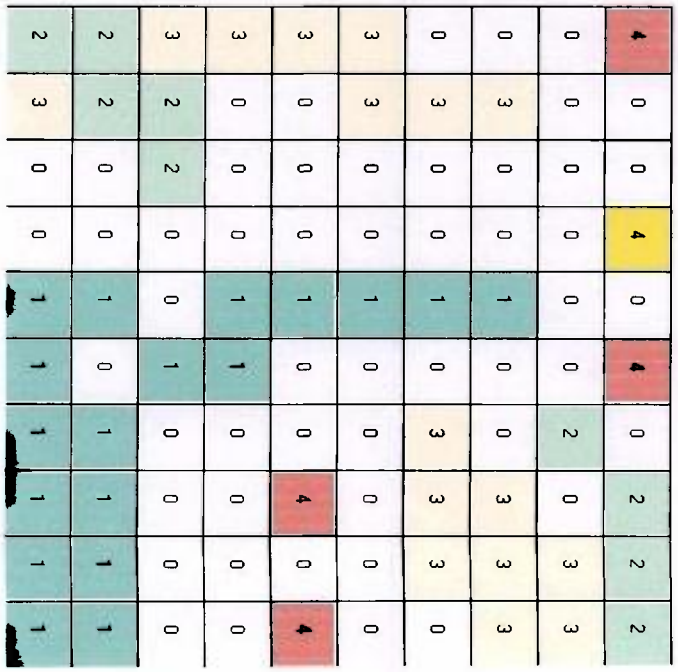


FIGURA 6.3 - Resultado equivalente à FIGURA 6.2, mas com camada de saída em grade retangular. Foi feita uma rotação de 90° para melhor comparação. η_0=0,1; σ_0=18; τ_1=350; τ_2=1000 para 2000 iterações.

6.1.2 Angra

Utilizando-se das entradas da TABELA 4.4, o SOM construído para este trabalho forneceu o mapeamento de eventos exibido na FIGURA 6.4, para rede com camada de saída bidimensional plana hexagonal de 93 unidades.

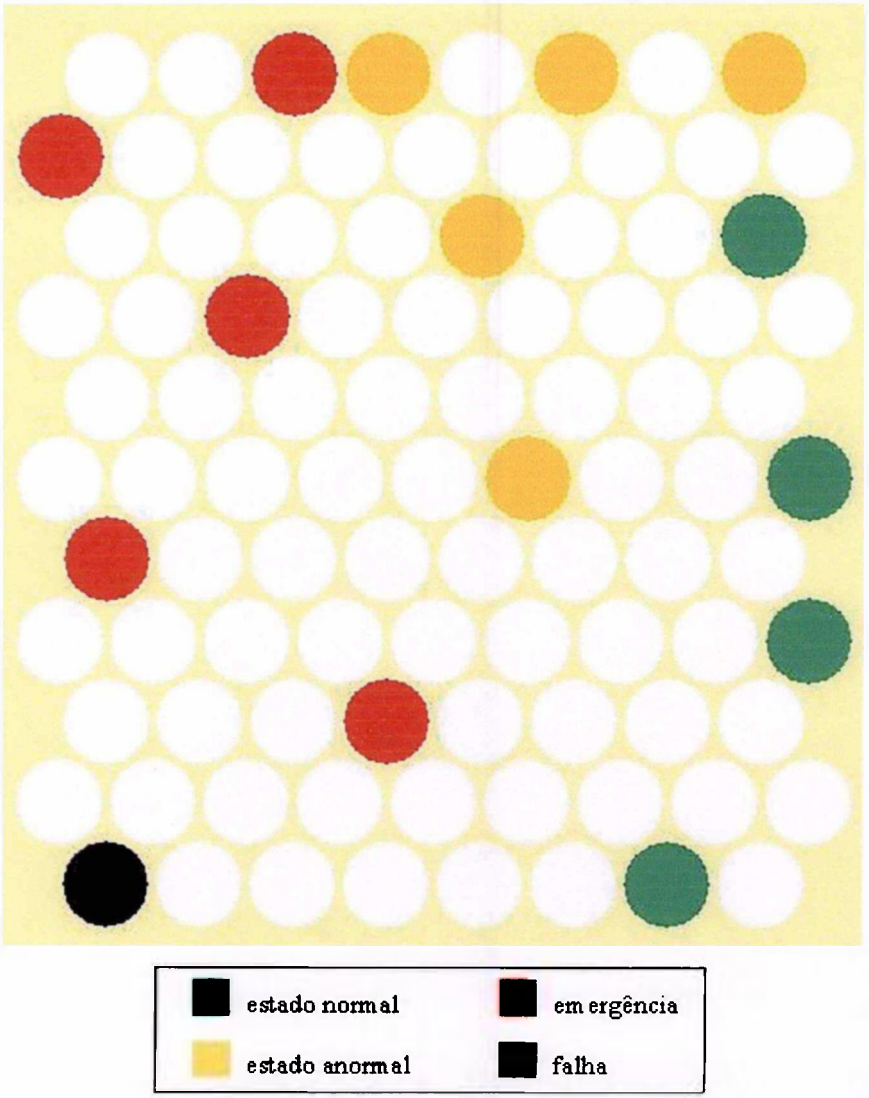


FIGURA 6.4 - Classificação de eventos pelo SOM em Java para o reator de Angra 2, onde verde indica transiente normal, laranja anormal, vermelho emergência e preto falha.

Este padrão foi comparado com o sistema mencionado em Baptista, 2002 (FIGURA 6.5) em rede retangular 10x10 unidades, mostrando mais uma vez compatibilidade nos resultados em separar eventos por regiões no mapeamento, e que a ferramenta proposta neste trabalho atinge a contento a classificação de padrões na Engenharia Nuclear.

Para obtenção deste padrão, utilizou-se taxa inicial de aprendizagem η igual a 0,25, e o tempo de processamento numa CPU de 1 GHz e 512 MB de RAM foi de aproximadamente 2 min., para ordenamento e convergência, chegando a 4000 iterações.

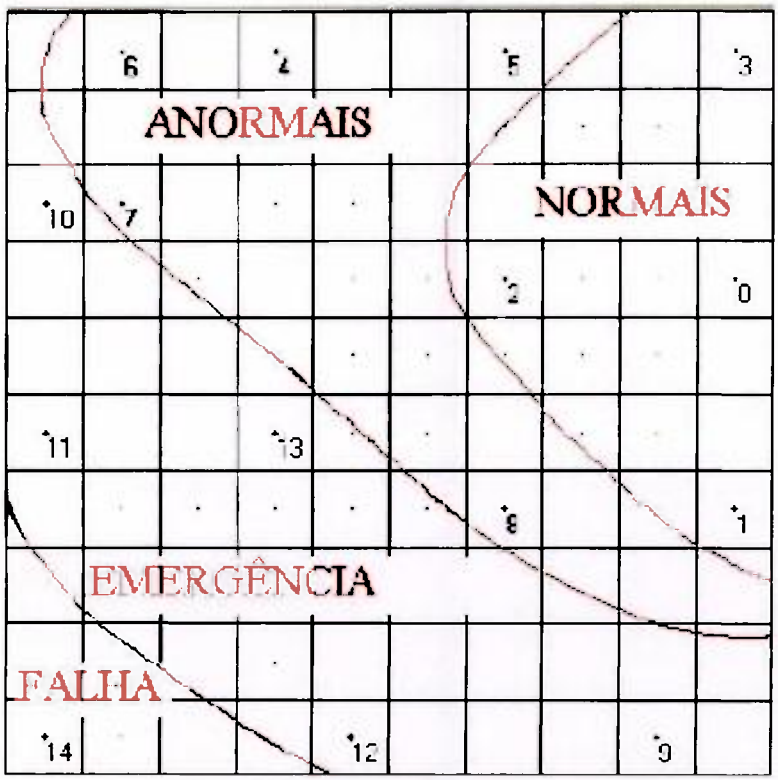


FIGURA 6.5 - Classificação de eventos segundo Baptista, 2003. Os dados de entrada foram os mesmos utilizados para o padrão obtido da FIGURA 6.4. η_0=0,9; σ_0=18; b_1=0,02; b_2=0,02.

6.2 Logística do gás natural

Para a primeira análise dos resultados, o conjunto de dados de entrada para os exemplos a seguir, tanto para treinamento como para teste, foi o consolidado de vários dias consecutivos e seguidos, de 8 de maio de 2003 a 2 de março de 2004 (FIGURA 6.6).

Para facilitar as análises seguintes, manteve-se fixa a escolha de valores ótimos (cor verde no painel). Para o ponto de medição 1 (City-Gate Capuava), manteve-se a escolha volume diário 5% acima de 2100000 m³, a 1 atm, 20°C e 9400 kcal/m³; PCS de entrega 0,5% acima de 9580 kcal/m³. Para o ponto 2 (City-Gate São José dos Campos), com as mesmas percentagens-limite, volume de 265000 m³ nas mesmas condições e PCS de 9545 kcal/m³. Abaixo de 80% destes volumes e de 99,5% dos PCSs, o valor foi considerado ruim (cor vermelha no painel). A faixa intermediária foi considerada boa ou regular (com laranja no painel). Uma unidade de saída é classificada de acordo com a

qualidade da medida que esteja mais próxima do peso da unidade. Ao se entrar com o índice do dado de entrada ou de teste, um alarme visual indica com qual unidade o dado possui mais afinidade (FIGURA 6.7).

Por motivo de segurança da informação, os dados de entrada não são os mais recentes devido à classificação de Reservado.

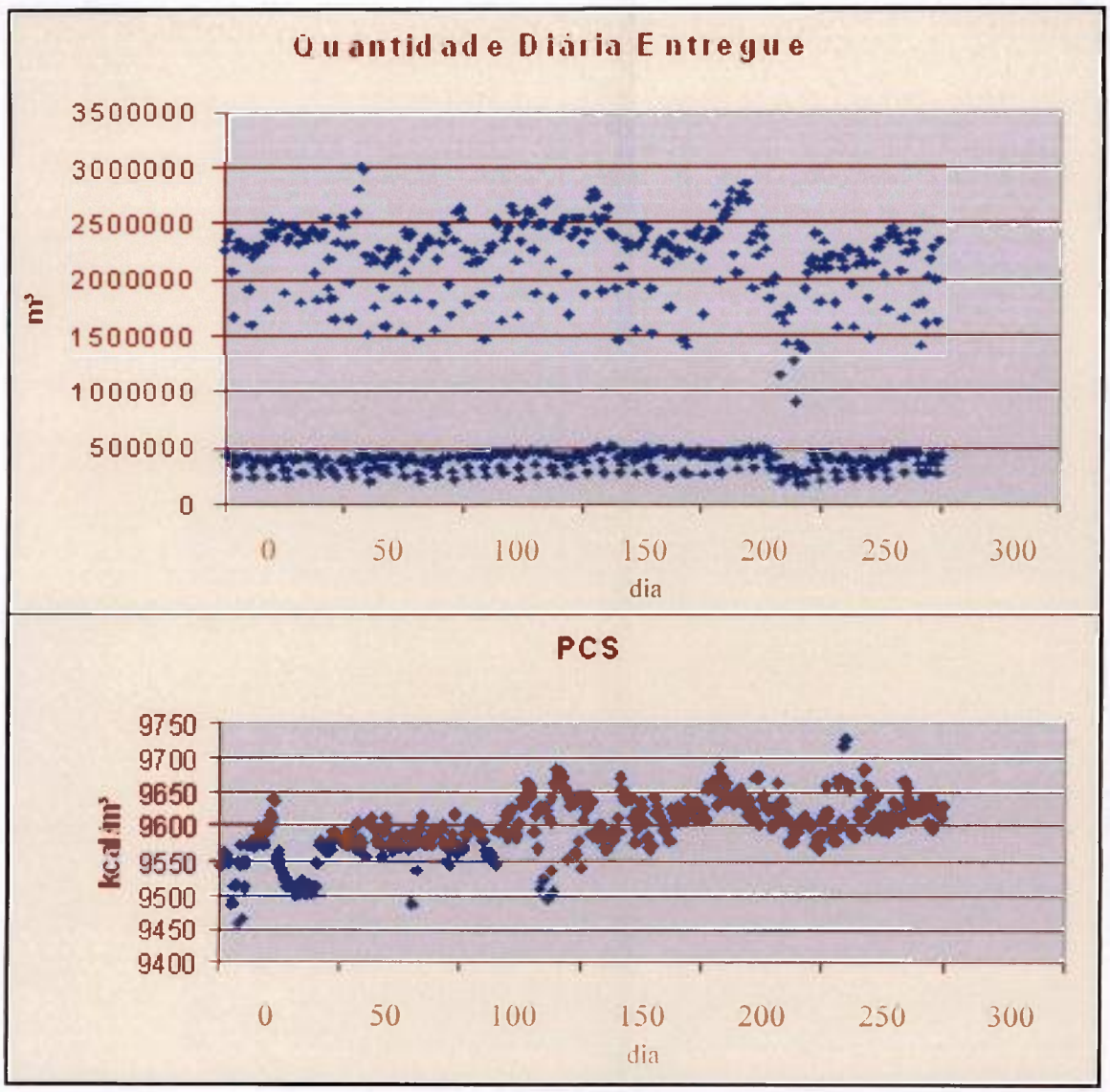


FIGURA 6.6 – Comparação entre os componentes volume entregue e PCS de amostra com 300 dias consecutivos, para dois pontos de medição, Capuava (Mauá-SP) e São José dos Campos (SP), de 8 de maio de 2003 a 2 de março de 2004. A perceptível queda de volume entre os dias 230 a 250 ocorreu na passagem de ano 2003/2004. No gráfico de quantidade diária entregue (QDE), percebe-se a distinção de regime dos dois pontos de medição, Capuava acima e São José abaixo, e, neste mesmo gráfico, a "camada" de dados logo abaixo dos regimes mais freqüentes de cada ponto de medição refere-se aos finais de semana.

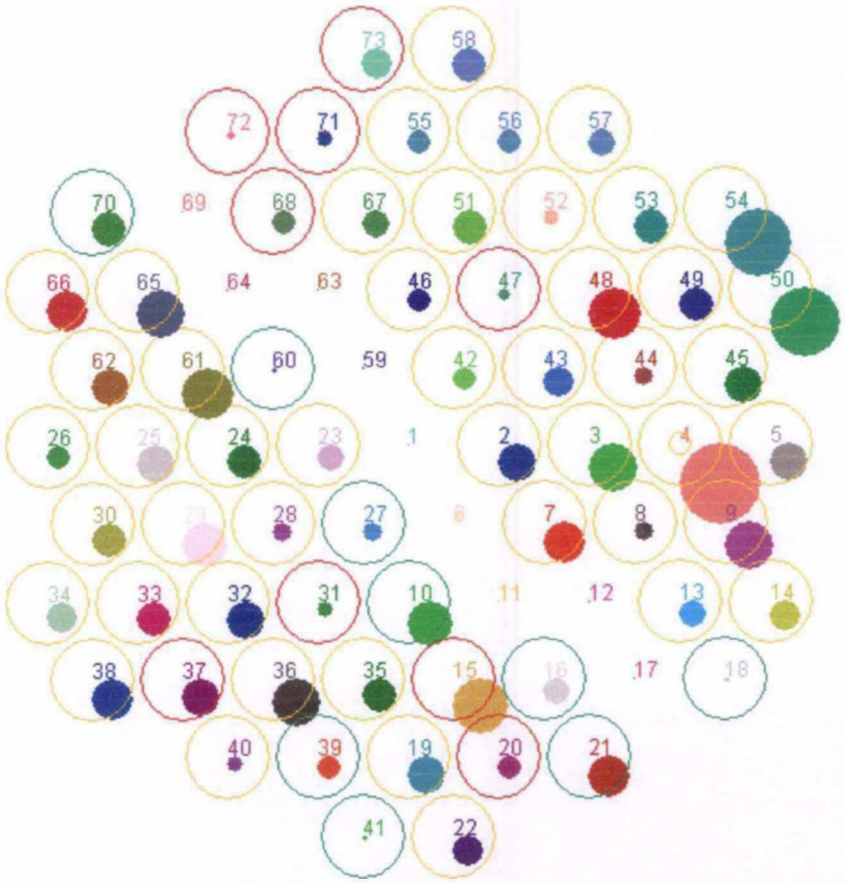


FIGURA 6.7 - Painel com o resultado da distribuição de 550 entradas para 73 unidades numa topologia plana hexagonal. A cor em torno das unidades significa classificação em dados ótimos (verde), bons (laranja) e ruins (vermelho) tendo como referência valores de contrato.

Vale mencionar a utilização de topologia bidimensional, porém esférica, para a camada de saída da rede de Kohonen. No exemplo da FIGURA 6.8, foram utilizadas 26 unidades de saída e os mesmos dados de treinamento do exemplo anterior. Ocorre uma separação clara os dois pontos de medição para até 350 entradas (aproximadamente 14 vezes o número de unidades), separação esta feita por um meridiano. No painel do *applet*, estão sendo representadas na parte superior as unidades projetadas do hemisfério frontal, e na parte inferior as unidades projetadas do hemisfério de trás. As unidades mais exteriores que se interseccionam pertencem ao meridiano que divide os hemisférios frontal e posterior.

A utilização de uma topologia esférica está baseada na observação anatômica do córtex cerebral, possuindo dezenas de circunvoluções, fora muitas outras estruturas elipsoidais e esféricas do cérebro. Os cálculos de distância se dão com os conceitos básicos de trigonometria esférica (ver ANEXO).

A topologia esférica evita o efeito das bordas que ocorre na topologia plana, em que as unidades da borda e próximas recebem menos influência da função vizinhança.

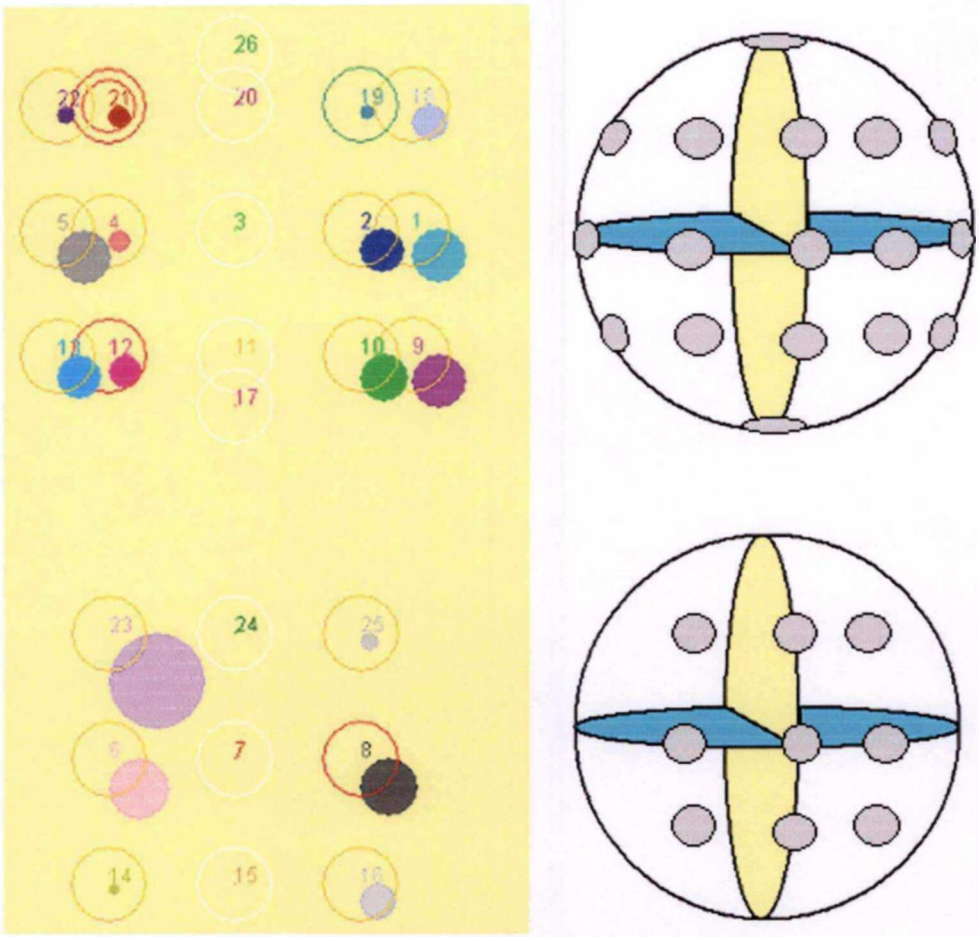


FIGURA 6.8 - Painel representando a rede esférica projetada no plano (à esquerda em fundo amarelo). Na parte superior, unidades frontais, na inferior, posteriores. Ao lado são apresentados desenhos de esferas como guias. Os círculos cinza representam a localização das unidades neurais. Alarme piscante (círculos concêntricos) na unidade 21.

Quanto às topologias bidimensionais planas, não há qualquer diferença de eficiência na obtenção de características ao utilizar uma camada neural em rede bidimensional retangular ou hexagonal. Porém a rede hexagonal, por ter seus elementos mais compactados, apresenta uma visualização mais agradável para o usuário quando este necessita verificar a separação das classes (FIGURA 6.9).

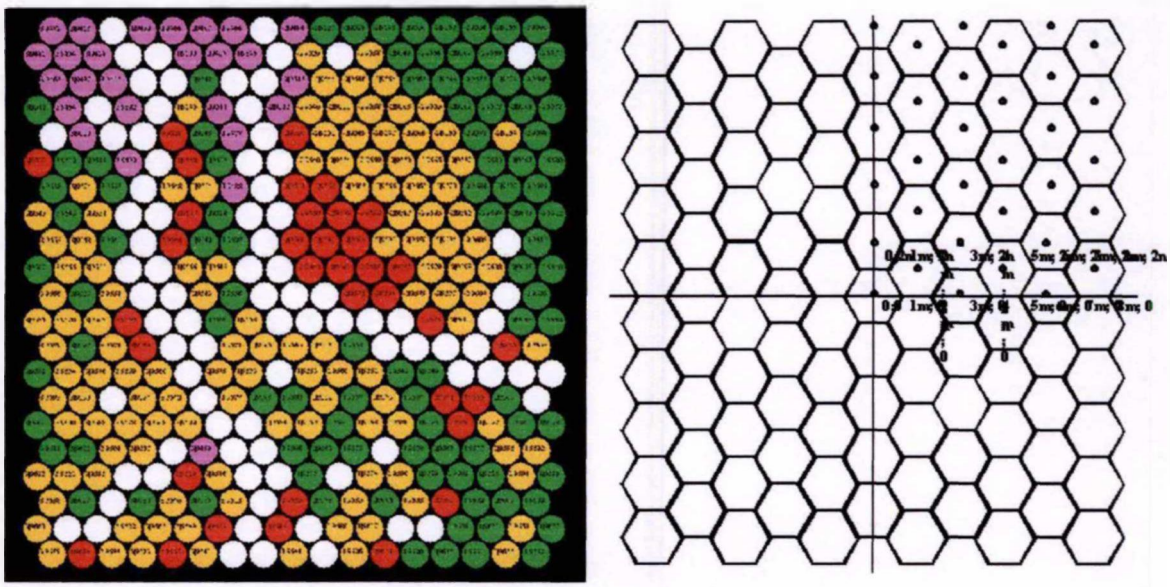


FIGURA 6.9 - Topologia bidimensional plana hexagonal para a rede de Kohonen.

Conforme pode ser observado na FIGURA 6.9, o mapeamento de características do ponto de medição na parte superior direita reflete uma menor dispersão para volume, PCS e demais variáveis, é o ponto de medição correspondente ao *label* –2 (TABELA 6.1). A unidade em roxo, isolada, na parte inferior esquerda, é um caso de PCS e volume baixos, simultaneamente.

TABELA 6.1 - Desvio padrão dividido pela média das variáveis envolvidas no treinamento, para cada ponto de medição, para amostras entre 22/out/2002 e 03/nov/2004.

Variável	Label -2	Label 1	Label 3
volume	0,163	0,183	0,171
PCS	0,003	0,004	0,006
pressão	0,011	0,022	0
temperatura	0,100	0,385	0
densidade	0,003	0,016	0,016

Nas três figuras seguintes são apresentados os resultados do treinamento para 750 entradas; 1000 iterações; $\eta_0 = 0{,}22$; $\sigma_0 = 55$; $C_R = 1{,}10$; $\zeta_{VOL} = 1$ e $\zeta_{PCS} = 50$, utilizando o mesmo conjunto de amostras (treinamento + teste) e a mesma quantidade de unidades neurais (367) da FIGURA 6.9.

A FIGURA 6.10, com a opção em fundo branco, apresenta a visão "número da unidade neural". Como não foi acionado o botão para preenchimento do círculo que tem em seu centro a unidade neural, as cores de classificação só estão na circunferência, permitindo ver a proporção de entradas distribuídas pelas unidades neurais (círculos cheios cinzas).

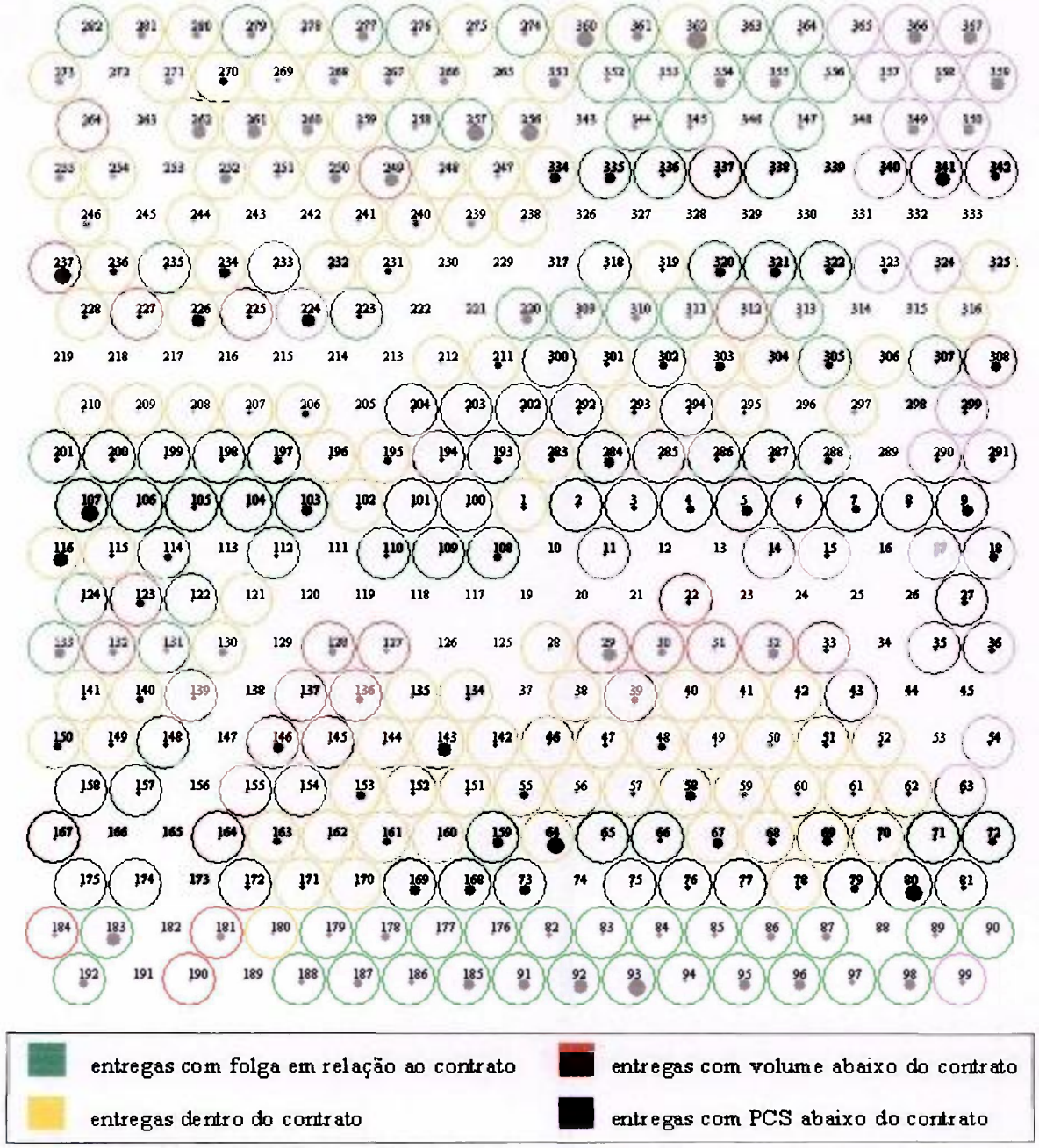


FIGURA 6.10 - Distribuição dos dados. 750 entradas; 1000 iterações; $\eta_0 = 0{,}22$; $\sigma_0 = 55$; $C_R = 1{,}10$; $\zeta_{VOL} = 1$ *e* $\zeta_{PCS} = 50$.

Na figura 6.11 é apresentada a visão "quantidade diária entregue", ou seja, sobre o círculo que representa a unidade neural é estampado o valor do componente volume (m^3) do vetor de entrada que está mais próximo do vetor peso desta unidade neural, dentre outros dados que tiveram esta unidade como vencedora .

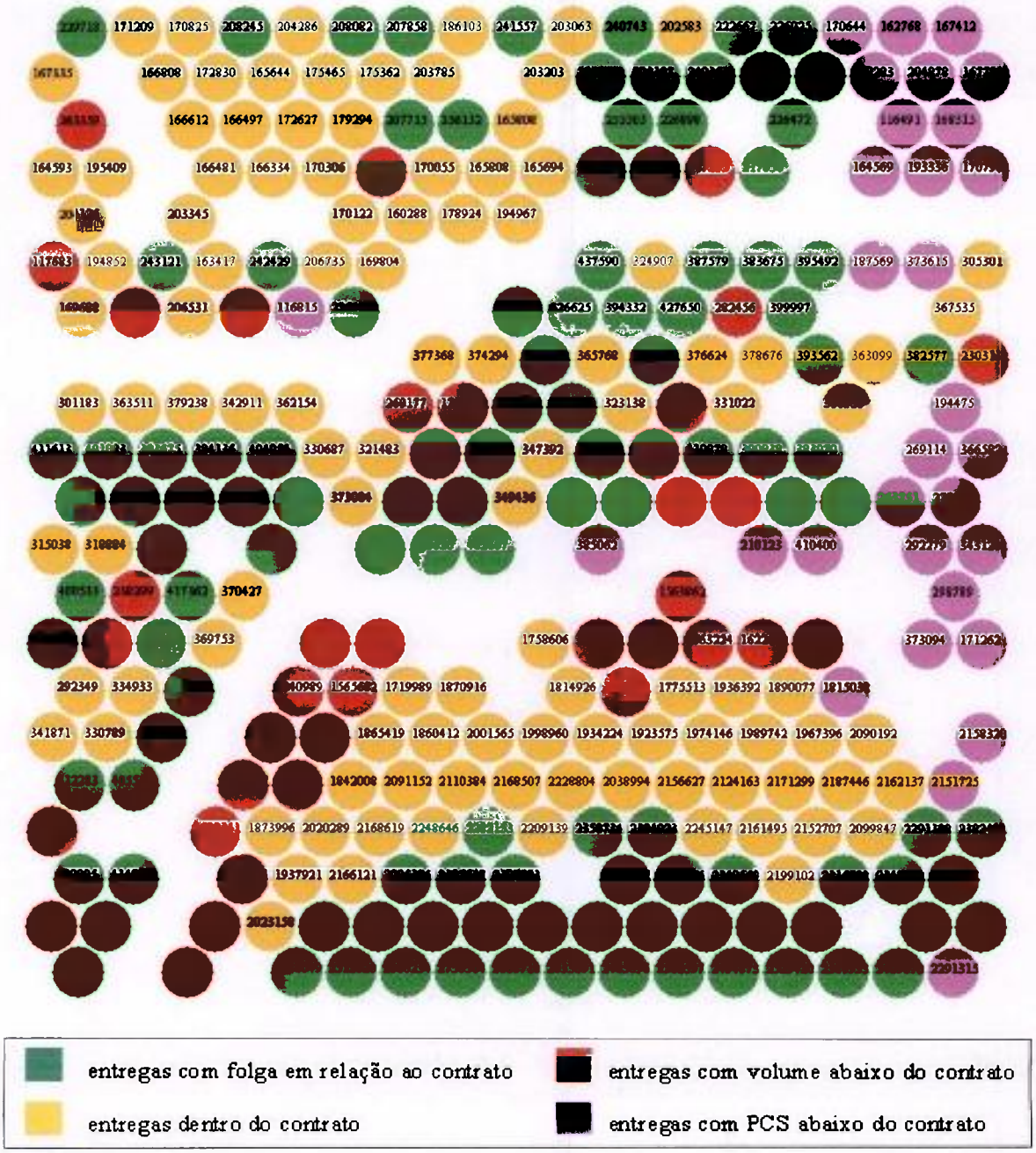


FIGURA 6.11 - Visão que imprime o volume do vetor de entrada que está mais próximo da unidade neural e que a classifica. 750 entradas; 1000 iterações; $\eta_0 = 0{,}22$; $\sigma_0 = 55$; $C_R = 1{,}10$; $\zeta_{VOL} = 1$ e $\zeta_{PCS} = 50$.

Na FIGURA 6.12 é apresentada a visão "*label* da unidade neural e PCS", em que é estampado sobre a figura que representa a unidade o valor de PCS do vetor entrada que está mais próximo do vetor peso da unidade neural.

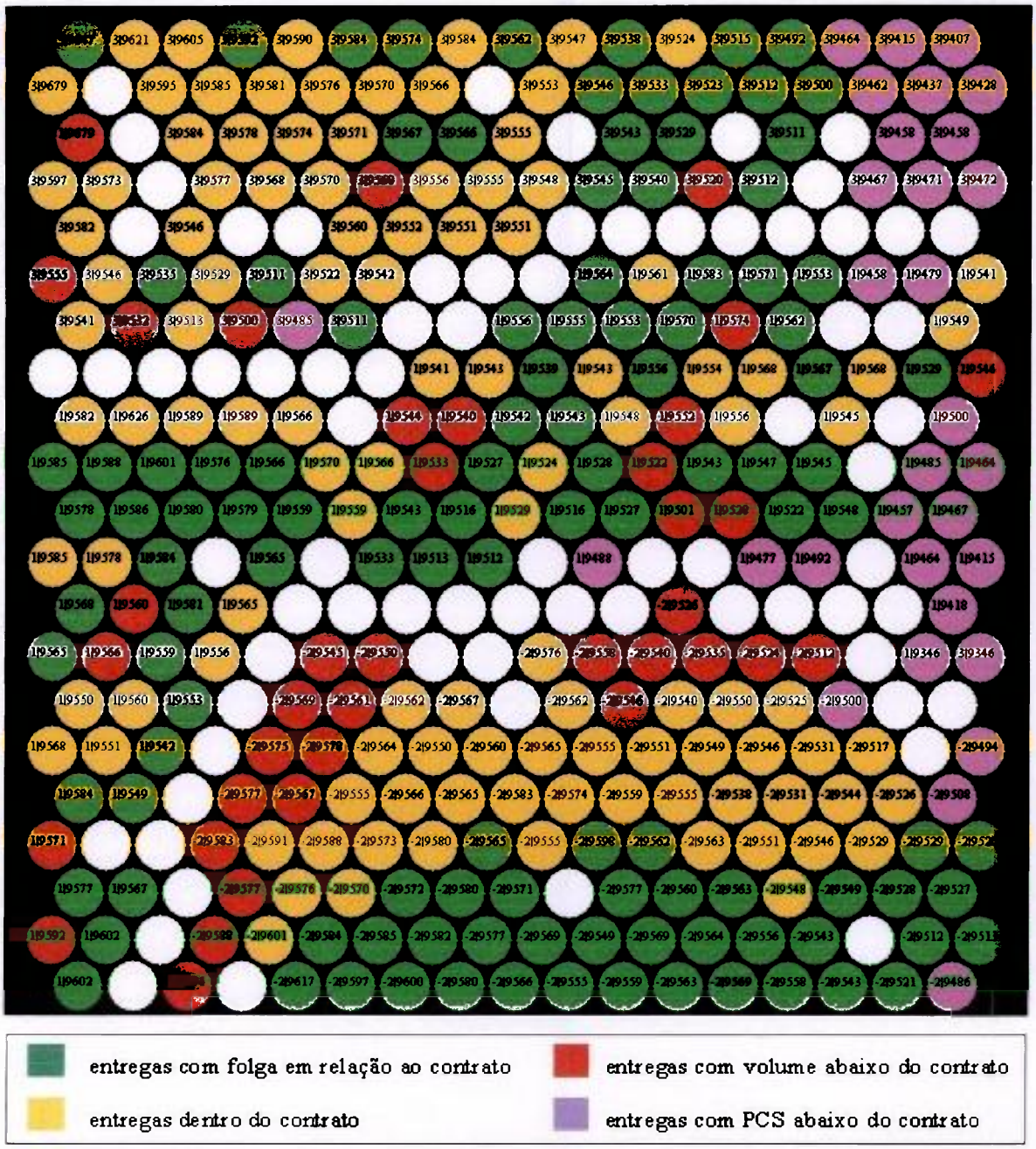


FIGURA 6.12 - Visão que imprime o PCS do vetor de entrada mais próximo da unidade neural e que a classifica. 750 entradas; 1000 iterações; $\eta_0 = 0{,}22$; $\sigma_0 = 55$; $C_R = 1{,}10$; $\zeta_{VOL} = 1$ *e* $\zeta_{PCS} = 50$.

7 CONCLUSÕES

7.1 Considerações finais

Os resultados deste e de outros trabalhos já publicados confirmam ser promissora a identificação de transientes através de mapas auto-organizáveis. Foram capazes de identificar tipos de transientes e condições de operação, separando nitidamente os transientes normais dos anormais e acidentais, mesmo sem recorrer à quantização vetorial, à retropropagação e ao mapa auto-organizável hierárquico.

Os mapas auto-organizáveis de Kohonen também se mostraram satisfatórios para a supervisão logística do gás natural mediante adaptações para o negócio, que por sua vez implicou em novas abordagens do próprio mapa auto-organizável, tais como topologias bidimensionais curvas para a rede de unidades neurais de saída e mudança de sensibilidade sobre os componentes de entrada. A interface do usuário construída para a aplicação se mostrou de boa usabilidade, com a própria rede servindo de painel de supervisão. Ao treinar com algumas centenas de entradas, a rede apresentou capacidade de generalização, pelos baixos desvios nos testes de classificação de dados, e apresentou robustez, por apresentar resultados semelhantes ao se variar a quantidade de unidades.

Apesar da plataforma ser 100% orientada a objetos, ou seja, um programa Java necessita obrigatoriamente da criação de objetos, seguiu-se um modelo estruturado de programação para o *core* do SOM, isso porque o algoritmo foi conseqüência de um longo processo evolutivo ocorrido durante este trabalho, iniciado com algoritmos adaptativos muito simples. A título de ilustração, em vários *softwares* de mercado, abertos ou proprietários, cada unidade neural é uma instanciação (objeto) de uma classe "UnidadeNeural", e dessa maneira ocorre a criação de objetos para outras entidades de uma rede neural. Após consolidação na produção do presente algoritmo, um *upgrade* poderá ser realizado para a construção de classes para cada entidade da rede neural.

Foi observado neste algoritmo um grande consumo de tempo para a mistura aleatória da ordem de entrada dos vetores de dados, que ocorre a cada iteração ou época, se rivalizando com o tempo dos cálculos. Isto se torna problemático acima de mil entradas, pois o crescimento do tempo de embaralhamento com o número de entradas não é linear. Não se obteve uma diminuição considerável deste tempo mesmo buscando otimização do

algoritmo de mistura e do uso da biblioteca Java. Uma pesquisa mais aprofundada para esta otimização deve ser levada adiante.

O programa possui um considerável módulo de tratamento dos dados. Trata os valores faltantes, inserindo uma média global, ou, ainda na falta desta, um valor típico. Por um dos componentes do vetor de dados variar pouco, mas devido a esta variação ser muito importante, recorreu-se ao artifício da sensibilidade diferenciada, à semelhança dos olhos que possuem maior sensibilidade ao vermelho e dos ouvidos quanto aos sons em torno de 2000 Hz. Seguindo o tratamento, como cada componente corresponde a grandezas diferentes, o algoritmo leva em conta o range de cada componente para que todos os valores de entrada se situem entre 0 e 1. Depois a soma dos componentes, à exceção do *label* de localização, é normalizado para 1, assim todos os componentes relativos ao processo de negócio chegam à rede com igual consideração. Como última etapa é feito um ajuste de escala para os valores se situarem entre -1 e 1, justificado para que os eixos de plotagem tenham como origem o centro de um quadro.

Não há diferenças de eficiência, em relação à extração de características, entre as topologias bidimensionais planas retangular e hexagonal, porém a hexagonal apresenta uma visualização mais agradável dos resultados por sua característica compacta. Procurando sanar o problema de que todas as unidades neurais não "vêem" a mesma vizinhança, foi lançada a inovação de se utilizar uma topologia bidimensional esférica, levando a uma ótima extração de características e boa classificação mesmo utilizando poucas unidades neurais, menos de 30. O ponto negativo encontrado nesta topologia foi uma maior dificuldade de visualização dos resultados sobre a rede neural. Para um número maior de unidades, vai ser obrigatoriamente necessário uma interface gráfica que trate de efeitos tridimensionais, o que implica em maior processamento da CPU ou do processador da placa de vídeo, se houver; e obviamente maior trabalho de programação sobre esta interface. Por outro lado, pode-se atribuir ao SOM com topologia bidimensional esférica o papel principal de entrada de uma rede com retropropação, nesse caso levando vantagem sobre as topologias planas quanto à classificação, como foi indicado aqui.

Problemas de performance foram encontrados com o uso muito continuado da aplicação, consumindo cada vez mais recurso de máquina, certamente uma incompatibilidade entre o compilador Java e o sistema operacional utilizado para os testes, no caso Windows 2000. Uma memória RAM acima de 512 MB é aconselhável para não ocorrer esta diminuição de performance no caso de uso prolongado. O *applet*, aplicado a este trabalho, é apontado na literatura como causador de problemas de desempenho no

hardware do cliente, uma vez que faz uso do processamento da sua máquina. Mas isto incorre em uma vantagem, livra o servidor do processamento, importante se o número de usuários simultâneos é grande. Para poucos usuários simultâneos e para uma supervisão classificada como crítica, o processamento deve ficar a cargo de um servidor de grande porte. Neste caso o código do *applet* deve ser passado para um conjunto *servlet*-páginas JSP-componentes JavaBeans.

7.2 Trabalhos futuros

É necessário continuar na busca de uma interface gráfica mais leve, como, por exemplo, uma possível integração entre Java, para cálculos, e Flash ou equivalente para interface gráfica.

O algoritmo SOM aqui apresentado deverá incorporar a quantização vetorial a fim de obter melhor definição na classificação das entradas (FIGURA 7.1).

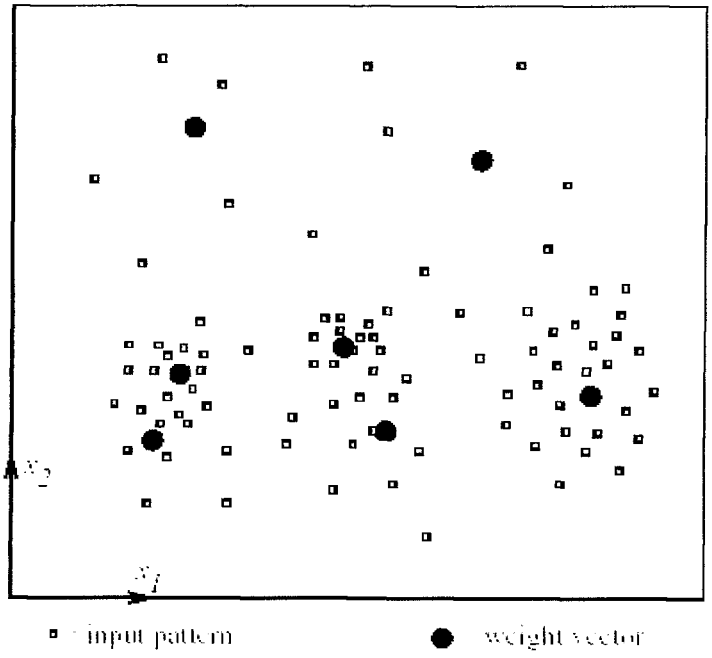


FIGURA 7.1 - Resultado de quantização vetorial.

A fim de se produzir previsões, o mapa auto-organizável servirá de entrada para outros tipos de redes neurais, tais como uma camada de propagação ou várias camadas de retropropagação (FIGURA 7.2).

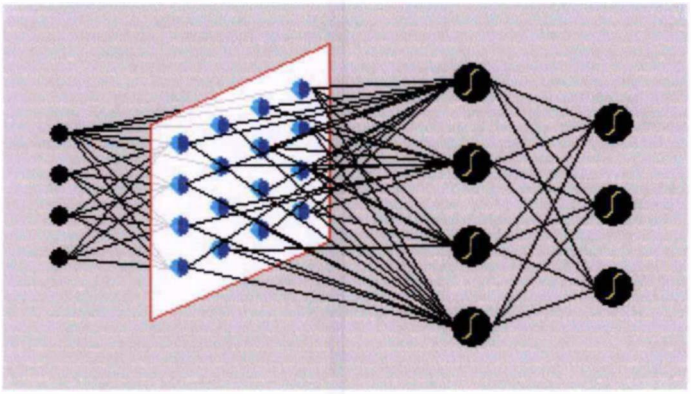


FIGURA 7.2 - O SOM como entrada de uma rede neural multicamada com retropropagação.

Uma maior levantamento da eficiência de topologias não planas, especialmente a esférica, vai ser levada adiante, inclusive por serem promissoras para servirem como entradas de outras redes neurais.

O uso efetivo em produção, para a supervisão de processos é iminente, inicialmente como mais uma opção de relatório ao lado dos gráficos de tendência, não direcionado somente para o negócio gás natural, mas para a geração de energia e proteção de dutos. Outro uso para produção será o acionamento de alarmes pelo SOM em painéis de supervisão de processos (FIGURA 7.3).

FIGURA 7.3 - Planta regional com algumas faixas de dutos alarmadas.

ANEXO – Trigonometria esférica

Grande círculo e pequeno círculo

Se um círculo construído sobre a superfície de uma esfera conter o centro dessa esfera, esse círculo é um grande círculo, caso contrário será um pequeno círculo.

Representação do vetor posição em coordenadas esféricas

As componentes do vetor posição **r** utilizando coordenadas esféricas (FIG. DE ANEXO 1) são:

$$\begin{aligned} r_x &= r.\cos\theta.\cos\varphi\,, \\ r_y &= r.\cos\theta.\operatorname{sen}\varphi\,, \\ \text{e}\quad r_z &= r.\operatorname{sen}\theta\,, \\ &\text{sendo } |\,\mathbf{r}\,| = r. \end{aligned} \qquad \text{(ANEXO.1)}$$

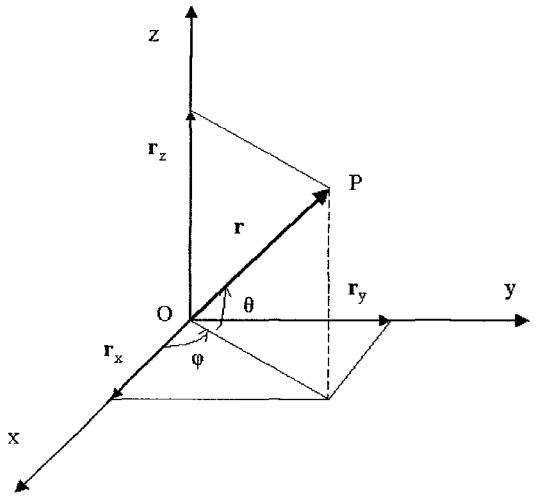


FIG. DE ANEXO 1 – Decomposição do vetor posição.

Ao fixar | r | = 1 u. m. (unidade de medida), elimina-se o grau de liberdade distância, só interessando as coordenadas θ e φ. Fixando todos os vetores posição para o mesmo módulo, significa que todos os pontos estarão sobre a superfície de uma esfera. E para o módulo igual a 1 obteremos a representação matricial de **r**:

$$\mathbf{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = \begin{pmatrix} \cos\theta . \cos\phi \\ \cos\theta . \operatorname{sen}\phi \\ \operatorname{sen}\phi \end{pmatrix}. \qquad \text{(ANEXO.2)}$$

Elementos do triângulo esférico

Através do produto escalar, o ângulo entre dois vetores pode ser relacionado com as coordenadas esféricas. Este ângulo é na verdade um dos ângulos diedros do triângulo esférico formado com o auxílio do eixo z (FIG. DE ANEXO 2), que possui como "lados" arcos de grandes círculos,

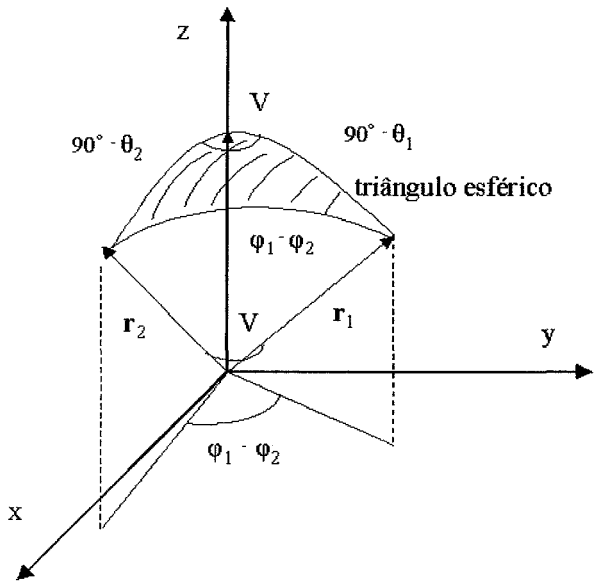


FIG. DE ANEXO 2 - Análise do triângulo esférico.

$$\mathbf{r}_1 . \mathbf{r}_2 = \cos V. \qquad \text{(ANEXO.3)}$$

Por outro lado,

$$\mathbf{r}_1 . \mathbf{r}_2 = \begin{pmatrix} r_{x1} & r_{y1} & r_{z1} \end{pmatrix} \begin{pmatrix} r_{x2} \\ r_{y2} \\ r_{z2} \end{pmatrix} = \qquad \text{(ANEXO.4)}$$

$$= \operatorname{sen}\theta_1 \operatorname{sen}\theta_2 + \cos\theta_1 \cos\varphi_1 \cos\theta_2 \cos\varphi_2 + \cos\theta_1 \operatorname{sen}\varphi_1 \cos\theta_2 \operatorname{sen}\varphi_2.$$

Pela propriedade da trigonometria plana,

$$\cos \varphi_1 \cos \varphi_2 + \operatorname{sen} \varphi_1 \operatorname{sen} \varphi_2 = \cos (\varphi_1 - \varphi_2) = \cos (\varphi_2 - \varphi_1). \qquad \text{(ANEXO.5)}$$

Combinando as equações ANEXO.3 e ANEXO.5:

$$\cos V = \operatorname{sen} \theta_1 \operatorname{sen} \theta_2 + \cos \theta_1 \cos \theta_2 \cos (\varphi_1 - \varphi_2)$$

ou (ANEXO.6)

$$\cos V = \cos (90^\circ - \theta_1) \cos (90^\circ - \theta_2) + \operatorname{sen} (90^\circ - \theta_1) \operatorname{sen} (90^\circ - \theta_2) \cos (\varphi_1 - \varphi_2).$$

Equações da trigonometria esférica

A dedução de (ANEXO.6) fornece o seguinte conjunto de equações para um triângulo esférico que utiliza símbolos para seus elementos conforme a FIG. DE ANEXO 3.

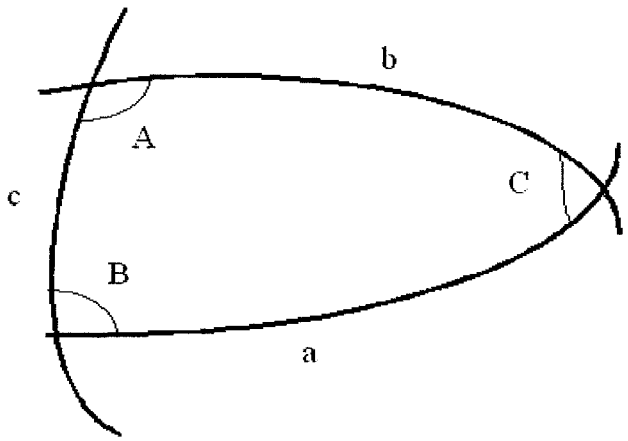


FIG. DE ANEXO 3 - Elementos do triângulo esférico.

$$\cos a = \cos b \cos c + \operatorname{sen} b \operatorname{sen} c \cos A, \qquad \text{(ANEXO.7a)}$$

$$\cos b = \cos a \cos c + \operatorname{sen} a \operatorname{sen} c \cos B, \qquad \text{(ANEXO.7b)}$$

$$\cos c = \cos a \cos b + \operatorname{sen} a \operatorname{sen} b \cos C. \qquad \text{(ANEXO.7c)}$$

Se as equações da série ANEXO.7 forem reordenadas como, por exemplo, sen b sen c cos A = cos a – cos b cos c, o segundo termo ao quadrado, de todas elas será: 1 – $\cos^2$a - $\cos^2$b - $\cos^2$c + 2 cos a cos b cos c. Então todos os primeiros termos poderão ser igualados, resultando na lei dos senos para o triângulo esférico,

$$\frac{\text{sen } A}{\text{sen } a} = \frac{\text{sen } B}{\text{sen } b} = \frac{\text{sen } C}{\text{sen } c}. \qquad \text{(ANEXO.8)}$$

Se agora, da equação (ANEXO.7a), for isolado num membro o cos b, e substituído na equação (ANEXO.7b) obtém-se a equação (ANEXO.9a); e fazendo o mesmo entre (ANEXO.7b) e (ANEXO.7c) e depois entre (ANEXO.7c) e (ANEXO.7a), obtém-se as equações (ANEXO.9b) e (ANEXO.9c) respectivamente, a terceira série de equações da trigonometria esférica,

$$\text{sen } b \cos A = \cos a \text{ sen } c + \text{sen } a \cos c \cos B, \qquad \text{(ANEXO.9a)}$$

$$\text{sen } c \cos B = \cos b \text{ sen } a + \text{sen } b \cos a \cos C, \qquad \text{(ANEXO.9b)}$$

$$\text{sen } a \cos C = \cos c \text{ sen } b + \text{sen } c \cos b \cos A. \qquad \text{(ANEXO.9c)}$$

Em todas as séries de equações da trigonometria esférica, os arcos de círculo (os "lados" do triângulo esférico) podem ser permutados pelos ângulos diedros (os "ângulos" do triângulo esférico), exemplo:

$$\cos a = \cos b \cos c + \text{sen } b \text{ sen } c \cos A \rightarrow$$
$$\cos A = \cos B \cos C + \text{sen } B \text{ sen } C \cos a. \qquad \text{(ANEXO.10)}$$

APÊNDICE A – RETROPROPAGAÇÃO, UMA APLICAÇÃO SIMPLES

A seguir será apresentado o código-fonte de uma aplicação simples de retropropagação (FIG. DE APÊNDICE A 1), ilustrando o subtítulo 3.1. O algoritmo resolve o OU-EXCLUSIVO com apenas dois passos, consistindo de neurônios RHW - Rummelhart, Hinton e Williams; é basicamente um ADALINE, de Widrow, seguido de uma função de ativação $g(v)$ (Kovács, 1998).

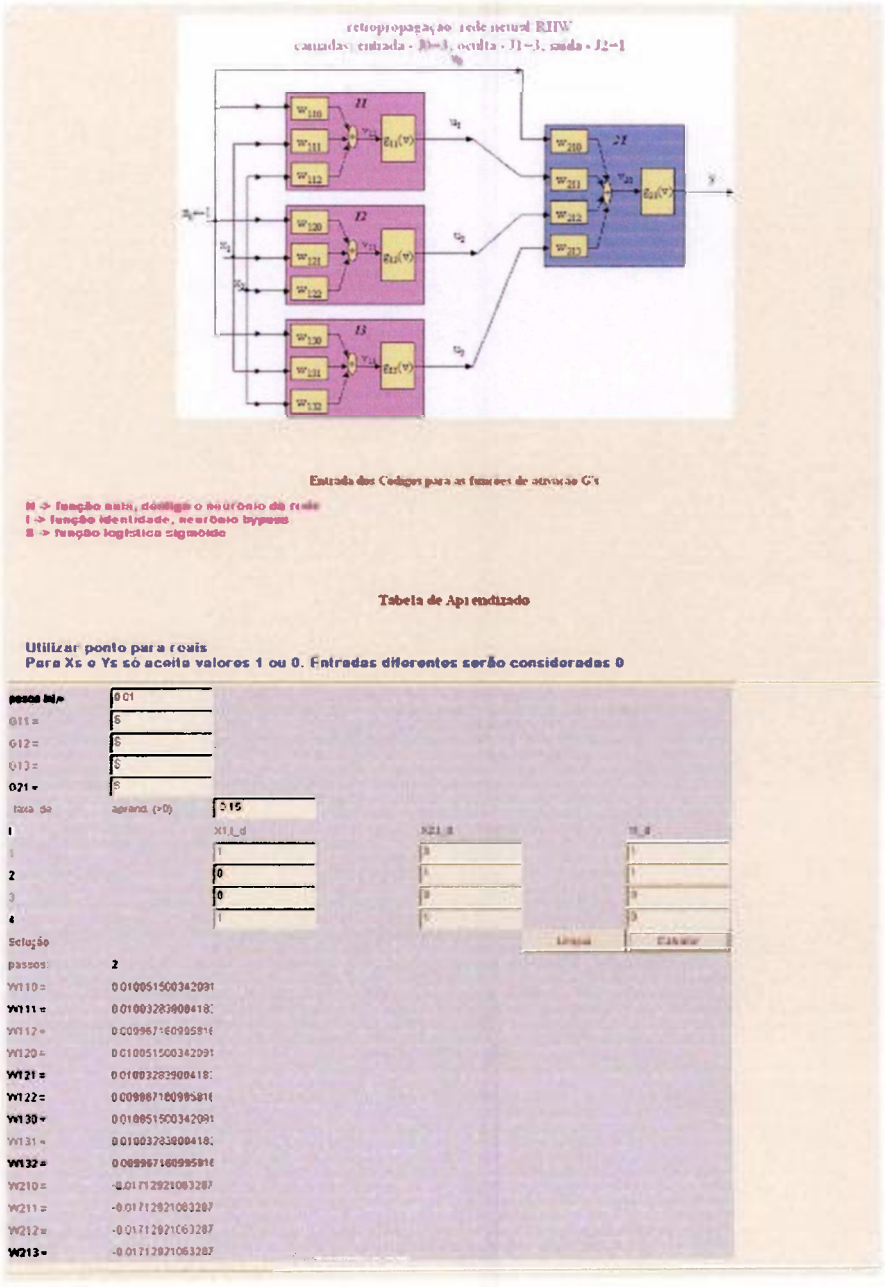


FIG. DE APÊNDICE A 1 - Aplicação Java que utiliza o algoritmo RHW.

Rummelhart4.html

```
<HTML>
<HEAD>

<TITLE>Project Rummelhart.jpx</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!-- Inicio do script netie
function netie(net, ie) {
  if ((navigator.appVersion.substring(0,3) >= net && navigator.appName
  == 'Netscape' && net != -1) || (navigator.appVersion.substring(0,3)
  >= ie && navigator.appName.substring(0,9) == 'Microsoft' && ie !=
  -1))
    return true;
else return false;
}
//  Fim do script netie -->
</SCRIPT>
<SCRIPT LANGUAGE="JavaScript">
<!-- Inicio do script transitionTxt
function transitionTxt() {
  if (idTransTxt.filters.item(0).status == 0) {
    idTransTxt.filters.item(0).apply();
    idTransTxt.style.visibility = "visible";
    idTransTxt.filters(0).play();
  }
  if (idTransTit.filters.item(0).status == 0) {
      idTransTit.filters.item(0).apply();
      idTransTit.style.visibility = "visible";
      idTransTit.filters(0).play();
  }
}
//  Fim do script transitionTxt -->
</SCRIPT>

</HEAD>



<BODY onLoad="netie(-1,4)?transitionTxt():null">

<H4 ><FONT color=#5A4033>By Waldemar Doraskevi&#269;ius
Jr    n<SUP>o</SUP> 198639</FONT><H4>

<BR>
<DIV ID="idTransTit" align = "left" STYLE=
"position:relative; left:20; height:40; width:700; color:#3322d2;
font-size:12; font-family:Arial Rounded MT Bold ;
filter:revealTrans(Duration=7, Transition=70); visibility:hidden">
                Melhor visualizado em tela 1024x786 pixels
</DIV>


<table width="100%" align="center">

    <tr>
        <td  align="center">
          <IMG SRC="NeuroRummelhart1.jpg">
        </td>
    </tr>

</table>


<H3 align=center><br> <br>Entrada dos C&oacute;digos para as
fun&ccedil;&otilde;es de ativa&ccedil;&atilde;o G's</H3>

<BR>
<DIV ID="idTrans" align = "left" STYLE=
"position:relative; left:20; height:40; width:700; color:#FF0000;
font-size:13; font-family:Arial Rounded MT Bold ;
filter:revealTrans(Duration=7, Transition=70)">
            N -> função nula, desliga o neurônio da rede<BR>
            I -> função identidade, neurônio bypass<BR>
            <!--T -> função logística tangente hiperbólica<BR>-->
```

```
63                  0 -> função logística sigmóide<BR>
64  </DIV>
65
66
67
68
69  <H2 align=center><br> <br>Tabela de Aprendizado</H2>
70
71  <BR>
72  <DIV ID="idTransTxt" align = "left" STYLE=
    "position:relative; left:20; height:40; width:700; color:#3322d2;
    font-size:15; font-family:Arial Rounded MT Bold ;
    filter:revealTrans(Duration=7, Transition=70); visibility:hidden" >
73                  Utilizar ponto para reais<BR>
74                  Para Xs e Ys só aceita valores 1 ou 0. Entradas
                    diferentes serão consideradas 0
75  </DIV>
76
77
78
79
80
81
82
83
84  <HR>
85
86
87
88  <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT"><!--
89
90
91  document.write("<Applet name='apple' code= 'Rummelhart4.class'  width
    = 800   height = 650>")
92
93  document.write("</applet>")
94
95  //--></Script>
96
97
98  <!-- este value é o número de sinapses, exceptuando o referente ao
    parâmetro de corte (thera) -->
99
100
101
102 <HR>
103
104
105
106
107
108 </BODY>
109 </HTML>
```

Rummelhart4.java

```
/*****************************************************************
*******************
 ***
 ***
 ***  Produced and created by W. Doraskevicius, Jr
waldj@petrobras.com.br      ***
 ***
 ***
 *** Rede neural RHW com duas camadas neurais: 3 neurônios na oculta e
1 na de saída ***
 ***          Retropropagação por cada linha da tabela de amostra
(sequencial)          ***
 *****************************************************************
 ******************/

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.*;
import java.text.DecimalFormat;

public class Rummelhart4 extends Applet implements ActionListener {

        String NAMEINI[] = {"pesos ini.=","G11 =","G12 =","G13 =",
        "G21 =","W4inicial=","W5inicial=","W6inicial="};
        String NAMESOL[] = {"pesos:","W1_0 =","W1_1 =","W1_2 =",
        "W1_20 =","W121 =","W122 =","W130 =","W131 =","W132 =","W210 =",
        "W211 =","W212 =","W213 =",};
        String NAMECOL[] = {"","X1,1_d","X2,1_d","X3,1_d","X4,1_d",
        "X5,1_d","X6,1_d"};

        Label seqAmostras;
        Label solucao[];

        TextField apre;
        TextField fields[];
        TextField paramFunc[];

        Button calcButton;
        Button limpButton;
        int gl=0, gc=0,j = 1,k = 1,l =1, n= 0, m=0, p=0;
        int size = 2;
        int amostras = 1;

        String firstNumber;


        public void init () {

         for (int i=0; i<size; i++){
           amostras=amostras*2;
         }

         paramFunc = new TextField[size+3];
         for (int i=0; i<= (size+2); i++) {paramFunc[i] = new TextField(
         "");}

         apre = new TextField(" ");

         fields = new TextField[amostras*(size+1)];
         for (int i = 0; i < ((size+1)*amostras); i++) {fields[i] = new
         TextField("");}

         solucao = new Label [4*(size+1)+2];
         for (int i=0; i < (4*(size+1)+2); i++) {solucao[i] = new Label
         ("");}

         gl = 26;
         gc = 2*size+3;


         //Montagem da interface gráfica
         setLayout (new GridLayout (gl, gc));
```

```
60
61          for(int m=0; m<=(size+2); m++) {
62            add(new Label(NAMEINI[m]));
63            add(paramFunc[m]);
64            for(int p=0;p<(gc-2);p++) {
65              add(new Label(""));}
66            }
67
68            add(new Label("  taxa  de "));
69            add(new Label("aprend.  (>0)"));
70            add(abre);
71            for(int m=0; m < (gc - 3); m++) {add (new Label (""));}
72
73            for(int m=0; m < (size+1); m++) {
74              add(new Label(NAMECOL[m]));
75              add(new Label(" "));}
76              add(new Label("Y1_d"));
77
78              p = 0;
79              for(int i=1; i <= amostras; i++) {
80                add(seqAmostras = new Label ("0"));
81                seqAmostras.setText(String.valueOf(i));
82                for(int m=0; m<(size+1); m++) {
83                  add(new Label(" "));
84                  add (fields[m+p]);}
85                  p=p+size+1;
86                }
87
88                add (new Label ("Solução:"));
89                for(int i=0; i < (gc - 3); i++) {add (new Label ("" ));}
90
91                limpButton = new Button ("Limpar");
92                add (limpButton);
93
94                calcButton = new Button ("Calcular");
95                add (calcButton);
96
97                for(int i=0; i < (4*(size+1)+2); i++) {
98                 add (new Label (NAMESOL[i]));
99                 add (solucao[i]);
100               for(int m=0; m < (gc - 2); m++) {add (new Label (""));}
101           }
102
103           paramFunc[0].requestFocus();
104
105           calcButton.addActionListener(this);
106           limpButton.addActionListener(this);
107
108       }// fim do init()
109
110
111
112
113       //Evento clica botão
114       public void actionPerformed(ActionEvent evt) {
115
116         Button source = (Button)evt.getSource();
117         if (source.getLabel().equals("Calcular")) {
118           calcSistema();
119         }
120         if (source.getLabel().equals("Limpar")) {
121           for (int i=0; i < ((size+1)*amostras); i++) {
122             fields[i].setText("");
123             fields[0].requestFocus();
124           }
125         }
126
127       }
128
129
130       //Declara e instancia funções
131       public static double g (String param, double x) {
132         int p;p=0;
```

```
133           if (param.equals("N")) p=0;
134           if (param.equals("I")) p=1;
135           if (param.equals("S")) p=2;
136           if (param.equals("T")) p=3;
137           switch (p){
138             case 0:
139               return 0.00d;
140             case 1:
141               return x;
142             case 2:
143               double BETAS=0.75d;
144               return 1.000d/(1.00d + Math.exp((-1.00d)*BETAS*x));
145             default:
146               return 99999.999d;
147           }
148         }
149
150
151         public static double dg (String param, double x) {
152           int p;p=0;
153           if (param.equals("N")) p=0;
154           if (param.equals("I")) p=1;
155           if (param.equals("S")) p=2;
156           if (param.equals("T")) p=3;
157           switch (p) {
158             case 0:
159               return 0.00d;
160             case 1:
161               return 1.0d;
162             case 2:
163               double BETAS=0.75d;
164               return BETAS*(1.0d - g(param,x))*g(param,x);
165             default:
166               return 99999.999d;
167           }
168         }
169
170
171         //Procedimento de cálculo desta classe
172         public void calcSistema() {
173
174         //Declarando e instanciando variáveis e constantes próprias da
            classe de cálculo
175         double x0_d;
176         double x_d[][];
177         double y_d[];
178         double u_d[][];
179         double w[][];
180         String param[];
181         double Dw;
182
183         double v[][];
184         double vMinus;
185         double y[];
186
187         double ETA; //fator de realimentação ou cte. de aprendizado
188         double dif;
189         double S;
190         double difex[][];
191
192         double argTheta=0.00d;
193         double heavySide=0.00d;
194
195         int indicadorLoop;
196         int passo = 0;
197
198         y= new double[amostras+1];
199
200         x0_d=-1d;
201         ETA=0.5d;
202         Dw=0.0d;
203
204         x_d = new double[amostras+1][size+1];
```

```
205         y_d = new double[amostras+1];
206         u_d = new double[amostras+1][size+2];
207
208         param = new String[size+3];
209         v = new double[amostras+1][3+1];
210         w = new double[size+3][size+2];
211         difex = new double[5][5];
212
213         //Limpa as soluções anteriores
214         for (int i=0; i < (4*(size+1)+2); i++) {
215           if (i==0) solucao[i].setText("Aguarde...");
216           else solucao[i].setText("#");
217         }
218
219         //Verifica e corrige consistência da tabela de treinamento
220         for (int i=1; i <= (size+2); i++) {
221           if ( paramFunc[i].getText().equals("X") || paramFunc[i].
              getText().equals("I") || paramFunc[i].getText().equals("S"))
              {
222             System.out.print(
                "Entrada da sequencia de escolha das funcoes");
223             System.out.print(i+1);
224             System.out.println(" Ok");
225           }
226           else {paramFunc[i].setText("S");}
227         }
228
229         for (int i=0; i < ((size+1)*amostras); i++) {
230           if ( fields[i].getText().equals("+1") || fields[i].getText().
              equals("1") || fields[i].getText().equals("0") ) {
231             System.out.print("Entrada de dados da sequencia ");
232             System.out.print(i+1);
233             System.out.println(" Ok");
234           }
235           else {fields[i].setText("0");}
236         }
237
238
239         //Preenchimento default dos vectores
240         for (int i=0; i <= (size+1); i++) {param[i] = "X";}
241         param[size+2] = "S";
242
243         for (int i=0; i < 5; i++) {
244           for (int j=0; j < (4); j++) {
245             w[i][j]=0.08d;
246           }
247         }
248
249         for (int i=0; i <= amostras; i++) {
250          for (int j=0; j <= (size); j++) {x_d[i][j] = 0.0d;}
251            y_d[i] = 0.0d;
252          }
253          for (int i=0; i <= amostras; i++) {
254            for (int j=0; j <= (size); j++) {u_d[i][j] = 0.0d;}
255          }
256          for (int i=1; i<=amostras; i++){
257            x_d[i][0]=-1.0d;
258          }
259          for (int i=1; i<=amostras; i++) {
260            u_d[i][0]=-1.0d;
261          }
262          for (int i=0; i < 5; i++) {
263            for (int j=0; j < 5; j++) {difex[i][j] = 0.0d;}
264          }
265
266         //Pega os valores da interface do usuário
267         for (int i=0; i < 5; i++) {
268           for (int j=0; j < (4); j++) {
269             w[i][j]=(Double.valueOf(paramFunc[0].getText())).
              doubleValue();
270           }
271         }
272
```

```
273         for (int j=1; j <= (size+2); j++) {
274           param[j] = (paramFunc[j].getText());
275         }
276
277         k=0;
278         for (int i=1; i <= (amostras); i++){
279           for (int j=1; j <= (size); j++) {
280             x_d[i][j] = (Double.valueOf(fields[k].getText())).
              doubleValue();k++;
281           }
282           y_d[i] = (Double.valueOf(fields[k].getText())).doubleValue();
              k++;
283
284
285         //subtrai da média
286         for (int i=1; i <= (amostras); i++) {
287           for (int j=1; j <= (size); j++) {
288             if (x_d[i][j] == 1d) {x_d[i][j]=1d;}
289             else {x_d[i][j]=-1d;}
290           }
291         }
292
293         //faz distâncias de 0 e 1 para y_d não simétricas
294         for (int i=1; i <= (amostras); i++) {
295           if (y_d[i]==1d) {y_d[i]=0.98d;}
296           else {y_d[i]=0.01d;}
297         }
298
299         //pega a constante de aprendizado
300         ETA = (Double.valueOf(apre.getText())).doubleValue();
301         if (ETA < 0.00000) {//consistência da cte. de aprendizado
302           ETA=ETA*(-1.0d); apre.setText(String.valueOf(ETA));
303         }
304
305         //Realizando propriamente o algoritmo de retropropagação da
          rede neural
306         int I=1;
307         indicadorLoop=4;
308
309         while (indicadorLoop > 0 && passo < 4000) {
310
311           for (int k=1; k<=3; k++) {
312             v[I][k]=0.0d;
313             for (int j=0; j<=(size);j++) {
314               v[I][k]=v[I][k] + (w[k][j]*x_d[I][j]);
315             }
316             u_d[I][k]=g(param[k],v[I][k]);
317             if (u_d[I][k]>=0.5d) {u_d[I][k]=1d;}
318             else {
319              if (u_d[I][k]==0d) {u_d[I][k]=0d;}
320              else u_d[I][k]=-1d;
321             }
322           }
323
324           //para atualizar os pesos da camada de saída
325           vMinus=0.0d;
326           for (int j=0; j<=(size+1);j++) {
327             vMinus=vMinus + (w[4][j]*u_d[I][j]);
328           }
329
330           y[I] = g(param[4],vMinus);
331
332           difex[4][I] = (y_d[I]-y[I])*dg(param[4],vMinus);
333
334           for (int n=0; n<=(size+1); n++) {
335             Dw=0.0d;
336             Dw = Dw + difex[4][I]*u_d[I][n];
337             Dw = ETA*Dw;
338             w[4][n] = w[4][n] + Dw;
339           }
340
341           //recalcula o delta da camada de saída
342           vMinus=0.0d;
```

```
343             for (int j=0; j<=(size+1);j++) {
344               vMinus=vMinus + (w[4][j]*u_d[I][j]);
345             }
346
347             y[I] = g(param[4],vMinus);
348
349             difex[4][I] = (y_d[I]-y[I])*dg(param[4],vMinus);
350
351             //para atualizar os pesos da camada oculta
352             for (int N=1; N<=(size+1); N++) {
353               difex[N][I] = difex[4][I]*w[4][N]*dg(param[N],v[I][N]);
354             }
355
356             for (int Y=1; Y<=(size+1); Y++) {
357              for (int n=0; n<=(size); n++) {
358                Dw=0.0d;
359                Dw = Dw + difex[Y][I]*x_d[I][n];
360                Dw = ETA*Dw;
361                w[Y][n] = w[Y][n] + Dw;
362              }
363             }
364
365             indicadorLoop=0;
366             for (int i=1; i<=amostras; i++) {
367              if ((y_d[i] >= 0.85d && y[i] > 0.5d) || (y_d[i] <= 0.15d &&
               y[i] < 0.5d)) {indicadorLoop=indicadorLoop+0;}
368              else {indicadorLoop=indicadorLoop+1;}
369             }
370
371             I++; if (I>4) {I=1;}
372             passo++;
373
374           }
375
376
377           //Imprimindo a solução do sistema
378           if (passo >= 3999) {
379             for (int i=0; i < (size+2); i++) {
380              solucao[i].setText("Este neural"); i++;
381              solucao[i].setText("nao assimila"); i++;
382              solucao[i].setText("esta tabela"); i++;
383              if (size>1){solucao[i].setText("de amostra."); i++;}
384             }
385           }
386           else {
387             int N=1; int M=0;
388             solucao[0].setText(String.valueOf(passo));
389             for (int i=1; i < (4*(size+1)+2); i++) {
390               if (M>2 && N<4) {M=0; N++;}
391                 solucao[i].setText(String.valueOf(w[N][M]));
392                 M++;
393               }
394             }
395           }
396
397
398   // final da classe Rummelhart4
```

APÊNDICE B – Códigos-fonte

Serão apresentados a seguir os códigos-fonte do .class de consulta e geração do arquivo texto (FIG. DE APÊNDICE B 1a) que contém as entradas para o mapa auto-organizável de Kohonen, da página HTML com o formulário dos parâmetros de treinamento do mapa auto-organizável (FIG. DE APÊNDICE B 1b), da página HTML que portará o *applet* Java (FIG. DE APÊNDICE B 1c) e finalmente do próprio *applet* Java, nessa ordem.

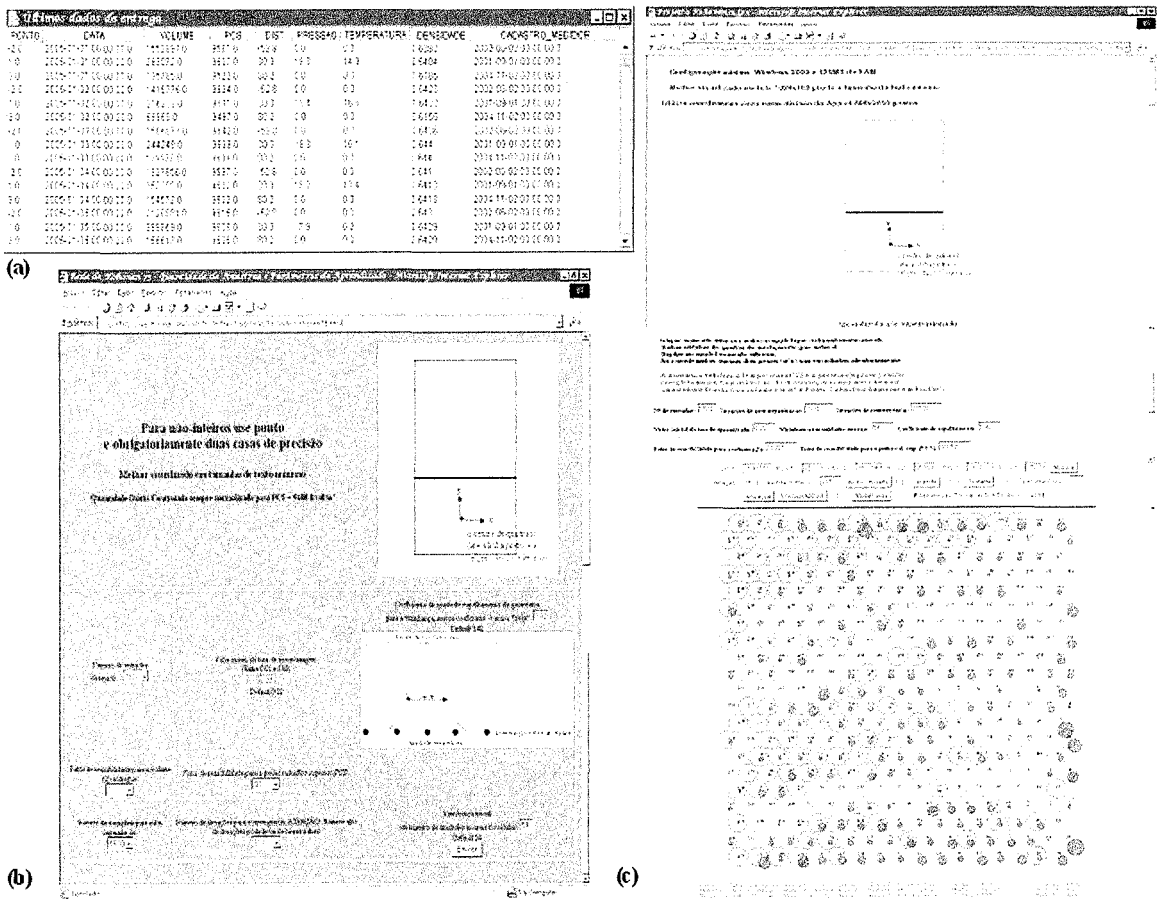


FIG. DE APÊNDICE B 1 - Interfaces gráficas dos códigos-fonte.

ExibeTabela9.java

```
1  /* Acesso a banco de dados Oracle com impressão de dados em arquivo
   texto **
2  **Volume em m³, PCS em kcal/m³ para três city-gates em linhas
   intercaladas**
3  **     com indicador inicial aleatorio de dia da semana no último campo
   **
4  **      Copyright Waldemar Dorazkevicius Jr. -- waldj@petrobras.com.br
   */
5
6  import java.sql.*;
7  import javax.swing.*;
8  import java.awt.*;
9  import java.awt.event.*;
10 import java.util.*;
11 import javax.sql.*;
12 import java.io.*;
13
14
15
16 public class ExibeTabela9 extends JFrame {
17
18   private JTable tabela;
19   private java.sql.Connection connexao;
20
21
22   public void ExibeTabela9() {
23
24
25    try {//passo 0: instancia driver de comunicação
26       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
27    }//0
28
29
30    catch (ClassNotFoundException e) {
31       System.err.println(e);
32       e.printStackTrace();
33    }
34
35
36     cargaTabela();
37     setSize(1000,500);
38     show();
39
40   }//fim do método ExibeTabela9
41
42
43
44   private  void cargaTabela() {
45        Statement declaracao;
46        ResultSet resultados;
47        String url1= "jdbc:odbc:INFOCHEM";//-- case sensitive
48        String usuario = "S9001";
49        String senha = "NONONONONO";//-- case sensitive
50
51        try {
52              String query = "SELECT AGFX.NR_ORDEM AS PONTO,"+
53                             "MEDN_DH_MEDICAO AS DATA,"+
54                             "SUM(MEDN_QN_VL_C_CORR) AS VOLUME,"+
55                             "AVG(J.MEDN_MD_PCS) AS PCS,"+
56                             "AGFX_MD_DIST_ORIGEM AS DIST,"+
57                             "MAX(V.VAOP_MD_PRES_OPE) AS PRESSAO,"+
58                             "MIN(V.VAOP_MD_TEMP) AS TEMPERATURA,"+
59                             "AVG(J.MEDN_MD_DENSIDADE) AS DENSIDADE,"+
60
                               "MAX(M.MEGA_DH_ATUA_INF) AS
                               CADASTRO_MEDIDOR "+
61                             "FROM "+
62                             "MEDICAO_GAS_NAT J,VW_MEGA_DTMAX M,"+
63
                               "AGRUPAMENTO_PONTO_MEDICAO,VARIAVEIS_OPER
                               V "+
64                             "WHERE "+
65                             "(AGFX_TL_PONTO_MEDICAO='CV-DINDA' OR "+
```

```
66                                  "AGPM_CD_PONTO_MEDICAO='CY-CAPUAVA' OR "+
67                                  "AGPM_CD_PONTO_MEDICAO='CY-SJC') AND "+
68
                                    "J.PONE_CD_ID=AGPM_CD_PONTO_MEDICAO(+) AND
                                    "+
69                                  "J.PONE_CD_ID=V.PONE_CD_ID(+) AND "+
70                                  "J.MEGA_CD_ID=V.MEGA_CD_ID(+) AND "+
71
                                    "J.MEGN_DH_MEDICAO=V.VAGP_DH_LEITURA(+)
                                    AND "+
72                                  "J.PONE_CD_ID=N.PONE_CD_ID(+) AND "+
73                                  "J.MEGA_CD_ID=N.MEGA_CD_ID(+) "+
74
                                    "AND MEGN_DH_MEDICAO>=TO_DATE('17102002
                                    00:00', 'DD/MM/YYYY HH24:MI') "+
75
                                    "AND MEGN_DH_MEDICAO<=TO_DATE('04112004
                                    00:00', 'DD/MM/YYYY HH24:MI') "+
76                                  "GROUP BY MEGN_DH_MEDICAO,"+
77                                  "AGPM_NR_ORDEM,AGPM_NO_DIST_ORIGEM "+
78
                                    "ORDER BY MEGN_DH_MEDICAO
                                    ASC,AGPM_NR_ORDEM";
79              try {//passo 1: conexão
80                  connexao = DriverManager.getConnection(url1,usuario,
                    senha);
81                  try {//passo 2: declaração
82                     declaracao=connexao.createStatement();
83                     try {//passo 3: executar query
84                        resultados = declaracao.executeQuery(query);
85                        mostraResultados(resultados);
86                     }//2
87                     finally {
88                        declaracao.close();
89                     }
90                  }//2
91                  finally {
92                  }
93              }//1
94
95              catch (java.sql.SQLException h) {
96                 System.out.println("Inapto para conectar");
97                 h.printStackTrace();
98              }
99
100         }
101
102
103         catch (Exception ex) {
104              System.out.println(
                 "Inapto para conexão/execução de query");
105              ex.printStackTrace();
106         }
107
108
109
110     }//fim do método cargaTabela
111
112
113
114     private void mostraResultados(ResultSet res) throws java.sql.
        SQLException {
115       //posiciona para o primeiro registro
116       boolean maisRegistros = res.next();
117       int numeroLinha=0;//conta as linhas do ResultSet
118       int numeroDiaSemana=1;//atribuição de inteiros a dias da semana
119
120       //se não houver registros, exibe uma mensagem
121       if (!maisRegistros) {
122         JOptionPane.showMessageDialog(this,
          "O ResultSet não possui nenhum registro");
123         setTitle("Nenhum registro disponível");
124         return;//sai do método
```

```
125       }
126
127       setTitle("Títulos das datas de entrega");
128
129       Vector colunasCabecalho = new Vector();
130       Vector linhas = new Vector();
131
132       try {
133          //gera o arquivo .txt e o seu processo de inscrição
134          FileWriter arqTextoSaida = new FileWriter("datones.txt");
135          PrintWriter arqDadosSaida = new PrintWriter(arqTextoSaida);
136          //obtém titulos da coluna
137          ResultSetMetaData rsmd = res.getMetaData();
138          for (int i=1; i<= rsmd.getColumnCount(); ++i) {
139            colunasCabecalho.addElement(rsmd.getColumnName(i));
140          }
141          //obtém dados da linha
142          do {
143            numeroLinha--;
144            if (numeroLinha>1 && (numeroLinha-1)%3==0)
               {numeroDiaSemana++;}
145            if (numeroDiaSemana>7) {numeroDiaSemana=1;}
146            linhas.addElement(pegaProximaLinha(res,rsmd,arqDadosSaida,
               numeroLinha,numeroDiaSemana));
147          } while (res.next());
148          //exibe a tabela com conteúdos de resultSet
149          tabela = new JTable(linhas,colunasCabecalho);
150          JScrollPane elevador = new JScrollPane(tabela);
151          getContentPane().add(elevador,BorderLayout.CENTER);
152          validate();
153          arqDadosSaida.close();
154       }
155       catch (IOException io) {
156          System.err.println("Erro na leitura do arquivo");
157       }
158       catch (java.sql.SQLException sqlex) {
159          System.out.println("Inapto para mostrar registros");
160          sqlex.printStackTrace();
161       }
162    }//fim do método mostraResultados
163
164
165
166    private Vector pegaProximaLinha(ResultSet res, ResultSetMetaData
       rsmd, PrintWriter arqDadosSaida, int numeroLinha, int
       numeroDiaSemana) throws java.sql.SQLException {
167      Vector correnteLinha = new Vector();
168       for (int i = 1; i <= rsmd.getColumnCount(); ++i) {//n
169         if (i==1){arqDadosSaida.print(numeroLinha); arqDadosSaida.print
          (";");}
170         switch(rsmd.getColumnType(i)) {
171             case Types.VARCHAR:
172                 correnteLinha.addElement(res.getString(i));
173                 arqDadosSaida.print(correnteLinha.elementAt(i-1));
174                 if (i==rsmd.getColumnCount())
                    {arqDadosSaida.print(";");}
175                 else {arqDadosSaida.print(";");}
176                 break;
177             case Types.CHAR:
178                 correnteLinha.addElement(res.getString(i));
179                 arqDadosSaida.print(correnteLinha.elementAt(i-1));
180                 if (i==rsmd.getColumnCount())
                    {arqDadosSaida.print(";");}
181                 else {arqDadosSaida.print(";");}
182                 break;
183             case Types.INTEGER:
184                  correnteLinha.addElement(new Long(res.getLong(i)));
185                  arqDadosSaida.print(correnteLinha.elementAt(i-1));
186                  if (i==rsmd.getColumnCount())
                     {arqDadosSaida.print(";");}
187                  else {arqDadosSaida.print(";");}
188                  break;
189             case Types.DOUBLE:
```

```
190                     correnteLinha.addElement(new Float(res.getFloat(i)));
191                     arqDadosSaida.print(correnteLinha.elementAt(i-1));
192                     if (i==rsmd.getColumnCount())
                        arqDadosSaida.print(";");}
193                     else {arqDadosSaida.print(";");}
194                     break;
195                 case Types.FLOAT:
196                     correnteLinha.addElement(new Float(res.getFloat(i)));
197                     arqDadosSaida.print(correnteLinha.elementAt(i-1));
198                     if (i==rsmd.getColumnCount())
                        {arqDadosSaida.print(";");}
199                     else {arqDadosSaida.print(";");}
200                     break;
201                 case Types.REAL:
202                     correnteLinha.addElement(new Float(res.getFloat(i)));
203                     arqDadosSaida.print(correnteLinha.elementAt(i-1));
204                     if (i==rsmd.getColumnCount())
                        {arqDadosSaida.print(";");}
205                     else {arqDadosSaida.print(";");}
206                     break;
207                 case Types.NUMERIC:
208                     correnteLinha.addElement(new Float(res.getFloat(i)));
209                     arqDadosSaida.print(correnteLinha.elementAt(i-1));
210                     if (i==rsmd.getColumnCount())
                        {arqDadosSaida.print(";");}
211                     else {arqDadosSaida.print(";");}
212                     break;
213                 case Types.DECIMAL:
214                     correnteLinha.addElement(new Float(res.getFloat(i)));
215                     arqDadosSaida.print(correnteLinha.elementAt(i-1));
216                     if (i==rsmd.getColumnCount())
                        {arqDadosSaida.print(";");}
217                     else {arqDadosSaida.print(";");}
218                     break;
219                 case Types.BIGINT:
220                     correnteLinha.addElement(new Long(res.getLong(i)));
221                     arqDadosSaida.print(correnteLinha.elementAt(i-1));
222                     if (i==rsmd.getColumnCount())
                        {arqDadosSaida.print(";");}
223                     else {arqDadosSaida.print(";");}
224                     break;
225                 case Types.TIMESTAMP:
226                     correnteLinha.addElement(res.getTimestamp(i));
227                     arqDadosSaida.print(correnteLinha.elementAt(i-1));
228                     if (i==rsmd.getColumnCount())
                        {arqDadosSaida.print(";");}
229                     else {arqDadosSaida.print(";");}
230                     break;
231                 case Types.OTHER:
232                     correnteLinha.addElement(new Float(res.getFloat(i)));
233                     arqDadosSaida.print(correnteLinha.elementAt(i-1));
234                     if (i==rsmd.getColumnCount())
                        {arqDadosSaida.print(";");}
235                     else {arqDadosSaida.print(";");}
236                     break;
237                 default:
238                     System.out.println("O tipo era: " + rsmd.
                        getColumnTypeName(i));
239                     break;
240             }
241         }//n
242         arqDadosSaida.print(numeroDiaSemana); arqDadosSaida.println("");
243         return correnteLinha;
244     }//fim do método pegaProximaLinha
245
246
247
248     public void encerraConnexao() {
249         try {
250             connexao.close();
251         }
252         catch (java.sql.SQLException sqlex) {
253             System.err.println("Impossível para desconectar");
```

```
254             sqlex.printStackTrace();
255           }
256       }//fim do método encerraConnexao
257
258
259
260       public static void main(String[] args) {
261         final ExibeTabela9 app = new ExibeTabela9();
262         app.ExibeTabela9();
263         app.addWindowListener(
264           new WindowAdapter() {
265             public void windowClosing(WindowEvent e) {
266               app.encerraConnexao();
267               System.exit(0);
268             }
269           }
270         );
271       }//fim do método main
272
273
274
275   }//fim da classe ExibeTabela9
```

indexKohonen73.html

```
<HTML>

<HEAD>



<TITLE>Rede de Kohonen 73 - Quantidade de Amostras e Parâmetros de Aprendizado

</TITLE>



</HEAD>



<BODY bgcolor="#66BF33">


<table width="100%" align="center">

    <tr>
        <td align="center">
           <FONT SIZE=-3><B>Para n&atilde;o-inteiros use ponto <br> e obrigatoriamente duas casas de preci&atilde;o</B>
           <br><br>
           <FONT SIZE=-2><B>Melhor visualizado em tamanho de texto m&iacute;nimo</B>
           <br><br>
           <FONT SIZE=-1><B>Quantidade D&aacute;ria Contratada sempre normalizada para PCS = 9400 kcal/m³</B>
           </FONT>
        </td>
        <td  align="center">
           <IMG SRC="../General/coordAppletKohonen.bmp">
        </td>
    </tr>

</table>



<FORM ACTION="Kohonen73.html" METHOD=GET>

 <table border="3" bordercolor="#FFFF00" width="100%" align="right">
   <tr>
     <td  align="center">
      <br>Número de entradas desejado:  
      <select name="sub">
      <option value="0021">21
      <option value="0030">30
      <option value="0060">60
      <option value="0090">90
      <option value="0100">100
      <option value="0132">132
      <option value="0150">150
      <option value="0200">200
      <option value="0250">250
      <option value="0300">300
      <option value="0360">360
      <option value="0400">400
      <option value="0450">450
      <option value="0500">500
      <option value="0550">550
      <option value="0600">600
      <option value="0750">750
      <option value="0900">900
      <option value="1200">1200
      <option value="1500">1500
      <option selected value="1800">1800
      <option value="2100">2100
      </select><br><br>
```

```
 70        </td>
 71        <td  align="center">
 72         <br>Valor inicial da taxa de aprendizagem.
 73         <br>(Entre 0.01 e 5.00)<br>
 74         <INPUT TYPE="TEXT"  MAXLENGHT="4" SIZE="1" NAME="bus" VALUE=
            "0.22">
 75         <br>Default 0.22<br>
 76        </td>
 77        <td  align="center">
 78         <br>Coeficiente de ajuste de espalhamento da gaussiana
 79         <br>para a vizinhança, menor coeficiente -> maior "bojo".
 80         <INPUT TYPE="TEXT"  MAXLENGHT="4" SIZE="1" NAME="usb2" VALUE=
            "1.40">
 81         <br>Default 1.40<br>
 82         <IMG SRC="../General/gaussIlustrII.jpg">
 83        </td>
 84       </tr>
 85       <tr align="center">
 86        <td align="center">
 87         <br>Fator de sensibilidade para o volume (Q) entregue:<br>
 88         <select name="ubs">
 89          <option value="0.001">0.001
 90          <option value="0.010">0.01
 91          <option value="0.500">0.5
 92          <option value="0.100">0.1
 93          <option selected value="00001">1
 94          <option value="00010">10
 95          <option value="00050">50
 96          <option value="00100">100
 97          <option value="01000">1000
 98         </select><br>
 99        </td>
100        <td align="center">
101         <br>Fator de sensibilidade para o poder calorífico superior
            (PCS):<br>
102         <select name="bsu">
103          <option value="0.001">0.001
104          <option value="0.010">0.01
105          <option value="0.500">0.5
106          <option value="0.100">0.1
107          <option value="00001">1
108          <option value="00010">10
109          <option selected value="00050">50
110          <option value="00100">100
111          <option value="01000">1000
112         </select><br>
113        <td align="center">
114
115        </td>
116       </tr>
117       <tr align="center">
118        <td align="center">
119         <br>Número de iterações para auto-organização:<br>
120         <select name="usb1">
121          <option value="0200">200
122          <option value="0500">500
123          <option value="1000">1000
124          <option value="1200">1200
125          <option selected value="1500">1500
126          <option value="2000">2000
127         </select><br>
128        </td>
129        <td align="center">
130         <br>Número de iterações para convergência:
131         ATENÇÃO! Número alto de iterações pode levar de horas a dias!<br
            >
132         <select name="usb2">
133          <option value="00000">0
134          <option selected value="00500">500
135          <option value="01000">1000
136          <option value="02000">2000
137          <option value="03000">3000
138          <option value="05000">5000
```

```
139          <option value="10000">10000
140          <option value="50000">50000
141          <option value="80000">80000
142          <option value="99999">99999
143         </select><br>
144        </td>
145        <td align="center">
146         <br>Vizinhança inicial
147         <br>em número de unidades neurais ou células.
148         <INPUT TYPE="TEXT"  MAXLENGHT="3" SIZE="1" NAME="usb1" VALUE=
            "54">
149         <br>Default 54<br>
150         <INPUT TYPE="SUBMIT" VALUE="Enviar">
151        </td>
152      </tr>
153    </table>
154  </form>
155
156
157  </BODY>
158
159
160
161  </HTML>
```

Kohonen73.html

```
<HTML>
<HEAD>

<TITLE>Projecto Kohonen73.jpx</TITLE>

<SCRIPT LANGUAGE="JavaScript">
<!-- Inicio do script netie
function netie(net, ie) {
  if ((navigator.appVersion.substring(0,3) >= net && navigator.appName
  == 'Netscape' && net != -1) || (navigator.appVersion.substring(0,3)
  >= ie && navigator.appName.substring(0,9) == 'Microsoft' && ie !=
  -1))
    return true;
  else return false;
}
//  Fim do script netie -->
</SCRIPT>

<SCRIPT LANGUAGE="JavaScript">
<!-- Inicio do script transitionTxt
function transitionTxt() {
  if (idTransTxt.filters.item(0).status == 0) {
    idTransTxt.filters.item(0).apply();
    idTransTxt.style.visibility = "visible";
    idTransTxt.filters(0).play();
  }
  if (idTransTit.filters.item(0).status == 0) {
    idTransTit.filters.item(0).apply();
    idTransTit.style.visibility = "visible";
    idTransTit.filters(0).play();
  }
}
//  Fim do script transitionTxt -->
</SCRIPT>

</HEAD>



<BODY onLoad="netie(-1,4)?transitionTxt():null">

<H4><FONT color=#6A4033>By Waldemar Doraskevi&#269;ius
Jr    n<SUP>o</SUP> 1987639</FONT><H4>
<BR>
<DIV ID="idTransTit" align = "left" STYLE=
"position:relative; left:20; height:40; width:700; color:#990033;
font-size:14; font-family:Arial Rounded MT Bold ;
filter:revealTrans(Duration=7, Transition=70); visibility:hidden">
           Configura&ccedil;&atilde;o m&iacute;nima:
        Windows 2000 e 128MB de RAM<BR><BR>
           Melhor visualizado em tela 1024x768 pixels e
        tamanho de texto m&iacute;nimo<BR><BR>
Utiliza coordenadas Java numa divis&atilde;o do Applet 700x800 pontos
</DIV>


<table width="100%" align="center">
  <tr>
    <td  align="center">
      <IMG SRC="../General/coordAppletKohonen.bmp">
    </td>
  </tr>
</table>


<H2 align=center><br> <br>Aprendizado fracamente supervisionado</H2>
<BR>
<DIV ID="idTransTxt" align = "left" STYLE=
"position:relative; left:20; height:40; width:700; color:#3322d2;
font-size:12; font-family:Arial Rounded MT Bold ;
filter:revealTrans(Duration=7, Transition=70); visibility:hidden">
        Clique somente uma vez sobre o applet que roda
        automaticamente.<BR>
```

```
 61          Dados obtidos de pontos de medição de g&aacute;s natural<BR>
 62          Digitar no applet somente inteiros.<BR>
 63          As coordenadas iniciais dos pesos (w's) s&atilde;o escolhidas
             aleatoriamente.
 64  </DIV>
 65
 66  <BR>
 67  <DIV align = "left" STYLE=
     "position:relative; left:20; height:30; width:600; color:#44DD33;
     font-size:13; font-family:Arial Rounded MT Bold;">
 68          A dimens&atilde;o relativa à temperatura(°C) e à press&atilde;
             o(kgf/cm²) est&atilde;o<BR>
 69          completamente suprimidas neste desenho, mas n&atilde;o nos
             c&aacute;lculos.<BR>
 70          Quantidade Diária Contratada em m³ e Poder Calorífico Superior
             em kcal/m³.
 71  </DIV>
 72
 73
 74  <FORM NAME="yip">
 75
 76  <SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT"><!--
 77
 78    var ll=location.search;
 79    var mm=location.search;
 80    var nn2=location.search;
 81    var oo=location.search;
 82    var rr=location.search;
 83    var pp1=location.search;
 84    var pp2=location.search;
 85    var nn1=location.search;
 86    ll=ll.substring(5,9);
 87    mm=mm.substring(14,18);
 88    nn2=nn2.substring(24,28);
 89    oo=oo.substring(33,38);
 90    rr=rr.substring(43,48);
 91    pp1=pp1.substring(54,58);
 92    pp2=pp2.substring(64,69);
 93    nn1=nn1.substring(75,79);
 94    this.yip.value=ll;
 95    this.yip.value=mm;
 96    this.yip.value=nn2;
 97    this.yip.value=oo;
 98    this.yip.value=rr;
 99    this.yip.value=pp1;
100    this.yip.value=pp2;
101    this.yip.value=nn1;
102
103
       document.write("  N<SUP>o</SUP>
        de entradas: <INPUT TYPE=TEXT NAME='sup'
       value=",ll," MAXLENGTH=2 SIZE=3>")
104
105    document.write("  Itera&ccedil;&otilde;es de
       auto-organiza&ccedil;&atilde;o: <INPUT TYPE=TEXT NAME='psu1'
       value=",pp1," MAXLENGTH=5 SIZE=6>")
106
107    document.write("  Itera&ccedil;&otilde;es de
       convergência: <INPUT TYPE=TEXT NAME='psu2' value=",pp2,"
       MAXLENGTH=6 SIZE=7>")
108
109    document.write("<HR>")
110
111    document.write("  Valor inicial da taxa de
       aprendizado: <INPUT TYPE=TEXT NAME='pus' value=",mm,"
       MAXLENGTH=3 SIZE=4>")
112
113    document.write("  Vizinhança em unidades
       neurais: <INPUT TYPE=TEXT NAME='ups1' value=",nn1," MAXLENGTH=3
       SIZE=4>")
114
115    document.write("  Coeficiente de espalhamento: <INPUT
       TYPE=TEXT NAME='ups2' value=",nn2," MAXLENGTH=3 SIZE=4>")
```

```
116
117     document.write("<HR>")
118
119     document.write("  Fator de sensibilidade para o volume
        (Q): <INPUT TYPE=TEXT NAME='usp' value=",oo," MAXLENGTH=6
        SIZE=7>")
120
121     document.write("  Fator de sensibilidade para o poder cal.
        sup. (PCS): <INPUT TYPE=TEXT NAME='psu' value=",rr,"
        MAXLENGTH=6 SIZE=7>")
122
123     document.write("<HR>")
124
125        document.write("<DIV ID='col' ALIGN='center'>")
126
127           document.write("<Applet name='apple' code='Kohonen73.class'
              width=800  height=2850>")
128           document.write("<param name='tam' value=",ll,">")
129           document.write("<param name='mat' value=",mm,">")
130           document.write("<param name='atm1' value=",nn1,">")
131           document.write("<param name='atm2' value=",nn2,">")
132           document.write("<param name='amt' value=",oo,">")
133           document.write("<param name='tma' value=",rr,">")
134           document.write("<param name='mta1' value=",pp1,">")
135           document.write("<param name='mta2' value=",pp2,">")
136           document.write("</applet>")
137
138        document.write("</DIV>")
139
140  --></SCRIPT>
141
142  <!-- o value é o número de sinapses, excetuando o referente ao
     parâmetro de corte (theta) -->
143
144  </FORM>
145
146  <HR>
147
148  <DIV ID='mail' ALIGN='right'STYLE=
     "position:relative; left:900; height:20; width:100; color:#9966CC;
     font-size:8; font-family:Arial Rounded MT Bold">
149           waldj@petrobras.com.br
150  </DIV>
151
152
153  </BODY>
154  </HTML>
```

Kohonen73.java

```
1  /** Produced and created by W. Doraskevicius, Jr ...
   waldi@petrobras.com.br ...**
2   **
    **
3   **             Aprendizado por competição entre 367 neurônios
    **
4   **                 numa topologia bidimensional hexagonal.
    **
5   **           Neurônios nos vértices e nos centros de 110 favos.
    **
6   **  Leitura de dados de arquivo .txt para até 9 pontos (estações de
    medição). **
7   ** Dimensão dos dados: 6; label do ponto, volume, PCS, pressão,
    temp. e dens. **
8   ** Qualidade dos dados por PCS e volume, normalizado também a PCS (4
    classes).**/
9
10
11  import java.applet.*;
12  import java.awt.*;
13  import java.awt.event.*;
14  import java.util.*;
15  import java.lang.*;
16  import java.io.*;
17  import java.text.DecimalFormat;
18  import java.util.List;
19
20
21
22  public class Kohonen73 extends Applet implements ActionListener,
    Runnable,Serializable {
23
24    //declaração de variáveis globais
25    Thread thread;
26
27    int dim = 6;//dimensão do vetor de dados pertinente
28    int nPesos=367;//número de unidades neurais de saída
29    int nEntradas=100;//declaração default do número de entradas
30    int nMaximoDados=2500;//declaração default do número máximo de
      dados no datasource .txt
31    int nTuplas=700;//declaração default do número máximo de dados
      treinamento+teste
32    int nVezesPassAmostra=1500;//número de iterações de ordenamento por
      toda a amostragem
33    int nVezesPassAjusteFino=2000;//número de iterações de convergência
      por toda a amostragem
34    int nPontos=3;//número default de pontos de medição considerados
      para a classificação
35
36    int ii=0;//iterações+click
37    int jj=0;//click para validação aleatória sobre tela (.V) - mín = 0
      e máx = 99
38
39    int pesoVictor=0;
40    String bandeiraAcende="nao";
41    String bandeiraPisca ="down";
42    String enche="nao";
43
44    Vector batida = new Vector();
45    Vector pesao = new Vector();
46    Vector prev = new Vector();
47    double prevDimAux[][];//para acima de duas dimensões para auxiliar
      prev (point) na GUI
48
49    Label grandezaQ[];
50    Label grandezaP[];
51    Label etiqueta1;
52    Label etiqueta2;
53    Label etiqueta3;
54    Label etiqueta4;
55    Label aviso1;
56    Label aviso2;
57    Label espaco0;
```

```
58      Label espaco1;
59      Label espaco2;
60      Label espaco3;
61      Label espaco4;
62      TextField selecao;
63      TextField selection;
64      TextField campoQ[];
65      TextField campoP[];
66      Button atribui;
67      Button apaga;
68      Button rest;
69      Button atribuicao;
70      Button mapeia;
71      Button preenche;
72      Button mudaBG;
73
74      String autorizacao = new String("ande");
75
76      double x_d[][];//as devidas entradas
77      int avalia_d[];//classificação de cada linha de entrada
78      int maisPerto[];//guarda o dado mais perto do neurônio indicado
        pelo índice
79      double w[][],w_inic[][];//vectores peso
80      int v[];//marca o neurônio vencedor para uma entrada
81      int vr[];//marca o neurônio vencedor para um dado de previsão
82      double neuro[][];//localização dos neurônios da camada de saída
83      double contaParaNeuro[];
84      int qNeuro[];
85      String alarme[];
86      String entrada[][];
87      String qualy[][];
88      int iteracione = 0;//número de épochas (# de passagem de todos os
        dados)
89      double media[][];//média de cada medida para cada ponto de medição
90      double desvioPadraoSignif[][];//desvio padrão de cada grandeza p/
        cada ponto de medição, algarismo significativo
91      double desvioPadraoNat[][];//desvio padrão de cada grandeza p/ cada
        ponto de medição, sem escala
92      int dPAlgarismoSignificativo[][];//o(s) algarismo(s)
        significativo(s) do D.P. como int
93      Random aleatorio;
94
95      int ciclicoBG =1;
96
97      //Parâmetros adicionais para os métdodos calcXXX()
98      double ETA_INI;//fator de aprendizado: inicial
99      double CR = 6.00d;//ajuste de espalhamento da gaussiana
100     double R_INI = 30d;//valor inicial da vizinhança, deve cobrir a >
        dist. entre 2 unids. neurais
101
102     //unidades de distância para as coordenadas das unidades neurais
103     double modulo = 2.0d;
104     double nodulo = 0.5d*Math.sqrt(3d)*modulo;
105
106     //Monta a interface gráphica
107     public void init() {
108
109      //coordenadas de 367 unidades neurais em estrutura hexagonal plana
110      neuro = new double[nPesos+1][3];
111      for(int i=0; i<(nPesos+1); i++) {
112       neuro[i][0]=-1d*modulo;
113      }
114
115      neuro[0][1]=-1d*modulo;    neuro[0][2]=-1d*nodulo;
116
117      //Primeiro quadrante dos pontos dos favos de corpo centrado
118      neuro[1][1]=0d;                        neuro[1][2]=0d;
119      neuro[2][1]=modulo;                    neuro[2][2]=0d;
120      neuro[3][1]=2d*modulo;                 neuro[3][2]=0d;
121      neuro[4][1]=3d*modulo;                 neuro[4][2]=0d;
122      neuro[5][1]=4d*modulo;                 neuro[5][2]=0d;
123      neuro[6][1]=5d*modulo;                 neuro[6][2]=0d;
124      neuro[7][1]=6d*modulo;                 neuro[7][2]=0d;
```

```
125     neuro[8][1]=7d*modulo;              neuro[8][2]=0d;
126     neuro[9][1]=8d*modulo;              neuro[9][2]=0d;
127     neuro[10][1]=0.5d*modulo;           neuro[10][2]=modulo;
128     neuro[11][1]=1.5d*modulo;           neuro[11][2]=modulo;
129     neuro[12][1]=2.5d*modulo;           neuro[12][2]=modulo;
130     neuro[13][1]=3.5d*modulo;           neuro[13][2]=modulo;
131     neuro[14][1]=4.5d*modulo;           neuro[14][2]=modulo;
132     neuro[15][1]=5.5d*modulo;           neuro[15][2]=modulo;
133     neuro[16][1]=6.5d*modulo;           neuro[16][2]=modulo;
134     neuro[17][1]=7.5d*modulo;           neuro[17][2]=modulo;
135     neuro[18][1]=8.5d*modulo;           neuro[18][2]=modulo;
136     neuro[19][1]=0d;                    neuro[19][2]=2d*modulo;
137     neuro[20][1]=modulo;                neuro[20][2]=2d*modulo;
138     neuro[21][1]=2d*modulo;             neuro[21][2]=2d*modulo;
139     neuro[22][1]=3d*modulo;             neuro[22][2]=2d*modulo;
140     neuro[23][1]=4d*modulo;             neuro[23][2]=2d*modulo;
141     neuro[24][1]=5d*modulo;             neuro[24][2]=2d*modulo;
142     neuro[25][1]=6d*modulo;             neuro[25][2]=2d*modulo;
143     neuro[26][1]=7d*modulo;             neuro[26][2]=2d*modulo;
144     neuro[27][1]=8d*modulo;             neuro[27][2]=2d*modulo;
145     neuro[28][1]=0.5d*modulo;           neuro[28][2]=3d*modulo;
146     neuro[29][1]=1.5d*modulo;           neuro[29][2]=3d*modulo;
147     neuro[30][1]=2.5d*modulo;           neuro[30][2]=3d*modulo;
148     neuro[31][1]=3.5d*modulo;           neuro[31][2]=3d*modulo;
149     neuro[32][1]=4.5d*modulo;           neuro[32][2]=3d*modulo;
150     neuro[33][1]=5.5d*modulo;           neuro[33][2]=3d*modulo;
151     neuro[34][1]=6.5d*modulo;           neuro[34][2]=3d*modulo;
152     neuro[35][1]=7.5d*modulo;           neuro[35][2]=3d*modulo;
153     neuro[36][1]=8.5d*modulo;           neuro[36][2]=3d*modulo;
154     neuro[37][1]=0d;                    neuro[37][2]=4d*modulo;
155     neuro[38][1]=modulo;                neuro[38][2]=4d*modulo;
156     neuro[39][1]=2d*modulo;             neuro[39][2]=4d*modulo;
157     neuro[40][1]=3d*modulo;             neuro[40][2]=4d*modulo;
158     neuro[41][1]=4d*modulo;             neuro[41][2]=4d*modulo;
159     neuro[42][1]=5d*modulo;             neuro[42][2]=4d*modulo;
160     neuro[43][1]=6d*modulo;             neuro[43][2]=4d*modulo;
161     neuro[44][1]=7d*modulo;             neuro[44][2]=4d*modulo;
162     neuro[45][1]=8d*modulo;             neuro[45][2]=4d*modulo;
163     neuro[46][1]=0.5d*modulo;           neuro[46][2]=5d*modulo;
164     neuro[47][1]=1.5d*modulo;           neuro[47][2]=5d*modulo;
165     neuro[48][1]=2.5d*modulo;           neuro[48][2]=5d*modulo;
166     neuro[49][1]=3.5d*modulo;           neuro[49][2]=5d*modulo;
167     neuro[50][1]=4.5d*modulo;           neuro[50][2]=5d*modulo;
168     neuro[51][1]=5.5d*modulo;           neuro[51][2]=5d*modulo;
169     neuro[52][1]=6.5d*modulo;           neuro[52][2]=5d*modulo;
170     neuro[53][1]=7.5d*modulo;           neuro[53][2]=5d*modulo;
171     neuro[54][1]=8.5d*modulo;           neuro[54][2]=5d*modulo;
172     neuro[55][1]=0d;                    neuro[55][2]=6d*modulo;
173     neuro[56][1]=modulo;                neuro[56][2]=6d*modulo;
174     neuro[57][1]=2d*modulo;             neuro[57][2]=6d*modulo;
175     neuro[58][1]=3d*modulo;             neuro[58][2]=6d*modulo;
176     neuro[59][1]=4d*modulo;             neuro[59][2]=6d*modulo;
177     neuro[60][1]=5d*modulo;             neuro[60][2]=6d*modulo;
178     neuro[61][1]=6d*modulo;             neuro[61][2]=6d*modulo;
179     neuro[62][1]=7d*modulo;             neuro[62][2]=6d*modulo;
180     neuro[63][1]=8d*modulo;             neuro[63][2]=6d*modulo;
181     neuro[64][1]=0.5d*modulo;           neuro[64][2]=7d*modulo;
182     neuro[65][1]=1.5d*modulo;           neuro[65][2]=7d*modulo;
183     neuro[66][1]=2.5d*modulo;           neuro[66][2]=7d*modulo;
184     neuro[67][1]=3.5d*modulo;           neuro[67][2]=7d*modulo;
185     neuro[68][1]=4.5d*modulo;           neuro[68][2]=7d*modulo;
186     neuro[69][1]=5.5d*modulo;           neuro[69][2]=7d*modulo;
187     neuro[70][1]=6.5d*modulo;           neuro[70][2]=7d*modulo;
188     neuro[71][1]=7.5d*modulo;           neuro[71][2]=7d*modulo;
189     neuro[72][1]=8.5d*modulo;           neuro[72][2]=7d*modulo;
190     neuro[73][1]=0d;                    neuro[73][2]=8d*modulo;
191     neuro[74][1]=modulo;                neuro[74][2]=8d*modulo;
192     neuro[75][1]=2d*modulo;             neuro[75][2]=8d*modulo;
193     neuro[76][1]=3d*modulo;             neuro[76][2]=8d*modulo;
194     neuro[77][1]=4d*modulo;             neuro[77][2]=8d*modulo;
195     neuro[78][1]=5d*modulo;             neuro[78][2]=8d*modulo;
196     neuro[79][1]=6d*modulo;             neuro[79][2]=8d*modulo;
197     neuro[80][1]=7d*modulo;             neuro[80][2]=8d*modulo;
```

```
198     neuro[81][1]=8d*modulo;             neuro[81][2]=8d*modulo;
199     neuro[82][1]=0.5d*modulo;           neuro[82][2]=9d*modulo;
200     neuro[83][1]=1.5d*modulo;           neuro[83][2]=9d*modulo;
201     neuro[84][1]=2.5d*modulo;           neuro[84][2]=9d*modulo;
202     neuro[85][1]=3.5d*modulo;           neuro[85][2]=9d*modulo;
203     neuro[86][1]=4.5d*modulo;           neuro[86][2]=9d*modulo;
204     neuro[87][1]=5.5d*modulo;           neuro[87][2]=9d*modulo;
205     neuro[88][1]=6.5d*modulo;           neuro[88][2]=9d*modulo;
206     neuro[89][1]=7.5d*modulo;           neuro[89][2]=9d*modulo;
207     neuro[90][1]=8.5d*modulo;           neuro[90][2]=9d*modulo;
208     neuro[91][1]=0d;                    neuro[91][2]=10d*modulo;
209     neuro[92][1]=modulo;                neuro[92][2]=10d*modulo;
210     neuro[93][1]=2d*modulo;             neuro[93][2]=10d*modulo;
211     neuro[94][1]=3d*modulo;             neuro[94][2]=10d*modulo;
212     neuro[95][1]=4d*modulo;             neuro[95][2]=10d*modulo;
213     neuro[96][1]=5d*modulo;             neuro[96][2]=10d*modulo;
214     neuro[97][1]=6d*modulo;             neuro[97][2]=10d*modulo;
215     neuro[98][1]=7d*modulo;             neuro[98][2]=10d*modulo;
216     neuro[99][1]=8d*modulo;             neuro[99][2]=10d*modulo;
217
218     //Segundo quadrante dos pontos dos favos de corpo centrado
219     neuro[100][1]=(-1d)*modulo;         neuro[100][2]=0d;
220     neuro[101][1]=(-1d)*2d*modulo;      neuro[101][2]=0d;
221     neuro[102][1]=(-1d)*3d*modulo;      neuro[102][2]=0d;
222     neuro[103][1]=(-1d)*4d*modulo;      neuro[103][2]=0d;
223     neuro[104][1]=(-1d)*5d*modulo;      neuro[104][2]=0d;
224     neuro[105][1]=(-1d)*6d*modulo;      neuro[105][2]=0d;
225     neuro[106][1]=(-1d)*7d*modulo;      neuro[106][2]=0d;
226     neuro[107][1]=(-1d)*8d*modulo;      neuro[107][2]=0d;
227     neuro[108][1]=(-1d)*0.5d*modulo;    neuro[108][2]=modulo;
228     neuro[109][1]=(-1d)*1.5d*modulo;    neuro[109][2]=modulo;
229     neuro[110][1]=(-1d)*2.5d*modulo;    neuro[110][2]=modulo;
230     neuro[111][1]=(-1d)*3.5d*modulo;    neuro[111][2]=modulo;
231     neuro[112][1]=(-1d)*4.5d*modulo;    neuro[112][2]=modulo;
232     neuro[113][1]=(-1d)*5.5d*modulo;    neuro[113][2]=modulo;
233     neuro[114][1]=(-1d)*6.5d*modulo;    neuro[114][2]=modulo;
234     neuro[115][1]=(-1d)*7.5d*modulo;    neuro[115][2]=modulo;
235     neuro[116][1]=(-1d)*8.5d*modulo;    neuro[116][2]=modulo;
236     neuro[117][1]=(-1d)*modulo;         neuro[117][2]=2d*modulo;
237     neuro[118][1]=(-1d)*2d*modulo;      neuro[118][2]=2d*modulo;
238     neuro[119][1]=(-1d)*3d*modulo;      neuro[119][2]=2d*modulo;
239     neuro[120][1]=(-1d)*4d*modulo;      neuro[120][2]=2d*modulo;
240     neuro[121][1]=(-1d)*5d*modulo;      neuro[121][2]=2d*modulo;
241     neuro[122][1]=(-1d)*6d*modulo;      neuro[122][2]=2d*modulo;
242     neuro[123][1]=(-1d)*7d*modulo;      neuro[123][2]=2d*modulo;
243     neuro[124][1]=(-1d)*8d*modulo;      neuro[124][2]=2d*modulo;
244     neuro[125][1]=(-1d)*0.5d*modulo;    neuro[125][2]=3d*modulo;
245     neuro[126][1]=(-1d)*1.5d*modulo;    neuro[126][2]=3d*modulo;
246     neuro[127][1]=(-1d)*2.5d*modulo;    neuro[127][2]=3d*modulo;
247     neuro[128][1]=(-1d)*3.5d*modulo;    neuro[128][2]=3d*modulo;
248     neuro[129][1]=(-1d)*4.5d*modulo;    neuro[129][2]=3d*modulo;
249     neuro[130][1]=(-1d)*5.5d*modulo;    neuro[130][2]=3d*modulo;
250     neuro[131][1]=(-1d)*6.5d*modulo;    neuro[131][2]=3d*modulo;
251     neuro[132][1]=(-1d)*7.5d*modulo;    neuro[132][2]=3d*modulo;
252     neuro[133][1]=(-1d)*8.5d*modulo;    neuro[133][2]=3d*modulo;
253     neuro[134][1]=(-1d)*modulo;         neuro[134][2]=4d*modulo;
254     neuro[135][1]=(-1d)*2d*modulo;      neuro[135][2]=4d*modulo;
255     neuro[136][1]=(-1d)*3d*modulo;      neuro[136][2]=4d*modulo;
256     neuro[137][1]=(-1d)*4d*modulo;      neuro[137][2]=4d*modulo;
257     neuro[138][1]=(-1d)*5d*modulo;      neuro[138][2]=4d*modulo;
258     neuro[139][1]=(-1d)*6d*modulo;      neuro[139][2]=4d*modulo;
259     neuro[140][1]=(-1d)*7d*modulo;      neuro[140][2]=4d*modulo;
260     neuro[141][1]=(-1d)*8d*modulo;      neuro[141][2]=4d*modulo;
261     neuro[142][1]=(-1d)*0.5d*modulo;    neuro[142][2]=5d*modulo;
262     neuro[143][1]=(-1d)*1.5d*modulo;    neuro[143][2]=5d*modulo;
263     neuro[144][1]=(-1d)*2.5d*modulo;    neuro[144][2]=5d*modulo;
264     neuro[145][1]=(-1d)*3.5d*modulo;    neuro[145][2]=5d*modulo;
265     neuro[146][1]=(-1d)*4.5d*modulo;    neuro[146][2]=5d*modulo;
266     neuro[147][1]=(-1d)*5.5d*modulo;    neuro[147][2]=5d*modulo;
267     neuro[148][1]=(-1d)*6.5d*modulo;    neuro[148][2]=5d*modulo;
268     neuro[149][1]=(-1d)*7.5d*modulo;    neuro[149][2]=5d*modulo;
269     neuro[150][1]=(-1d)*8.5d*modulo;    neuro[150][2]=5d*modulo;
270     neuro[151][1]=(-1d)*modulo;         neuro[151][2]=6d*modulo;
```

```
271     neuro[152][1]=(-1d)*2d*modulo;      neuro[152][2]=6d*modulo;
272     neuro[153][1]=(-1d)*3d*modulo;      neuro[153][2]=6d*modulo;
273     neuro[154][1]=(-1d)*4d*modulo;      neuro[154][2]=6d*modulo;
274     neuro[155][1]=(-1d)*5d*modulo;      neuro[155][2]=6d*modulo;
275     neuro[156][1]=(-1d)*6d*modulo;      neuro[156][2]=6d*modulo;
276     neuro[157][1]=(-1d)*7d*modulo;      neuro[157][2]=6d*modulo;
277     neuro[158][1]=(-1d)*8d*modulo;      neuro[158][2]=6d*modulo;
278     neuro[159][1]=(-1d)*0.5d*modulo;    neuro[159][2]=7d*modulo;
279     neuro[160][1]=(-1d)*1.5d*modulo;    neuro[160][2]=7d*modulo;
280     neuro[161][1]=(-1d)*2.5d*modulo;    neuro[161][2]=7d*modulo;
281     neuro[162][1]=(-1d)*3.5d*modulo;    neuro[162][2]=7d*modulo;
282     neuro[163][1]=(-1d)*4.5d*modulo;    neuro[163][2]=7d*modulo;
283     neuro[164][1]=(-1d)*5.5d*modulo;    neuro[164][2]=7d*modulo;
284     neuro[165][1]=(-1d)*6.5d*modulo;    neuro[165][2]=7d*modulo;
285     neuro[166][1]=(-1d)*7.5d*modulo;    neuro[166][2]=7d*modulo;
286     neuro[167][1]=(-1d)*8.5d*modulo;    neuro[167][2]=7d*modulo;
287     neuro[168][1]=(-1d)*modulo;         neuro[168][2]=8d*modulo;
288     neuro[169][1]=(-1d)*2d*modulo;      neuro[169][2]=8d*modulo;
289     neuro[170][1]=(-1d)*3d*modulo;      neuro[170][2]=8d*modulo;
290     neuro[171][1]=(-1d)*4d*modulo;      neuro[171][2]=8d*modulo;
291     neuro[172][1]=(-1d)*5d*modulo;      neuro[172][2]=8d*modulo;
292     neuro[173][1]=(-1d)*6d*modulo;      neuro[173][2]=8d*modulo;
293     neuro[174][1]=(-1d)*7d*modulo;      neuro[174][2]=8d*modulo;
294     neuro[175][1]=(-1d)*8d*modulo;      neuro[175][2]=8d*modulo;
295     neuro[176][1]=(-1d)*0.5d*modulo;    neuro[176][2]=9d*modulo;
296     neuro[177][1]=(-1d)*1.5d*modulo;    neuro[177][2]=9d*modulo;
297     neuro[178][1]=(-1d)*2.5d*modulo;    neuro[178][2]=9d*modulo;
298     neuro[179][1]=(-1d)*3.5d*modulo;    neuro[179][2]=9d*modulo;
299     neuro[180][1]=(-1d)*4.5d*modulo;    neuro[180][2]=9d*modulo;
300     neuro[181][1]=(-1d)*5.5d*modulo;    neuro[181][2]=9d*modulo;
301     neuro[182][1]=(-1d)*6.5d*modulo;    neuro[182][2]=9d*modulo;
302     neuro[183][1]=(-1d)*7.5d*modulo;    neuro[183][2]=9d*modulo;
303     neuro[184][1]=(-1d)*8.5d*modulo;    neuro[184][2]=9d*modulo;
304     neuro[185][1]=(-1d)*modulo;         neuro[185][2]=10d*modulo;
305     neuro[186][1]=(-1d)*2d*modulo;      neuro[186][2]=10d*modulo;
306     neuro[187][1]=(-1d)*3d*modulo;      neuro[187][2]=10d*modulo;
307     neuro[188][1]=(-1d)*4d*modulo;      neuro[188][2]=10d*modulo;
308     neuro[189][1]=(-1d)*5d*modulo;      neuro[189][2]=10d*modulo;
309     neuro[190][1]=(-1d)*6d*modulo;      neuro[190][2]=10d*modulo;
310     neuro[191][1]=(-1d)*7d*modulo;      neuro[191][2]=10d*modulo;
311     neuro[192][1]=(-1d)*8d*modulo;      neuro[192][2]=10d*modulo;
312
313     //Terceiro quadrante dos pontos dos favos de corpo centrado
314     neuro[193][1]=(-1d)*0.5d*modulo;    neuro[193][2]=(-1d)*modulo;
315     neuro[194][1]=(-1d)*1.5d*modulo;    neuro[194][2]=(-1d)*modulo;
316     neuro[195][1]=(-1d)*2.5d*modulo;    neuro[195][2]=(-1d)*modulo;
317     neuro[196][1]=(-1d)*3.5d*modulo;    neuro[196][2]=(-1d)*modulo;
318     neuro[197][1]=(-1d)*4.5d*modulo;    neuro[197][2]=(-1d)*modulo;
319     neuro[198][1]=(-1d)*5.5d*modulo;    neuro[198][2]=(-1d)*modulo;
320     neuro[199][1]=(-1d)*6.5d*modulo;    neuro[199][2]=(-1d)*modulo;
321     neuro[200][1]=(-1d)*7.5d*modulo;    neuro[200][2]=(-1d)*modulo;
322     neuro[201][1]=(-1d)*8.5d*modulo;    neuro[201][2]=(-1d)*modulo;
323     neuro[202][1]=0d;                   neuro[202][2]=(-1d)*2d*modulo;
324     neuro[203][1]=(-1d)*modulo;         neuro[203][2]=(-1d)*2d*modulo;
325     neuro[204][1]=(-1d)*2d*modulo;      neuro[204][2]=(-1d)*2d*modulo;
326     neuro[205][1]=(-1d)*3d*modulo;      neuro[205][2]=(-1d)*2d*modulo;
327     neuro[206][1]=(-1d)*4d*modulo;      neuro[206][2]=(-1d)*2d*modulo;
328     neuro[207][1]=(-1d)*5d*modulo;      neuro[207][2]=(-1d)*2d*modulo;
329     neuro[208][1]=(-1d)*6d*modulo;      neuro[208][2]=(-1d)*2d*modulo;
330     neuro[209][1]=(-1d)*7d*modulo;      neuro[209][2]=(-1d)*2d*modulo;
331     neuro[210][1]=(-1d)*8d*modulo;      neuro[210][2]=(-1d)*2d*modulo;
332     neuro[211][1]=(-1d)*0.5d*modulo;    neuro[211][2]=(-1d)*3d*modulo;
333     neuro[212][1]=(-1d)*1.5d*modulo;    neuro[212][2]=(-1d)*3d*modulo;
334     neuro[213][1]=(-1d)*2.5d*modulo;    neuro[213][2]=(-1d)*3d*modulo;
335     neuro[214][1]=(-1d)*3.5d*modulo;    neuro[214][2]=(-1d)*3d*modulo;
336     neuro[215][1]=(-1d)*4.5d*modulo;    neuro[215][2]=(-1d)*3d*modulo;
337     neuro[216][1]=(-1d)*5.5d*modulo;    neuro[216][2]=(-1d)*3d*modulo;
338     neuro[217][1]=(-1d)*6.5d*modulo;    neuro[217][2]=(-1d)*3d*modulo;
339     neuro[218][1]=(-1d)*7.5d*modulo;    neuro[218][2]=(-1d)*3d*modulo;
340     neuro[219][1]=(-1d)*8.5d*modulo;    neuro[219][2]=(-1d)*3d*modulo;
341     neuro[220][1]=0d;                   neuro[220][2]=(-1d)*4d*modulo;
342     neuro[221][1]=(-1d)*modulo;         neuro[221][2]=(-1d)*4d*modulo;
343     neuro[222][1]=(-1d)*2d*modulo;      neuro[222][2]=(-1d)*4d*modulo;
```

```
344     neuro[223][1]=(-1d)*3d*modulo;      neuro[223][2]=(-1d)*4d*modulo;
345     neuro[224][1]=(-1d)*4d*modulo;      neuro[224][2]=(-1d)*4d*modulo;
346     neuro[225][1]=(-1d)*5d*modulo;      neuro[225][2]=(-1d)*4d*modulo;
347     neuro[226][1]=(-1d)*6d*modulo;      neuro[226][2]=(-1d)*4d*modulo;
348     neuro[227][1]=(-1d)*7d*modulo;      neuro[227][2]=(-1d)*4d*modulo;
349     neuro[228][1]=(-1d)*8d*modulo;      neuro[228][2]=(-1d)*4d*modulo;
350     neuro[229][1]=(-1d)*0.5d*modulo;    neuro[229][2]=(-1d)*5d*modulo;
351     neuro[230][1]=(-1d)*1.5d*modulo;    neuro[230][2]=(-1d)*5d*modulo;
352     neuro[231][1]=(-1d)*2.5d*modulo;    neuro[231][2]=(-1d)*5d*modulo;
353     neuro[232][1]=(-1d)*3.5d*modulo;    neuro[232][2]=(-1d)*5d*modulo;
354     neuro[233][1]=(-1d)*4.5d*modulo;    neuro[233][2]=(-1d)*5d*modulo;
355     neuro[234][1]=(-1d)*5.5d*modulo;    neuro[234][2]=(-1d)*5d*modulo;
356     neuro[235][1]=(-1d)*6.5d*modulo;    neuro[235][2]=(-1d)*5d*modulo;
357     neuro[236][1]=(-1d)*7.5d*modulo;    neuro[236][2]=(-1d)*5d*modulo;
358     neuro[237][1]=(-1d)*8.5d*modulo;    neuro[237][2]=(-1d)*5d*modulo;
359     neuro[238][1]=0d;                   neuro[238][2]=(-1d)*6d*modulo;
360     neuro[239][1]=(-1d)*modulo;         neuro[239][2]=(-1d)*6d*modulo;
361     neuro[240][1]=(-1d)*2d*modulo;      neuro[240][2]=(-1d)*6d*modulo;
362     neuro[241][1]=(-1d)*3d*modulo;      neuro[241][2]=(-1d)*6d*modulo;
363     neuro[242][1]=(-1d)*4d*modulo;      neuro[242][2]=(-1d)*6d*modulo;
364     neuro[243][1]=(-1d)*5d*modulo;      neuro[243][2]=(-1d)*6d*modulo;
365     neuro[244][1]=(-1d)*6d*modulo;      neuro[244][2]=(-1d)*6d*modulo;
366     neuro[245][1]=(-1d)*7d*modulo;      neuro[245][2]=(-1d)*6d*modulo;
367     neuro[246][1]=(-1d)*8d*modulo;      neuro[246][2]=(-1d)*6d*modulo;
368     neuro[247][1]=(-1d)*0.5d*modulo;    neuro[247][2]=(-1d)*7d*modulo;
369     neuro[248][1]=(-1d)*1.5d*modulo;    neuro[248][2]=(-1d)*7d*modulo;
370     neuro[249][1]=(-1d)*2.5d*modulo;    neuro[249][2]=(-1d)*7d*modulo;
371     neuro[250][1]=(-1d)*3.5d*modulo;    neuro[250][2]=(-1d)*7d*modulo;
372     neuro[251][1]=(-1d)*4.5d*modulo;    neuro[251][2]=(-1d)*7d*modulo;
373     neuro[252][1]=(-1d)*5.5d*modulo;    neuro[252][2]=(-1d)*7d*modulo;
374     neuro[253][1]=(-1d)*6.5d*modulo;    neuro[253][2]=(-1d)*7d*modulo;
375     neuro[254][1]=(-1d)*7.5d*modulo;    neuro[254][2]=(-1d)*7d*modulo;
376     neuro[255][1]=(-1d)*8.5d*modulo;    neuro[255][2]=(-1d)*7d*modulo;
377     neuro[256][1]=0d;                   neuro[256][2]=(-1d)*8d*modulo;
378     neuro[257][1]=(-1d)*modulo;         neuro[257][2]=(-1d)*8d*modulo;
379     neuro[258][1]=(-1d)*2d*modulo;      neuro[258][2]=(-1d)*8d*modulo;
380     neuro[259][1]=(-1d)*3d*modulo;      neuro[259][2]=(-1d)*8d*modulo;
381     neuro[260][1]=(-1d)*4d*modulo;      neuro[260][2]=(-1d)*8d*modulo;
382     neuro[261][1]=(-1d)*5d*modulo;      neuro[261][2]=(-1d)*8d*modulo;
383     neuro[262][1]=(-1d)*6d*modulo;      neuro[262][2]=(-1d)*8d*modulo;
384     neuro[263][1]=(-1d)*7d*modulo;      neuro[263][2]=(-1d)*8d*modulo;
385     neuro[264][1]=(-1d)*8d*modulo;      neuro[264][2]=(-1d)*8d*modulo;
386     neuro[265][1]=(-1d)*0.5d*modulo;    neuro[265][2]=(-1d)*9d*modulo;
387     neuro[266][1]=(-1d)*1.5d*modulo;    neuro[266][2]=(-1d)*9d*modulo;
388     neuro[267][1]=(-1d)*2.5d*modulo;    neuro[267][2]=(-1d)*9d*modulo;
389     neuro[268][1]=(-1d)*3.5d*modulo;    neuro[268][2]=(-1d)*9d*modulo;
390     neuro[269][1]=(-1d)*4.5d*modulo;    neuro[269][2]=(-1d)*9d*modulo;
391     neuro[270][1]=(-1d)*5.5d*modulo;    neuro[270][2]=(-1d)*9d*modulo;
392     neuro[271][1]=(-1d)*6.5d*modulo;    neuro[271][2]=(-1d)*9d*modulo;
393     neuro[272][1]=(-1d)*7.5d*modulo;    neuro[272][2]=(-1d)*9d*modulo;
394     neuro[273][1]=(-1d)*8.5d*modulo;    neuro[273][2]=(-1d)*9d*modulo;
395     neuro[274][1]=0d;                   neuro[274][2]=(-1d)*10d*modulo;
396     neuro[275][1]=(-1d)*modulo;         neuro[275][2]=(-1d)*10d*modulo;
397     neuro[276][1]=(-1d)*2d*modulo;      neuro[276][2]=(-1d)*10d*modulo;
398     neuro[277][1]=(-1d)*3d*modulo;      neuro[277][2]=(-1d)*10d*modulo;
399     neuro[278][1]=(-1d)*4d*modulo;      neuro[278][2]=(-1d)*10d*modulo;
400     neuro[279][1]=(-1d)*5d*modulo;      neuro[279][2]=(-1d)*10d*modulo;
401     neuro[280][1]=(-1d)*6d*modulo;      neuro[280][2]=(-1d)*10d*modulo;
402     neuro[281][1]=(-1d)*7d*modulo;      neuro[281][2]=(-1d)*10d*modulo;
403     neuro[282][1]=(-1d)*8d*modulo;      neuro[282][2]=(-1d)*10d*modulo;
404
405     //Quartoo quadrante dos pontos dos favos de corpo centrado
406     neuro[283][1]=0.5d*modulo;          neuro[283][2]=(-1d)*nodulo;
407     neuro[284][1]=1.5d*modulo;          neuro[284][2]=(-1d)*nodulo;
408     neuro[285][1]=2.5d*modulo;          neuro[285][2]=(-1d)*nodulo;
409     neuro[286][1]=3.5d*modulo;          neuro[286][2]=(-1d)*nodulo;
410     neuro[287][1]=4.5d*modulo;          neuro[287][2]=(-1d)*nodulo;
411     neuro[288][1]=5.5d*modulo;          neuro[288][2]=(-1d)*nodulo;
412     neuro[289][1]=6.5d*modulo;          neuro[289][2]=(-1d)*nodulo;
413     neuro[290][1]=7.5d*modulo;          neuro[290][2]=(-1d)*nodulo;
414     neuro[291][1]=8.5d*modulo;          neuro[291][2]=(-1d)*nodulo;
415     neuro[292][1]=modulo;               neuro[292][2]=(-1d)*2d*nodulo;
416     neuro[293][1]=2d*modulo;            neuro[293][2]=(-1d)*2d*nodulo;
```

```
417     neuro[294][1]=3d*modulo;          neuro[294][2]=(-1d)*2d*modulo;
418     neuro[295][1]=4d*modulo;          neuro[295][2]=(-1d)*2d*modulo;
419     neuro[296][1]=5d*modulo;          neuro[296][2]=(-1d)*2d*modulo;
420     neuro[297][1]=6d*modulo;          neuro[297][2]=(-1d)*2d*modulo;
421     neuro[298][1]=7d*modulo;          neuro[298][2]=(-1d)*2d*modulo;
422     neuro[299][1]=8d*modulo;          neuro[299][2]=(-1d)*2d*modulo;
423     neuro[300][1]=0.5d*modulo;        neuro[300][2]=(-1d)*3d*modulo;
424     neuro[301][1]=1.5d*modulo;        neuro[301][2]=(-1d)*3d*modulo;
425     neuro[302][1]=2.5d*modulo;        neuro[302][2]=(-1d)*3d*modulo;
426     neuro[303][1]=3.5d*modulo;        neuro[303][2]=(-1d)*3d*modulo;
427     neuro[304][1]=4.5d*modulo;        neuro[304][2]=(-1d)*3d*modulo;
428     neuro[305][1]=5.5d*modulo;        neuro[305][2]=(-1d)*3d*modulo;
429     neuro[306][1]=6.5d*modulo;        neuro[306][2]=(-1d)*3d*modulo;
430     neuro[307][1]=7.5d*modulo;        neuro[307][2]=(-1d)*3d*modulo;
431     neuro[308][1]=8.5d*modulo;        neuro[308][2]=(-1d)*3d*modulo;
432     neuro[309][1]=modulo;             neuro[309][2]=(-1d)*4d*modulo;
433     neuro[310][1]=2d*modulo;          neuro[310][2]=(-1d)*4d*modulo;
434     neuro[311][1]=3d*modulo;          neuro[311][2]=(-1d)*4d*modulo;
435     neuro[312][1]=4d*modulo;          neuro[312][2]=(-1d)*4d*modulo;
436     neuro[313][1]=5d*modulo;          neuro[313][2]=(-1d)*4d*modulo;
437     neuro[314][1]=6d*modulo;          neuro[314][2]=(-1d)*4d*modulo;
438     neuro[315][1]=7d*modulo;          neuro[315][2]=(-1d)*4d*modulo;
439     neuro[316][1]=8d*modulo;          neuro[316][2]=(-1d)*4d*modulo;
440     neuro[317][1]=0.5d*modulo;        neuro[317][2]=(-1d)*5d*modulo;
441     neuro[318][1]=1.5d*modulo;        neuro[318][2]=(-1d)*5d*modulo;
442     neuro[319][1]=2.5d*modulo;        neuro[319][2]=(-1d)*5d*modulo;
443     neuro[320][1]=3.5d*modulo;        neuro[320][2]=(-1d)*5d*modulo;
444     neuro[321][1]=4.5d*modulo;        neuro[321][2]=(-1d)*5d*modulo;
445     neuro[322][1]=5.5d*modulo;        neuro[322][2]=(-1d)*5d*modulo;
446     neuro[323][1]=6.5d*modulo;        neuro[323][2]=(-1d)*5d*modulo;
447     neuro[324][1]=7.5d*modulo;        neuro[324][2]=(-1d)*5d*modulo;
448     neuro[325][1]=8.5d*modulo;        neuro[325][2]=(-1d)*5d*modulo;
449     neuro[326][1]=modulo;             neuro[326][2]=(-1d)*6d*modulo;
450     neuro[327][1]=2d*modulo;          neuro[327][2]=(-1d)*6d*modulo;
451     neuro[328][1]=3d*modulo;          neuro[328][2]=(-1d)*6d*modulo;
452     neuro[329][1]=4d*modulo;          neuro[329][2]=(-1d)*6d*modulo;
453     neuro[330][1]=5d*modulo;          neuro[330][2]=(-1d)*6d*modulo;
454     neuro[331][1]=6d*modulo;          neuro[331][2]=(-1d)*6d*modulo;
455     neuro[332][1]=7d*modulo;          neuro[332][2]=(-1d)*6d*modulo;
456     neuro[333][1]=8d*modulo;          neuro[333][2]=(-1d)*6d*modulo;
457     neuro[334][1]=0.5d*modulo;        neuro[334][2]=(-1d)*7d*modulo;
458     neuro[335][1]=1.5d*modulo;        neuro[335][2]=(-1d)*7d*modulo;
459     neuro[336][1]=2.5d*modulo;        neuro[336][2]=(-1d)*7d*modulo;
460     neuro[337][1]=3.5d*modulo;        neuro[337][2]=(-1d)*7d*modulo;
461     neuro[338][1]=4.5d*modulo;        neuro[338][2]=(-1d)*7d*modulo;
462     neuro[339][1]=5.5d*modulo;        neuro[339][2]=(-1d)*7d*modulo;
463     neuro[340][1]=6.5d*modulo;        neuro[340][2]=(-1d)*7d*modulo;
464     neuro[341][1]=7.5d*modulo;        neuro[341][2]=(-1d)*7d*modulo;
465     neuro[342][1]=8.5d*modulo;        neuro[342][2]=(-1d)*7d*modulo;
466     neuro[343][1]=modulo;             neuro[343][2]=(-1d)*8d*modulo;
467     neuro[344][1]=2d*modulo;          neuro[344][2]=(-1d)*8d*modulo;
468     neuro[345][1]=3d*modulo;          neuro[345][2]=(-1d)*8d*modulo;
469     neuro[346][1]=4d*modulo;          neuro[346][2]=(-1d)*8d*modulo;
470     neuro[347][1]=5d*modulo;          neuro[347][2]=(-1d)*8d*modulo;
471     neuro[348][1]=6d*modulo;          neuro[348][2]=(-1d)*8d*modulo;
472     neuro[349][1]=7d*modulo;          neuro[349][2]=(-1d)*8d*modulo;
473     neuro[350][1]=8d*modulo;          neuro[350][2]=(-1d)*8d*modulo;
474     neuro[351][1]=0.5d*modulo;        neuro[351][2]=(-1d)*9d*modulo;
475     neuro[352][1]=1.5d*modulo;        neuro[352][2]=(-1d)*9d*modulo;
476     neuro[353][1]=2.5d*modulo;        neuro[353][2]=(-1d)*9d*modulo;
477     neuro[354][1]=3.5d*modulo;        neuro[354][2]=(-1d)*9d*modulo;
478     neuro[355][1]=4.5d*modulo;        neuro[355][2]=(-1d)*9d*modulo;
479     neuro[356][1]=5.5d*modulo;        neuro[356][2]=(-1d)*9d*modulo;
480     neuro[357][1]=6.5d*modulo;        neuro[357][2]=(-1d)*9d*modulo;
481     neuro[358][1]=7.5d*modulo;        neuro[358][2]=(-1d)*9d*modulo;
482     neuro[359][1]=8.5d*modulo;        neuro[359][2]=(-1d)*9d*modulo;
483     neuro[360][1]=modulo;             neuro[360][2]=(-1d)*10d*modulo;
484     neuro[361][1]=2d*modulo;          neuro[361][2]=(-1d)*10d*modulo;
485     neuro[362][1]=3d*modulo;          neuro[362][2]=(-1d)*10d*modulo;
486     neuro[363][1]=4d*modulo;          neuro[363][2]=(-1d)*10d*modulo;
487     neuro[364][1]=5d*modulo;          neuro[364][2]=(-1d)*10d*modulo;
488     neuro[365][1]=6d*modulo;          neuro[365][2]=(-1d)*10d*modulo;
489     neuro[366][1]=7d*modulo;          neuro[366][2]=(-1d)*10d*modulo;
```

```
490     neuro[367][1]=8d*modulo;            neuro[367][2]=(-1d)*10d*modulo;
491
492
493     //Cata o parâmetro submetido para a pag. html para o # de
        entradas
494     try {
495       if (getParameter("tam") != null){
496        nEntradas=Integer.parseInt(getParameter("tam"));
497       }
498     }
499     catch (Exception e) {
500       System.out.println(
        "Usuário colocou valor não válido de #Entradas, então uso do
        default.");
501       nEntradas=100;
502     }
503
504     try {
505       if (getParameter("mta1") != null){
506        nVezesPassAmostra=Integer.parseInt(getParameter("mta1"));
507       }
508     }
509     catch (Exception e) {
510       System.out.println(
        "Usuário colocou valor não válido de #Iterações, então uso do
        default.");
511       nVezesPassAmostra=1500;
512     }
513
514     try {
515       if (getParameter("mta2") != null){
516        nVezesPassAjusteFino=Integer.parseInt(getParameter("mta2"));
517       }
518     }
519     catch (Exception e) {
520       System.out.println(
        "Usuário colocou valor não válido de #AjusteFino, então uso do
        default.");
521       nVezesPassAjusteFino=1000;
522     }
523
524     //Cata os parâmetros de formulário para uso dos métodos
        calcMesmoX()
525
526     //pega o parâmetro de applet "taxa de aprendizado inicial"
527     try {
528       if (getParameter("mat") != null){
529         ETA_INI=Double.parseDouble(getParameter("mat"));
530         if (ETA_INI>5 || ETA_INI<0.01) { ETA_INI=0.22d; }
531       }
532       else {
533         ETA_INI=0.22d;
534       }
535     }
536     catch (Exception e) {
537       System.out.println(
        "Usuário colocou valor não válido, então uso do default.");
538       ETA_INI=0.22d;
539     }
540
541     //pega o parâmetro de applet "vizinhança inicial"
542     try {
543       if (getParameter("atm1") != null){
544         R_INI=Double.parseDouble(getParameter("atm1"));
545         if (R_INI>100.0 || R_INI<3.0) { R_INI=modulo*54d; }
546       }
547       else {
548         R_INI=modulo*54d;
549       }
550     }
551     catch (Exception e) {
552       System.out.println(
        "Usuário colocou valor não válido, então uso do default.");
```

```
553         R_INI=modulo*54d;
554       }
555
556       //pega o parâmetro de applet "ajuste de espalhamento"
557       try {
558         if (getParameter("atm2") != null){
559           CR=Double.parseDouble(getParameter("atm2"));
560         }
561         else {
562           CR=1.40d;
563         }
564       }
565       catch (Exception e) {
566         System.out.println(
          "Usuário colocou valor não válido, então uso do default.");
567         CR=1.40d;
568       }
569
570      //final da pegada de parâmetros de formulário para uso de
         calcMesmoX()
571
572       entrada = new String[nMaximoDados+1][11];
573       for (int i=0; i<=nMaximoDados; i++) {
574          for (int j=0; j<=10; j++) {
575              entrada[i][j] = "";
576          }
577       }
578
579       qualy = new String[nMaximoDados+1][9];
580       for (int i=0; i<=nMaximoDados; i++) {
581          for (int j=0; j<=8; j++) {
582              qualy[i][j] = "";
583          }
584       }
585
586       try {
587         leInformacao();
588       }
589       catch (Exception e) {
590         System.err.println(e);
591         System.out.println("Não realizou o método leInformacao()");
592       }
593
594
595       //instanciação das variáveis
596       w = new double[nPesos+1][dim+1]; w_inic = new double[nPesos+1][
          dim+1];
597       x_d = new double[nMaximoDados+1][dim+1+2];//duas dimensões extras
          para os PCSs,necessário para avaliação
598       avalia_d = new int[nMaximoDados+1];
599       maisPerto = new int[nPesos+1];
600       v = new int[nEntradas+1];
601       or = new int[nEntradas+1];
602       media = new double[nPontos+1][dim+1];
603       desvioPadraoSignif = new double[nPontos+1][dim+1];
604       desvioPadraoNat = new double[nPontos+1][dim+1];
605       dPAlgarismoSignificativo = new int[nPontos+1][dim+1];
606       aleatorio = new Random();
607
608       contaParaNeuro = new double[nPesos+1];
609       qNeuro = new int[nPesos+1];
610       alarme = new String[nEntradas+1];
611       campoQ = new TextField[nPontos];
612       campoP = new TextField[nPontos];
613       grandezaQ = new Label[nPontos];
614       grandezaP = new Label[nPontos];
615
616       prevDimAux = new double[16][100];
617
618       setBackground (new Color(250,247,153));
619
620       //monta "formulário" da interface grafica
621       for (int i=0; i<nPontos; i++) {
```

```
622         grandezaQ[i]=new Label("QDO "+ (i+1));
623         grandezaP[i]=new Label("POS "+ (i+1));
624         //grandezaP[i]=new Label("Press "+ (i-1));
625       }
626       for (int i=0; i<nPontos; i++) {
627         campoQ[i]=new TextField(""); campoQ[i].setColumns(5);
628         campoP[i]=new TextField(""); campoP[i].setColumns(3);
629       }
630       etiqueta1=new Label("   Iteracão");
631       etiqueta2=new Label("0      ");
632       aviso1=new Label("Medida número ");
633       aviso2=new Label("Selecione Click");
634       espaco0=new Label("      ");
635       espaco1=new Label("<   >",Label.RIGHT);
636       espaco2=new Label("   ||   ",Label.LEFT);
637       espaco3=new Label("|    |",Label.LEFT);
638       espaco4=new Label("<   <",Label.LEFT);
639       selecao=new TextField(""); selecao.setColumns(3);
640       selection=new TextField(""); selection.setColumns(1);
641       atribui=new Button("AtribuiUnidade");
642       apaga=new Button("eApague");
643       rest=new Button("Restaura");
644       atribuicao=new Button("UnidadeAoClick");
645       mapeia=new Button("Mapeia");
646       preenche=new Button("Acende");
647       mudaBG=new Button("MudaFundo");
648       etiqueta3=new Label("# de medidas (Treinamento + Teste)  = ");
649       etiqueta4=new Label("0      ");
650
651       setLayout (new FlowLayout());
652
653       for (int i=0; i<nPontos; i++) {
654         add(grandezaQ[i]);
655         add(campoQ[i]);
656         add(grandezaP[i]);
657         add(campoP[i]);
658         add(espaco2);
659       }
660
661       add(mapeia);
662
663       add(espaco0);
664       add(etiqueta1);
665       add(etiqueta2);
666
667       add(aviso1);
668       add(selecao);
669       add(atribui);
670
671       add(espaco3);
672       add(preenche);
673
674       add(espaco1);
675       add(rest);
676       add(espaco4);
677
678       add(aviso2);
679       add(selection);
680       add(apaga);
681       add(espaco0);
682       add(atribuicao);
683
684       add(espaco2);
685       add(mudaBG);
686
687       add(espaco0);
688       add(etiqueta3);
689       add(etiqueta4);
690
691      //atribuição default para as variaveis
692      for (int k=0; k <= nEntradas; k++){
693        v[k]=0;
694      }
```

```
695
696     for (int k=0; k <= nEntradas; k++){
697       pr[k]=0;
698     }
699
700     for (int i=0; i<=nPesos; i++) {
701       contaParaNeuro[i]=1d;
702     }
703
704     for (int i=0; i<=nPesos; i++) {
705       qNeuro[i]=1;
706     }
707
708     for (int k=0; k <= nEntradas; k++){
709       alarme[k]=" ";
710     }
711
712
713
714
715     //registra os manipuladores de evento de mouse
716     addMouseListener (new MouseHandler());
717
718     //registra os botões para a ação manipular
719     atribui.addActionListener(this);
720     apaga.addActionListener(this);
721     rest.addActionListener(this);
722     atribuicao.addActionListener(this);
723     mapeia.addActionListener(this);
724     preenche.addActionListener(this);
725     mudaBG.addActionListener(this);
726
727
728   }/*chave final do método init*/
729
730
731
732   //parte a thread (linha de processo)
733   public void start() {
734       thread = new Thread(this);
735       thread.start();
736   }/*final do método start*/
737
738
739
740   //roda continuamente o seu código
741   public void run() {
742
743    while(autorizacao.equals("ande")) {/*'while' externo*/
744       try {
745        if (ii >= nEntradas) {//paintFinal
746          while(ii<=nEntradas+nVezesPassAmostra+nVezesPassAjusteFino)
           {/*'while' interno1*/
747
748           while(ii<=nEntradas+nVezesPassAmostra) {/*'while'
            interno2*/
749
750             if (ii > nEntradas) {
751               calcMesmo();
752               if (Math.abs(ii % 100)==0) {
753                 repaint();
754                 etiqueta2.setText(String.valueOf(iteracione));
755                 if (iteracione < 300) {
756                   thread.sleep((int)Math.rint(800*Math.exp((-1.0d)*
                    iteracione)/170)));
757                 }
758                 else {
759                   thread.sleep( (int)Math.rint(800*Math.exp((-1.0d)*
                    1.68d)) );
760                 }
761               }
762
763             if (ii == nEntradas) {
```

```
764                 calcSistema();
765                 repaint(); thread.sleep(2500);
766               }
767               ii++;
768
769             }/*final do while interno2 de paintFinal*/
770
771             calcMesmo2();
772             if (Math.abs(iteracione % 500)==0) {
773               repaint();
774               etiqueta2.setText(String.valueOf(iteracione));
775               thread.sleep( (int)Math.rint(800*Math.exp(-1.0d)*1.68d))
                  );
776             }
777             ii++;
778
779           }/*final do while interno1 de paintFinal*/
780           if (ii == nEntradas+nVezesPassAmostra+nVezesPassAjusteFino+1)
              {
781             for (int i=1; i<=nPesos; i++) {
782              if (maisPerto[i]>0) {
783               System.out.println(
                  "O dado mais próximo da unidade neural "+ i +
                  " é a linha " + (maisPerto[i]) /*+ " com a " +
                  (2*maisPerto[i])*/ + ".");
784              }
785              else {
786               System.out.println(
                  "Não foi atribuido nenhum dado para a unidade neural " +
                  i + ".");
787              }
788             }
789           ii++;
790           }
791         }/*final do if externo de paintFinal*/
792         if (bandeiraPisca.equals("up")) { bandeiraPisca="down"; }
793         else { bandeiraPisca = "up"; }
794         repaint();
795         thread.sleep(800);
796       }/*fim do try de Interrupted*/
797       catch (InterruptedException e) {
798         System.out.println("Algo parou a contagem"); break;
799       }
800     }/*fim do 'while' externo*/
801
802   }/*fim de run*/
803
804
805
806   //desenha dinamicamente o conteúdo em rodagem
807   public void paint (Graphics g) {
808
809       double[] fracaoParaNeuro = new double[nPesos+1];
810       String[][] recipiente = new String[nPesos][dim];
811       String[] ingredientes;
812       ingredientes = new String[nPesos];
813       String[] abrevGrandeza =
          {"NUL","LBL","V","Q","P","I","D","NUL","NUL","NUL"};
814       String[] abrevUnidade = {"NUL"," ","m³","kcal/m³","kgf/cm²",
          "°C","","NUL","NUL","NUL"};
815       //declaração das cores utilizadas para aprendizagem e
          classificação
816       Color blackAndWhite = new Color(120,120,120);//darkrev
817       Color greedark = (Color.green).darker(); Color bludark =
          (Color.blue).darker();Color orchidea = new Color (218,112,214);
818       Color[] corValor =
          {Color.white,greedark,Color.orange,Color.red,orchidea,Color.
          black};
819
820       //função que rastreia a cor de fundo escolhida (traz também
          algumas mudancas qto. ao tipo de variável exibida)
821       fixaBackground();
822
```

```
823         //VALIDAÇÃO DA APRENDIZAGEM POR CLICKS NA TELA ONDE SÃO
            PLOTADOS OS DADOS DE ENTRADA
824         for (int i=0; i<prev.size(); i++) {
825          g.setColor(blackAndWhite);//g.setColor(cor[or[i+1]]);
826          g.drawString(".V"+String.valueOf(i+1)+"|"+or[i+1]+" "+alarme[i
             +1],(Point)prev.elementAt(i)).x,(Point)prev.elementAt(i)).y)
             ;
827          g.fillOval( (Point)prev.elementAt(i)).x,(Point)prev.
             elementAt(i)).y,(int)Math.rint( (prevDimAux[0][i])+ld)*8d ),(
             int)Math.rint( (prevDimAux[1][i])+ld)*8d ) );
828         }
829
830         //LINHAS SEPARADORAS DE SUBTELAS NO APPLET
831         g.setColor(Color.black);
832         g.fillRect(0,100,800,3);
833         g.fillRect(0,2050,800,3);
834
835
836         //MÓDULO DE PREENCHIMENTO DAS CÉLULAS - para plano
837         int APPUPLEFT_X = 380;
838         int APPUPLEFT_Y = 460;
839         int DIAMETER = 38;
840         int DIAGLIT_IN = 12;
841         int DIAGLIT_ON = 28;
842         for (int l=1; l<=nPesos; l++) {
843           g.setColor(blackAndWhite);
844           g.setFont( new Font("Serif",Font.PLAIN,9) );
845           if (enche.equals("nao")) {//Cuidado! o fillOval abaixo se
              torna ponto quando não há nenhum dado atribuido a uma unidade
              neural
846            g.fillOval((int)Math.rint((neuro[l][1]*20)+APPUPLEFT_X+(
               DIAMETER/2)),(int)Math.rint((neuro[l][2]*20)+APPUPLEFT_Y+(
               DIAMETER/2)),(int)Math.round(contaParaNeuro[l]),(int)Math.
               round(contaParaNeuro[l]));
847           }
848           g.setColor(Color.black);
849           g.drawString( String.valueOf(l),(int)Math.rint((neuro[l][1]*
              20)+APPUPLEFT_X+(DIAMETER/2)),(int)Math.rint((neuro[l][2]*20)
              +APPUPLEFT_Y+(DIAMETER/2)) );
850           g.setColor(corValor[avalia_d[maisPerto[l]]]);
851           if (enche.equals("nao")) {
852
               g.drawOval((int)Math.rint((neuro[l][1]*20)
               +APPUPLEFT_X),(int)Math.rint((neuro[l][2]*20)+APPUPLEFT_Y),
               DIAMETER,DIAMETER);
853           }
854           else {
855
               g.fillOval((int)Math.rint((neuro[l][1]*20)
               +APPUPLEFT_X),(int)Math.rint((neuro[l][2]*20)+APPUPLEFT_Y),
               DIAMETER,DIAMETER);
856            g.setFont( new Font("Serif",Font.PLAIN,9) );
857            g.setColor(Color.black);
858            switch (cicliceBG) {//preenche o conteúdo das células com
               #dado, volume e pos respectivamente
859             case 1:
860              try {
861               if (entrada[maisPerto[l]][2].equals("")) {
               //x_d[maisPerto[l]][0] -> número seqüencial do dado
862             g.drawString( String.valueOf((int)Math.rint(x_d[
                maisPerto[l]][0])),(int)Math.rint((neuro[l][1]*20)+
                APPUPLEFT_X+15),(int)Math.rint((neuro[l][2]*20)+
                APPUPLEFT_Y+(DIAMETER/2)) );
863               }
864               else {
865             g.drawString( String.valueOf((int)Math.rint(Double.
                valueOf(entrada[maisPerto[l]][2]).doubleValue()))+"|"+
                String.valueOf((int)Math.rint(x_d[maisPerto[l]][0])),
866
                (int)Math.rint((neuro[l][1]*20)+APPUPLEFT_X+5),(int)
                Math.rint((neuro[l][2]*20)+APPUPLEFT_Y+(DIAMETER/2)) );
867               }
868              }
```

```
869             catch (Exception e) {
870              g.drawString( String.valueOf((int)Math.rint(x_d[maisPerto
                 [1][0])),(int)Math.rint((neuro[1][1]*20)+APPUPLEFT_X+15)
                 ,(int)Math.rint((neuro[1][2]*20)+APPUPLEFT_Y+(DIAMETER/2
                 )) );
871              }
872             break;
873            case 2:
874             g.setFont( new Font("Serif",Font.PLAIN,9) );
875             try {
876              if (entrada[maisPerto[1]][2].equals("")) {
                 //qualy[maisPerto[1]][3]->volume normalizado
877              g.drawString( qualy[maisPerto[1]][3],(int)Math.rint((
                 neuro[1][1]*20)+APPUPLEFT_X),(int)Math.rint((neuro[1][2]
                 *20)+APPUPLEFT_Y+(DIAMETER/2)) );
878              }
879              else {
880               g.drawString( String.valueOf((int)Math.rint(Double.
                 valueOf(qualy[maisPerto[1][3]).doubleValue())),
881
                 (int)Math.rint((neuro[1][1]*20)+APPUPLEFT_X+3),(int)
                 Math.rint((neuro[1][2]*20)-APPUPLEFT_Y+(DIAMETER/2)) );
882              }
883             }
884             catch (Exception e) {
885              g.drawString( qualy[maisPerto[1]][3],(int)Math.rint((
                 neuro[1][1]*20)+APPUPLEFT_X),(int)Math.rint((neuro[1][2]*
                 20)+APPUPLEFT_Y+(DIAMETER/2)) );
886              }
887             break;
888            case 3:
889             g.setFont( new Font("Serif",Font.PLAIN,9) );
890             try {
891              if (entrada[maisPerto[1]][2].equals("")) {
                 //qualy[maisPerto[1]][4]->PCS,
                 qualy[maisPerto[1]][5]->pressão
892              g.drawString( qualy[maisPerto[1]][4],(int)Math.rint((
                 neuro[1][1]*20)+APPUPLEFT_X-3),(int)Math.rint((neuro[1][
                 2]*20)+APPUPLEFT_Y+(DIAMETER/2)) );
893              }
894              else {
895               g.drawString( String.valueOf((int)Math.rint(Double.
                 valueOf(entrada[maisPerto[1][2]).doubleValue()))+"]"+
                 String.valueOf((int)Math.rint(Double.valueOf(qualy[
                 maisPerto[1][4]).doubleValue())),
896
                 (int)Math.rint((neuro[1][1]*20)+APPUPLEFT_X+3),(int)
                 Math.rint((neuro[1][2]*20)+APPUPLEFT_Y+(DIAMETER/2)) );
897              }
898             }
899             catch (Exception e) {
900              g.drawString( qualy[maisPerto[1]][4],(int)Math.rint((
                 neuro[1][1]*20)+APPUPLEFT_X-3),(int)Math.rint((neuro[1][2
                 ]*20)+APPUPLEFT_Y+(DIAMETER/2)) );
901              }
902             break;
903            }//final do switch ciclico background
904           }//final do else de enche-não
905          if (l == pesoVictor) {
906           if (bandeiraAcende.equals("sim") && pesoVictor>0) {
907             if (enche.equals("sim")) {
908               g.setColor(Color.white);
909
                 g.fillOval((int)Math.rint((neuro[1][1]*20)
                 +APPUPLEFT_X),(int)Math.rint((neuro[1][2]*20)+
                 APPUPLEFT_Y),DIAMETER,DIAMETER);
910              g.setColor(corValor[avalia_d[maisPerto[1]]]);
911              }
912             if (bandeiraPisca.equals("down")) {
913               if (enche.equals("sim")) {
914                 g.fillOval((int)Math.rint((neuro[1][1]*20)+
                    APPUPLEFT_X+(DIAMETER/2)-(DIAGLIT_IN/2)),(int)Math.
                    rint((neuro[1][2]*20)+APPUPLEFT_Y+(DIAMETER/2)-(
```

```
914                 DIAGLIT_IN/2)),DIAGLIT_IN,DIAGLIT_IN);
915               }
916               else {
917                 g.drawOval((int)Math.rint((neuro[l][1]*20)+
                    APPUPLEFT_X+(DIAMETER/2)-(DIAGLIT_IN/2)),(int)Math.
                    rint((neuro[l][2]*20)+APPUPLEFT_Y+(DIAMETER/2)-(
                    DIAGLIT_IN/2)),DIAGLIT_IN,DIAGLIT_IN);
918               }
919             }
920             else {
921               if (enche.equals("sim")) {
922                 g.fillOval((int)Math.rint((neuro[l][1]*20)+
                    APPUPLEFT_X+(DIAMETER/2)-(DIAGLIT_ON/2)),(int)Math.
                    rint((neuro[l][2]*20)+APPUPLEFT_Y+(DIAMETER/2)-(
                    DIAGLIT_ON/2)),DIAGLIT_ON,DIAGLIT_ON);
923               }
924               else {
925                 g.drawOval((int)Math.rint((neuro[l][1]*20)+
                    APPUPLEFT_X+(DIAMETER/2)-(DIAGLIT_ON/2)),(int)Math.
                    rint((neuro[l][2]*20)+APPUPLEFT_Y+(DIAMETER/2)-(
                    DIAGLIT_ON/2)),DIAGLIT_ON,DIAGLIT_ON);
926               }
927             }
928           }
929         }
930       }//final do MÓDULO DE PREENCHIMENTO DAS CÉLULAS - para plano
          (do for)
931
932       /*//MODULO DE PREENCHIMENTO DAS CELULAS - para superfície
          sphérica
933       int APPUPLEFT_X = 960;
934       int APPUPLEFT_Y = 400;
935       int APPUPLEFT_YDOWN = 700;
936       int DIAMETER = 42;
937       int DIAGLIT_IN = 10;
938       int DIAGLIT_ON = 30;
939       for (int l=1; l<=nPesos; l++) {
940
            g.setColor(corValor[avalia_d[maisPerto[l]]]);//g.setColor(co
            r[l]);
941         if (neuro[l][1]<=Math.PI && Math.abs(neuro[l][2])<Math.PI/2)
            {
942
             g.fillOval((int)Math.rint(((neuro[l][0]*Math.cos(neuro[l]
             [1]))*20)+APPUPLEFT_X),(int)Math.rint(((neuro[l][0]
             *(Math.sin(neuro[l][2])))*20)+APPUPLEFT_Y),(int)
             Math.round(contaParaNeuro[l]),(int)
             Math.round(contaParaNeuro[l]));
943          g.drawString(
             String.valueOf(l),(int)Math.rint(((neuro[l][0]
             *Math.cos(neuro[l][1]))*20)+APPUPLEFT_X),(int)
             Math.rint(((neuro[l][0]*(Math.sin(neuro[l][2])))*20)
             +APPUPLEFT_Y) );
944
             g.setColor(corValor[avalia_d[maisPerto[l]]]);//g.setColor(C
             olor.white);
945
             g.drawOval((int)Math.rint(((neuro[l][0]*Math.cos(neuro[l]
             [1]))*20)+APPUPLEFT_X-(DIAMETER/2)),(int)
             Math.rint(((neuro[l][0]*(Math.sin(neuro[l][2])))*20)
             +APPUPLEFT_Y-(DIAMETER/2)),DIAMETER,DIAMETER);
946          if (l == pesoVictor) {
947            if (bandeiraAcende.equals("sim") && pesoVictor>0) {
948              if (bandeiraPisca.equals("down")) {
949
                   g.drawOval((int)Math.rint(((neuro[l][0]
                   *Math.cos(neuro[l][1]))*20)
                   +APPUPLEFT_X-(DIAGLIT_IN/2)),(int)
                   Math.rint(((neuro[l][0]*(Math.sin(neuro[l][2])))*20)
                   +APPUPLEFT_Y-(DIAGLIT_IN/2)),DIAGLIT_IN,DIAGLIT_IN);
950              }
951              else {
952
```

```
952                 g.drawOval((int)Math.rint(((neuro[l][0]
                    *Math.cos(neuro[l][1]))*20)
                    +APPUPLEFT_X-(DIAGLIT_ON/2)),(int)
                    Math.rint((neuro[l][0]*(Math.sin(neuro[l][2]))))*20)
                    +APPUPLEFT_Y-(DIAGLIT_ON/2)),DIAGLIT_ON,DIAGLIT_ON);
953               }
954             }
955           }
956         }
957         else {
958          if (Math.abs(neuro[l][2])==Math.PI/2) {
959
             g.fillOval((int)Math.rint(((neuro[l][0]*Math.sin(neuro[l]
             [1]))*20)+APPUPLEFT_X),(int)Math.rint((neuro[l][0]
             *(Math.sin(neuro[l][2]))))*20)+APPUPLEFT_Y),(int)
             Math.round(contaParaNeuro[l]),(int)
             Math.round(contaParaNeuro[l]));
960          g.drawString(
             String.valueOf(l),(int)Math.rint(((neuro[l][0]
             *Math.sin(neuro[l][1]))*20)+APPUPLEFT_X),(int)
             Math.rint((neuro[l][0]*(Math.sin(neuro[l][2]))))*20)
             +APPUPLEFT_Y) );
961
             g.setColor(corValor[avalia_d[maisPerto[l]]]);//g.setColor(
             Color.white);
962
             g.drawOval((int)Math.rint(((neuro[l][0]*Math.sin(neuro[l]
             [1]))*20)+APPUPLEFT_X-(DIAMETER/2)),(int)
             Math.rint((neuro[l][0]*(Math.sin(neuro[l][2]))))*20)
             +APPUPLEFT_Y-(DIAMETER/2)),DIAMETER,DIAMETER);
963             if (l == pesoVictor) {
964               if (bandeiraAcende.equals("sim") && pesoVictor>0) {
965                 if (bandeiraPisca.equals("down")) {
966
                    g.drawOval((int)Math.rint(((neuro[l][0]
                    *Math.sin(neuro[l][1]))*20)
                    +APPUPLEFT_X-(DIAGLIT_IN/2)),(int)
                    Math.rint((neuro[l][0]*(Math.sin(neuro[l][2]))))*20)
                    +APPUPLEFT_Y-(DIAGLIT_IN/2)),DIAGLIT_IN,DIAGLIT_IN);
967                 }
968                 else {
969
                    g.drawOval((int)Math.rint(((neuro[l][0]
                    *Math.sin(neuro[l][1]))*20)
                    +APPUPLEFT_X-(DIAGLIT_ON/2)),(int)
                    Math.rint((neuro[l][0]*(Math.sin(neuro[l][2]))))*20)
                    +APPUPLEFT_Y-(DIAGLIT_ON/2)),DIAGLIT_ON,DIAGLIT_ON);
970                 }
971               }
972             }
973           }
974           else {
975
             g.fillOval((int)Math.rint(((neuro[l][0]*Math.cos(neuro[l]
             [1]))*20)+APPUPLEFT_X),(int)Math.rint((neuro[l][0]
             *(Math.sin(neuro[l][2]))))*20)+APPUPLEFT_YDOWN),(int)
             Math.round(contaParaNeuro[l]),(int)
             Math.round(contaParaNeuro[l]));
976          g.drawString(
             String.valueOf(l),(int)Math.rint(((neuro[l][0]
             *Math.cos(neuro[l][1]))*20)+APPUPLEFT_X),(int)
             Math.rint((neuro[l][0]*(Math.sin(neuro[l][2]))))*20)+700)
             );
977
             g.setColor(corValor[avalia_d[maisPerto[l]]]);//g.setColor(
             Color.white);
978
             g.drawOval((int)Math.rint(((neuro[l][0]*Math.cos(neuro[l]
             [1]))*20)+APPUPLEFT_X-(DIAMETER/2)),(int)
             Math.rint((neuro[l][0]*(Math.sin(neuro[l][2]))))*20)
             +APPUPLEFT_YDOWN-(DIAMETER/2)),DIAMETER,DIAMETER);
979             if (l == pesoVictor) {
980               if (bandeiraAcende.equals("sim") && pesoVictor>0) {
```

```
981                 if (bandeiraPisca.equals("down")) {
982
                      g.drawOval((int)Math.rint(((neuro[1][0]
                      *Math.cos(neuro[1][1]))*20)
                      +APPUPLEFT_X-(DIAGLIT_IN/2)),(int)
                      Math.rint(((neuro[1][0]*(Math.sin(neuro[1][2])))*20)
                      +APPUPLEFT_YDOWN-(DIAGLIT_IN/2)),DIAGLIT_IN,DIAGLIT_
                      IN);
                    }
983                 else {
984
985                   g.drawOval((int)Math.rint(((neuro[1][0]
                      *Math.cos(neuro[1][1]))*20)
                      +APPUPLEFT_X-(DIAGLIT_ON/2)),(int)
                      Math.rint(((neuro[1][0]*(Math.sin(neuro[1][2])))*20)
                      +APPUPLEFT_YDOWN-(DIAGLIT_ON/2)),DIAGLIT_ON,DIAGLIT_
                      ON);
                    }
986                 }
987               }
988             }
989           }
990         }
991       }//final do MÓDULO DE PREENCHIMENTO DAS CÉLULAS - para
          superfície sphérica (do for)*/
992
993
994       //DESENHA A LISTA DOS RESULTADOS PARA CADA CÉLULA
995       try {//tryImp - per calculatarum positionibus imprentione
996        if (ii > nEntradas)/*solo per non imprentare valores defalta*/
          //paintFinal
997
998         //imprime dados de entrada e pesos
999         for (int i=0; i<nEntradas; i++) {//dados de entrada
1000          g.setColor(blackAndWhite);//g.setColor(cor[v[i+1]]);
1001          g.drawString(  ".x"+String.valueOf(i+1)+"|"+v[i+1],(int)
              Math.rint( (350d*(x_d[i+1][1]+1d))+50d ),(int)Math.rint(
              2800d-325d*(x_d[i+1][2]+1d) )  );
1002          if ( ((int)Math.round((x_d[i+1][3]+1d)*8)) > 0 && ((int)
              Math.round((x_d[i+1][6]+1d)*8)) > 0 ) {//correção para que
              a elipse não tenha algum semieixo = 0, x_d mais sensível a
              tal
1003           g.fillOval( (int)Math.rint((350d*(x_d[i+1][1]+1d))+50d),(
               int)Math.rint(2800d-325d*(x_d[i+1][2]+1d)),(int)Math.
               round((x_d[i+1][3]+1d)*8),(int)Math.round((x_d[i+1][6]+1d
               )*8) );
1004          }
1005          else {
1006           if ( ((int)Math.round((x_d[i+1][3]+1d)*8)) <= 0 ) {
1007            g.fillOval( (int)Math.rint((350d*(x_d[i+1][1]+1d))+50d),
                (int)Math.rint(2800d-325d*(x_d[i+1][2]+1d)),1,(int)Math.
                round((x_d[i+1][6]+1d)*8) );
1008           }
1009           else {
1010            g.fillOval( (int)Math.rint((350d*(x_d[i+1][1]+1d))+50d),
                (int)Math.rint(2800d-325d*(x_d[i+1][2]+1d)),(int)Math.
                round((x_d[i+1][3]+1d)*8),1 );
1011           }
1012          }
1013        }
1014
1015        for (int i=1;i<=nPesos;i++) {//valores iniciais aleatórios
            dos vectores pesos
1016          g.setColor(blackAndWhite);
1017          g.drawString(  ".0"+String.valueOf(i),(int)Math.rint( (350
              d*(w_inic[i][1]+1d))+50d ),(int)Math.rint( 2800d - 325d*(
              w_inic[i][2]+1d) )  );
1018          g.fillOval( (int)Math.floor((350d*(w_inic[i][1]+1d))+50d),
              (int)Math.floor(2800d - 325d*(w_inic[i][2]+1d)),(int)Math.
              rint((w_inic[i][3]+1d)*8),(int)Math.rint((w_inic[i][6]+1d)
              *8) );
1019        }
1020
1021        g.setFont( new Font("Helvetica",Font.BOLD,15) );
```

```
1022          for (int i=1;i<=nPesos;i++) {//valores finais dos vectores
              pesos a cada grupo de iteração
1023            g.setColor(blackAndWhite);
1024            g.drawString(  ".@"+String.valueOf(i),(int)Math.rint( (350
                d*(w[i][1]+1d))+50d ),(int)Math.rint( 2800d -
                325d*(w[i][2]+1d) )  );
1025            g.fillOval( (int)Math.rint((350d*(w[i][1]+1d))+50d),(int)
                Math.rint(2800d - 325d*(w[i][2]+1d)),(int)Math.rint((w[i][
                3]+1d)*8),(int)Math.rint((w[i][6]+1d)*8) );
1026          }
1027
1028          //lista das coordenadas relativas dos pesos e lista das
              médias de cada componente de entrada
1029          for(int i=1;i<=nPesos;i++) {
1030            fracaoParaNeuro[i] = ( (double)Math.rint(((contaParaNeuro[
                i-1)/(nEntradas*2))*1000) )/1000;
1031          }
1032          DecimalFormat digits = new DecimalFormat("###.###");
1033          for (int i=1;i<=nPesos;i++){
1034            ingredientes[i-1]=
1035            "w"+String.valueOf(i)+"=["+String.valueOf( (int)Math.rint(
                (350d*(w[i][1]+1d)-350d)/10 ) )+" , "
1036            +String.valueOf( (int)Math.rint( (350d*(w[i][2]+1d)-350d)/
                10 ) )+" , "
1037            +String.valueOf( (int)Math.rint( (350d*(w[i][3]+1d)-350d)/
                10 ) )+" , "
1038            +String.valueOf( (int)Math.rint( (350d*(w[i][4]+1d)-350d)/
                10 ) )+" , "
1039            +String.valueOf( (int)Math.rint( (350d*(w[i][5]+1d)-350d)/
                10 ) )+" , "
1040            +String.valueOf( (int)Math.rint( (350d*(w[i][6]+1d)-350d)/
                10 ) )+" ] "+" "+digits.format(fracaoParaNeuro[i]);
1041          }
1042
1043          g.setFont( new Font("Serif",Font.PLAIN,10) );
1044          FontMetrics fm = g.getFontMetrics();
1045          for (int j=0;j<nPesos;j++) {
1046            g.setColor(blackAndWhite);//g.setColor(cor[j+1]);
1047            if (j<=99) {
1048
                    g.drawString(ingredientes[j],5,(fm.getAscent()
                    *(j+1))+950);
1049            }
1050            else {
1051              if (j<=199) {
1052                g.drawString(ingredientes[j],195,(fm.getAscent()
                    *(j-99))+950);
1053              }
1054              else {
1055                if (j<=299) {
1056                  g.drawString(ingredientes[j],385,(fm.
                      getAscent()*(j-199))+950);
1057                }
1058                else {
1059                  g.drawString(ingredientes[j],575,(fm.
                      getAscent()*(j-299))+950);
1060                }
1061              }
1062            }
1063          }
1064          DecimalFormat digitos = new DecimalFormat("0");
1065          for (int j=1;j<=dim;j++) {
1066            for (int k=1;k<=nPontos;k++) {
1067              if(desvioPadraoSignif[k][j]>=1d) {
1068                recipiente[k-1][j-1]="Média"+String.valueOf(k)+","+
                  abrevGrandeza[j]+" = "
1069                +String.valueOf(digitos.format(media[k][j]))+"("+
                  String.valueOf(digitos.format(dPAlgarismoSignificativo
                  [k][j]))+") "
1070                +abrevUnidade[j];
1071              }
1072              else {
1073                recipiente[k-1][j-1]="Média"+String.valueOf(k)+","+
```

```
1073                abrevGrandeza[j]+" = "
1074                +digits.format(media[k][j])+"("+digitos.format(
                    dPAlgarismoSignificativo[k][j])+") "
1075                +abrevUnidade[j];
1076              }
1077            }
1078          }
1079          g.setFont( new Font("Times New Roman",Font.BOLD,11) );
1080          g.setColor(Color.gray);
1081          for (int j=2;j<=dim;j++) {
1082          for (int k=1;k<=nPontos;k++) {
1083              g.drawString(String.valueOf(recipiente[k-1][j-1]),3+168*
                    (j-2),(fm.getAscent()*(k))+950-40);
1084            }
1085          }
1086
1087        }/* endif da impressão dos resultados no método paint (o 'if'
                    paintFinal)*/
1088      }//final de tryImp
1089      catch(Exception e) {
1090       System.out.println(
          "Verifique tryImp. Verifique as atribuições de dimensão em
                    tryImp");
1091      }//final de catchImp e final da lista de resultados para os
                    pontos (células)
1092
1093    }/*chave final do método paint*/
1094
1095
1096
1097    //evento clica mouse sobre o applet
1098    class MouseHandler extends MouseAdapter {
1099
1100      public void mousePressed(MouseEvent e) {
1101
1102          if (ii < nEntradas) {
1103              ii=nEntradas;
1104          }
1105
1106          if (ii > nEntradas+nVezesPassAmostra+nVezesPassAjusteFino &&
                    jj<100) {
1107             prev.addElement(new Point (e.getX(),e.getY()));
1108             for (int i=3; i<=dim; i++) {
1109               double ddd=(  Math.random()*0.99d*Math.pow( -1.0d,Math.
                    rint(Math.random()*10d) )   );
1110               prevDimAux[i-3][jj] = ddd;
1111             }
1112             jj++;
1113             repaint();
1114          }
1115       }/*chave final do método mousePressed*/
1116
1117    }/*chave final da classe MouseHandler*/
1118
1119
1120
1121    //Evento clica botão
1122    public void actionPerformed(ActionEvent evt)  {
1123
1124         Button source = (Button)evt.getSource();
1125         if (source.getLabel().equals("eApague")) {
1126            apagaPonto();
1127         }
1128         if (source.getLabel().equals("AtribuiUnidade")) {
1129            calcMaisProx();
1130            calcClassifDado();
1131         }
1132         if (source.getLabel().equals("UnidadeAoClick")) {
1133            calcPrev();
1134         }
1135         if (source.getLabel().equals("Mapeia")) {
1136            mapeiaQualidade();
1137         }
```

```
1138          if (source.getLabel().equals("Acende")) {
1139             if (enche.equals("nao")) { enche="sim"; }
1140             else { enche="nao"; }
1141          }
1142          if (source.getLabel().equals("Restarta")) {
1143             restartApplet();
1144          }
1145          if (source.getLabel().equals("MudaFundo")) {
1146             ciclicoBG++; if (ciclicoBG>=4 || ciclicoBG<=0) { ciclicoBG=
                 1; }
1147          }
1148
1149       }/*chave final do método actionPerformed*/
1150
1151
1152
1153       public void apagaPonto() {
1154
1155
1156         if (ii > nEntradas+nVezesPassAmostra+nVezesPassAjusteFino) {
1157           prev.removeElement((Point)prev.elementAt(Integer.parseInt(
                 selection.getText())-1));
1158           for (int i=3; i<=dim; i++) {
1159             for (int j= (Integer.parseInt(selection.getText())-1);
                   j<prev.size(); j++) {
1160               prevDimAux[i-3][j] = prevDimAux[i-3][j+1];
1161             }
1162           }
1163         }
1164
1165
1166         repaint();
1167
1168
1169       }/*final do método apagaPonto*/
1170
1171
1172
1173       //método do botão Restarta
1174       public void restartApplet() {
1175           autorizacao="pause";
1176           int u1=pesao.size();
1177           int u2=batida.size();
1178           int u3=prev.size();
1179           int uu=u1-1;
1180           for (int u=u1-1; u>=0; u--) {
1181             pesao.removeElement((Point)pesao.elementAt(uu));
1182             uu--;
1183           }
1184           uu=u2-1;
1185           for (int u=u2-1; u>=0; u--) {
1186             batida.removeElement((Point)batida.elementAt(uu));
1187             uu--;
1188           }
1189           uu=u3-1;
1190           for (int u=u3-1; u>=0; u--) {
1191             prev.removeElement((Point)prev.elementAt(uu));
1192             uu--;
1193           }
1194
1195           //valores default (alguns reatribuídos por redundância
1196           for (int i=0; i<=nTuplas; i++) {
1197             for (int j=0; j<=dim+2; j++) {
1198                x_d[i][j] = 0d;
1199             }
1200           }
1201
1202
1203           for (int i=0; i<=nPesos; i++) {
1204             for (int j=0; j<=dim; j++) {
1205                w_inic[i][j] = 0d;
1206                w[i][j] = 0d;
1207             }
```

```
1208        }
1209
1210
1211        for (int k=0; k <= nEntradas; k++){
1212             v[k]=0;
1213        }
1214
1215        for (int k=0; k <= nEntradas; k++){
1216             pr[k]=0;
1217        }
1218
1219        for (int i=0; i<=nPesos; i++) {
1220             contaParaNeuro[i]=id;
1221        }
1222
1223        for (int i=0; i<=nPesos; i++) {
1224             qNeuro[i]=0;
1225        }
1226
1227        for (int k=0; k <= nEntradas; k++){
1228             alarme[k]=" ";
1229        }
1230
1231        for (int i=0; i<= nMaximoDados; i++) {
1232             avalia_d[i]=0;
1233        }
1234
1235        ii=0;
1236        iteracione=0;
1237        autorizacao="nde";
1238        selecao.setText("");
1239        bandeiraAcende="nao";
1240
1241    }/*final de restartApplet*/
1242
1243
1244
1245    //atribuição dos valores vetorias
1246    public void calcSistema() {
1247
1248      double somaLinha=0d;
1249      double somaPres=0d;
1250      double somaTemp=0d;
1251      int prestemp=0;
1252      double SENSE_Q,SENSE_POS;
1253      //Random aleatorio;
1254
1255
1256      ///BLOCO DE ATRIBUIÇÃO DE VALORES PARA A ENTRADA E PARA OS
         PESOS///
1257
1258      //valores default
1259      for (int i=0; i<=nEntradas; i++) {
1260        avalia_d[i] = 0;
1261        for (int j=0; j<=dim+2; j++) {
1262         x_d[i][j] = 0d;
1263        }
1264      }
1265
1266      for (int i=0; i<=nPesos; i++) {
1267        for (int j=0; j<=dim; j++) {
1268         w_inic[i][j] = 0d;
1269         w[i][j] = 0d;
1270        }
1271      }
1272
1273      for (int k=0; k <= nEntradas; k++){
1274         v[k]=0;
1275      }
1276
1277      double menor[];
1278      menor = new double [dim+1];
1279      for (int j=0;j<=dim;j++) {
```

```
1280          menor[j]=+99999999d;
1281        }
1282
1283        double maior[];
1284        maior = new double [dim+1];
1285        for (int j=0;j<=dim;j++) {
1286          maior[j]=-99999999d;
1287        }
1288
1289
1290        //atribuição dada pelo texto de entrada (advindo de banco de
            dados)
1291        int k=0;
1292        somaPres=0d; prestemp=0; somaTemp=0d;
1293        for (int i=1; i<=nMaximoDados; i=i+1) {
1294            k++;
1295            x_d[i][0]=i;
1296            x_d[k][1]=Double.valueOf(entrada[i][2]).doubleValue()/10;
                //label do dado
1297            x_d[k][2]=Math.rint(   (Double.valueOf(entrada[i][4]).
                doubleValue())*(Math.rint(    (
                (Double.valueOf(entrada[i][5]).doubleValue()) /9400 )+
                0.00000556d  )*10000    ))/10000) + 0.055d )   );
1298            x_d[k][3]=Double.valueOf(entrada[i][5]).doubleValue();//<-
                PCS; ^ volume normalizado a PCS (depois de estar normalizado
                a P = 1 atm e T = 20°C)
1299            x_d[k][4]=Double.valueOf(entrada[i][7]).doubleValue();
                //pressão
1300            x_d[k][5]=Double.valueOf(entrada[i][8]).doubleValue();
                //temperatura
1301            x_d[k][6]=Double.valueOf(entrada[i][9]).doubleValue();
                //densidade relativa ao ar seco
1302            if (x_d[k][4]!=0d || x_d[k][5]!=0d) { somaPres+=x_d[k][4];
                somaTemp+=x_d[k][5]; prestemp++;}
1303        }
1304        if (prestemp!=0) {
1305          somaPres=somaPres/prestemp; somaTemp=somaTemp/prestemp;
1306          System.out.println("");
1307          System.out.println("pres. média = " + somaPres + "kgf/cm²" +
              "    " + "temp. média = " + somaTemp + "°C");
1308          System.out.println("");
1309          k=0;
1310          for (int i=1; i<=nMaximoDados; i=i+1) {
1311              k++;
1312              if (x_d[k][4]==0d && x_d[k][5]==0d) { x_d[k][4]=somaPres;
                  x_d[k][5]=somaTemp; }
1313          }
1314        }
1315        else {
1316          k=0;
1317          for (int i=1; i<=nMaximoDados; i=i+1) {
1318              k++;
1319              if (x_d[k][4]==0d && x_d[k][5]==0d) { x_d[k][4]=15d;
                  x_d[k][5]=15d; }
1320          }
1321        }
1322
1323        filtratiTextus();
1324        extractatMediaDP();
1325        for (int i=0; i<nPontos; i++) {
1326
1327          campoQ[i].setText(String.valueOf(Math.rint(media[i+1][2])));
1328
1329          campoP[i].setText(String.valueOf((media[i+1][3])));//3-PCS
1330          //campoP[i].setText(String.valueOf((media[i+1][4])));//4-
              pressao
1331
1332        }
1333
1334        for (int i=0; i<=nPesos; i++) {
1335          maisPerto[i]=0;
1336        }
1337
```

```
1338      //pega os parâmetros de applet "fator de sensibilidade"
1339      try {
1340         if (getParameter("amt") != null){
1341           SENSE_Q=Double.parseDouble(getParameter("amt"));
1342         }
1343         else {
1344           SENSE_Q=0.1d;
1345         }
1346         if (getParameter("tma") != null){
1347           SENSE_PCS=Double.parseDouble(getParameter("tma"));
1348         }
1349         else {
1350           SENSE_PCS=10d;
1351         }
1352      }
1353      catch (Exception e) {
1354         System.out.println(
          "Usuário inseriu valor não válido para sensibilidade: uso do
          default.");
1355         SENSE_Q=0.1d;
1356         SENSE_PCS=10d;
1357      }
1358
1359      //Normaliza para soma 1 a tupla de dados
1360      for (int i=1;i<=nMaximoDados;i++) {
1361        somaLinha=0d;
1362        x_d[i][2]=((3500000d-x_d[i][2])/3500000d)*SENSE_Q;
1363        x_d[i][3]=((10000d-x_d[i][3])/(10000d-9300d))*SENSE_PCS;
1364        x_d[i][4]=(25d-x_d[i][4])/(25d-15d);
1365        x_d[i][5]=(40d-x_d[i][5])/(40d-(-5d));
1366        x_d[i][6]=((0.700d-x_d[i][6])/(0.700d-0.600d));
1367        for (int j=2;j<=dim;j++) {
1368           somaLinha+=x_d[i][j];
1369        }
1370        for (int j=1;j<=dim;j++) {
1371           x_d[i][j]=x_d[i][j]/somaLinha;
1372        }
1373      }
1374
1375      //Normaliza dados para ficarem entre -1 e 1
1376      for (int i=1;i<=nMaximoDados;i++) {
1377        for (int j=1;j<=dim;j++) {
1378          if (x_d[i][j]>maior[j]) { maior[j]=x_d[i][j]; }
1379          if (x_d[i][j]<menor[j]) { menor[j]=x_d[i][j]; }
1380        }
1381      }
1382      for (int i=1;i<=nMaximoDados;i++) {
1383        for (int j=1;j<=dim;j++) {
1384          x_d[i][j]=( (2d*(x_d[i][j]-menor[j]))/(maior[j]-menor[j]) ) -
          1d;
1385          //x_d[i][j]=( (2d*(x_d[i][j]-0d))/(1d) ) - 1d;
1386          if (j<dim) {System.out.print(x_d[i][j]);System.out.print("
          ");} else { System.out.println(x_d[i][j]); }
1387        }
1388      }
1389
1390
1391      for (int i=1; i<=nPesos; i++) {//2
1392        for (int j=1; j<=dim; j++) {//1
1393          w[i][j]=aleatorio.nextDouble()*0.085d*Math.pow( -1.0d,
          aleatorio.nextInt(2) );
1394        }//1
1395      }//2
1396
1397      for (int i=1; i<=nPesos; i++) {//2
1398        for (int j=1; j<=dim; j++) {//1
1399          w_inic[i][j]=w[i][j];
1400        }//1
1401      }//2
1402
1403      ///FIM DO BLOCO DE ATRIBUIÇÃO DE VALORES DEFAULT PARA X E W///
1404
1405     try {
```

```
1406        leFiltrato();
1407       }
1408       catch (Exception e) {
1409         System.err.println(e);
1410         System.out.println("Não realizou o método leFiltrato()");
1411       }
1412
1413
1414      }/*fim do método de calcSistema*/
1415
1416
1417
1418       //////////REALIZAÇÃO DO ALGORITMO PROPRIAMENTE DITO/////////
1419      public void calcMesmo() {
1420
1421
1422           int guia;
1423           String repetido;
1424           int sorte;
1425           int sors[];
1426           double vencedor[];
1427           double ETA;//fator de aprendizado: temporal
1428           double R;//desvio padrão, equivale à antiga vizinhança
1429           double dist[]; //double distAnt=0.0d;
1430           double d = 0d;
1431           double H = 0d;
1432
1433
1434           //inicialização de variáveis
1435           sors = new int[nEntradas+1];
1436           dist = new double[nPesos+1];
1437           vencedor = new double[nPesos+1];
1438
1439
1440           //atribuição inicial de variáveis: por default e por segurança
1441           for (int i=0; i<=nPesos; i++) {
1442             dist[i]=0.0d;
1443             vencedor[i]=2000d;
1444           }
1445
1446
1447           //mistura a ordem dos vetores de entrada
1448           guia=1;
1449           //aleatorio = new Random();
1450           while (guia<=nEntradas) {
1451             sorte=(aleatorio.nextInt(nEntradas))+1;
1452             repetido="nao";
1453             for (int i=1; i<guia; i++) {
1454               if (sors[i]==sorte) {repetido="sim";}
1455             }
1456             if (repetido.equals("nao")) {sors[guia]=sorte; guia++;}
1457           }//fim do while
1458
1459           for (int k=1; k <= (nEntradas-(Math.rint(0.05f*nEntradas))); k
             ++){//3
1460             ETA=ETA_INI*Math.exp((-1.0d)*iteracione)/(nVezesPassAmostra
               /(ETA_INI*10)));
1461             R=R_INI*Math.exp( (-1.0d)*iteracione)/(nVezesPassAmostra/(
               Math.log(R_INI)/Math.log(2.71828d)) ) );
1462
1463             vencedor[0] = 2000.0d;
1464             //for (int i=1; i <= pesao.size(); i++){
1465             for (int i=1; i <= nPesos; i++){//2
1466
1467               dist[0]=0.0d;
1468
1469               for (int j=1; j <= dim; j++){//1
1470                 dist[0] += (x_d[sors[k]][j] - w[i][j])*(x_d[sors[k]][j]
                   - w[i][j]);
1471               }//1
1472
1473               try {
1474                 dist[0] = Math.sqrt(dist[0]);
```

```
1475              }
1476              catch (Exception e) {
1477                System.out.println("Distância negativa!!!");
1478                break;
1479              }
1480
1481              if (vencedor[0] > dist[0]) {vencedor[0] = dist[0]; v[sors[
                  k]]=i;}
1482
1483            }//2
1484
1485            //System.out.println("Oi, linha ~ 1259!");
1486
1487            for (int i=1; i<(nPesos+1); i++) {//3
1488
1489              d=0d;
1490              for (int j=1; j<3; j++) //ATENÇÃO!
                  j<2,sphericus;j<3,planus
1491
                /*planus*/d+=(neuro[i][j]-neuro[v[sors[k]]][j])
                *(neuro[i][j]-neuro[v[sors[k]]][j]);
1492              /*sphericus*//*d= Math.acos( ( Math.sin(neuro[i][j]) *
                  Math.sin(neuro[v[sors[k]]][j]) ) + (
                  Math.cos(neuro[i][j]) * Math.cos(neuro[v[sors[k]]][j]) *
                  Math.cos(neuro[v[sors[k]]][j+1]-neuro[i][j+1]) ) );
1493              if (d<0.0) d*=-1.0;
1494              if (d>Math.PI) d-=Math.PI;
1495              d*=neuro[i][0];
1496              d=Math.pow(d,2.0);*/
1497              }
1498              //d=Math.sqrt(d);
1499              H=Math.exp((-0.5d)*CR*d)/(R*R));
1500              //if (d<=R) {
1501              for (int j=1; j <= dim; j++) {
1502                w[i][j] += ETA*H*(x_d[sors[k]][j]-w[i][j]);
1503              }
1504
1505            }//3
1506
1507          }//3
1508          iteration++;
1509
1510
1511          for (int i=1; i<=nPesos; i++) {
1512            qNeuro[i]=0;
1513            contaParaNeuro[i]=1;
1514          }
1515          for (int i=0; i<=nPesos; i++) {
1516            maisPerto[i]=0;
1517          }
1518          for (int i=1; i<=nPesos; i++) {
1519
1520            for (int l=1; l<=(nEntradas-(Math.rint(0.05f*nEntradas))); l
                  ++) {//"(nEntradas-(Math.rint(0.05f*nEntradas)))" para uma
                  mostra aleatória de nEntradas
1521              if (v[l]==i) {
1522                for (int j=1; j<=dim; j++) {
1523                  dist[i] += (x_d[l][j] - w[i][j])*(x_d[l][j] -
                    w[i][j]);
1524                }
1525                try {
1526                   dist[i] = Math.sqrt(dist[i]);
1527                }
1528                catch (Exception e) {
1529                   System.out.println("Distância negativa!!!");
1530                   break;
1531                }
1532                if (vencedor[i] > dist[i]) { vencedor[i] = dist[i];
                  maisPerto[i]=1; }
1533                qNeuro[i] += 1;
1534                contaParaNeuro[i]+=2f;
1535              }
1536            }
```

```
1537
1538          }
1539
1540      }/*fim do método de calcMesmo*/
1541
1542
1543       ///////////REALIZAÇÃO DO ALGORITMO DE CONVERGÊNCIA/////////
1544      public void calcMesmo2() {
1545
1546
1547          //declaraçao de variáveis
1548          int guia;
1549          String repetido;
1550          int sorte;
1551          int sors[];
1552          double vencedor[];
1553          double ETA;//fator de aprendizado
1554          //double CR = 4.00d;//ajuste de espalhamento da gaussiana
1555          double R;//desvio padrão, equivale à antiga vizinhança
1556          double dist[]; //double distAnt=0.0d;
1557          double d = 0d;
1558          double H = 0d;
1559
1560
1561          //inicialização de variáveis
1562          sors = new int[nEntradas+1];
1563          dist = new double[nPesos+1];
1564          vencedor = new double[nPesos+1];
1565
1566          //atribuição inicial de variáveis: por default e por segurança
1567          for (int i=0; i<=nPesos; i++) {
1568            dist[i]=0.0d;
1569            vencedor[i]=2000d;
1570          }
1571
1572          //parâmetro de applet "taxa de aprendizado 'inicial'" e
             distância 'inicial'"
1573              ETA_INI=0.015d;
1574              if (ii <= 40000) { R_INI = modulo; }
1575              else { R_INI = modulo/10; }
1576
1577
1578          //mistura a ordem dos vetores de entrada
1579          guia=1;
1580          //aleatorio = new Random();
1581          while (guia<=nEntradas) {
1582            sorte=(aleatorio.nextInt(nEntradas))+1;
1583            //sorte=((int)Math.rint(Math.random()*nMaximoDados) %
               nEntradas)+1;
1584            repetido="nao";
1585            for (int i=1; i<guia; i++) {
1586              if (sors[i]==sorte) {repetido="sim";}
1587            }
1588            if (repetido.equals("nao")) {sors[guia]=sorte; guia++;}
1589          }//fim do while
1590
1591
1592          for (int k=1; k <= (nEntradas-(Math.rint(0.05f*nEntradas))); k
             ++){//3
1593            ETA=ETA_INI;
1594            R=R_INI;
1595            vencedor[0] = 2000.0d;
1596            for (int i=1; i <= nPesos; i++){//2
1597
1598              dist[0]=0.0d;
1599
1600              for (int j=1; j <= dim; j++){//1
1601                dist[0] += (x_d[sors[k]][j] - w[i][j])*(x_d[sors[k]][j]
                   - w[i][j]);
1602              }//1
1603
1604              try {
1605                dist[0] = Math.sqrt(dist[0]);
```

```
1636              }
1637              catch (Exception e) {
1638                System.out.println("Distância negativa!!!");
1639                break;
1640              }
1641
1642              if (vencedor[0] > dist[0]) {vencedor[0] = dist[0]; v[sors[
                  k]]=i;}
1643
1644            }//2
1645
1646
1647            for (int i=1; i<=nPesos+1; i++) {//3
1648
1649              d=0d;
1650              for (int j=1; j<3; j++) {//ATENÇÃO!
                  j<2,spherious;j<3,planus
1651
                  /*planus*/d+=(neuro[i][j]-neuro[v[sors[k]]][j])
                  *(neuro[i][j]-neuro[v[sors[k]]][j]);
1652              /*spherious*//*d= Math.acos( ( Math.sin(neuro[i][j]) *
                  Math.sin(neuro[v[sors[k]]][j]) ) + (
                  Math.cos(neuro[i][j]) * Math.cos(neuro[v[sors[k]]][j]) *
                  Math.cos(neuro[v[sors[k]]][j+1]-neuro[i][j+1]) ) );
1653              if (d<0.0) d*=-1.0;
1654              if (d>Math.PI) d-=Math.PI;
1655              d*=neuro[i][0];
1656              d=Math.pow(d,2.0);*/
1657              }
1658              //d=Math.sqrt(d);
1659              H=Math.exp((-0.5d)*CR*d)/(R*R));
1660              //if (d<=R) {
1661              for (int j=1; j <= dim; j++) {
1662                w[i][j] += ETA*H*(x_d[sors[k]][j]-w[i][j]);
1663              }
1664
1665            }//3
1666
1667          }//3
1668          iteracione++;
1669
1670
1671          for (int i=1; i<=nPesos; i++) {
1672            qNeuro[i]=0;
1673            contaParaNeuro[i]=1;
1674          }
1675          for (int i=0; i<=nPesos; i++) {
1676            maisPerto[i]=0;
1677          }
1678          for (int i=1; i<=nPesos; i++) {
1679
1680            for (int l=1; l<=(nEntradas-(Math.rint(0.05f*nEntradas))); l
                ++) {//"(nEntradas-(Math.rint(0.05f*nEntradas)))" p/ uma
                mostra aleatória de nEntradas
1681              if (v[l]==i) {
1682                for (int j=1; j<=dim; j++) {
1683                  dist[i] += (x_d[l][j] - w[i][j])*(x_d[l][j] -
                    w[i][j]);
1684                }
1685                try {
1686                   dist[i] = Math.sqrt(dist[i]);
1687                }
1688                catch (Exception e) {
1689                   System.out.println("Distância negativa!!!");
1690                   break;
1691                }
1692                if (vencedor[i] > dist[i]) { vencedor[i] = dist[i];
                  maisPerto[i]=1; }
1693                qNeuro[i] += 1;
1694                contaParaNeuro[i]+=2f;
1695              }
1696            }
1697
```

```
1668          }
1669
1670      }/*fim do método de calcMesmo2*/
1671
1672
1673      //calcula qual a unidade neural mais próxima da linha de dado
          escolhida na interface gráphica
1674      public void calcMaisProx() {
1675       int linhaDado = 1;
1676       double distance = 0d;
1677       if (ii >= nEntradas+nVezesPassAmostra+2) {
1678        pesoVictor = 0;
1679        double victor = 9999999d;
1680        try {
1681         linhaDado = Integer.parseInt(selecao.getText());
1682         if (linhaDado>nMaximoDados) { linhaDado=nTuplas*2; selecao.
          setText(String.valueOf(nMaximoDados)); }
1683         if (linhaDado<=0) { linhaDado = 1; }
1684        }
1685        catch (Exception e) {
1686         System.out.println("Valor invalido de seleção, uso do
          default.");
1687         linhaDado = 1;
1688         selecao.setText("1");
1689        }
1690        int l = linhaDado;
1691        for (int i=1; i<=nPesos; i++) {
1692            for (int j=1; j<=dim; j++) {
1693              distance += (x_d[l][j] - w[i][j])*(x_d[l][j] - w[i][j]);
1694            }
1695            try {
1696              distance = Math.sqrt(distance);
1697            }
1698            catch (Exception e) {
1699               System.out.println("Distância negativa!!!");
1700               break;
1701            }
1702            if (victor > distance) { victor = distance; pesoVictor=i;}
1703            distance = 0d;
1704        }
1705        System.out.println("A unidade vitoriosa para a linha de dado " +
          linhaDado + " é a " + pesoVictor + ".");
1706        if (bandeiraAcende.equals("nao")) { bandeiraAcende="sim"; }
1707        else { bandeiraAcende="nao"; }
1708       }
1709      }//fim do método calcMaisProx
1710
1711
1712
1713      //classifica as unidades pela qualidade de dados atribuídos
1714      public void mapeiaQualidade() {
1715
1716
1717        double QDC[],PCS[];//P[];
1718        QDC=new double[nPontos]; PCS=new double[nPontos];//P=new
          double[nPontos];
1719
1720        //pega o parâmetro de applet "Quantidade Diária Contratada (m3)"
1721        try {
1722             for (int i=0;i<nPontos;i++) {
1723               if (campoQ[i].getText() != null){
1724                 QDC[i]=Double.parseDouble(campoQ[i].getText());
1725               }
1726               else {
1727                 QDC[i]=2000000d;
1728               }
1729             }
1730        }
1731        catch (Exception e) {
1732             System.out.println(
             "Usuário colocou valor de QDC não valido, então uso do
             default.");
1733             for (int i=0;i<nPontos;i++) {
```

```
1734              QDC[i]=2000000d;
1735            }
1736        }
1737
1738        //pega o parâmetro de applet "Poder Calorífico Superior
            (kcal/m3)"
1739        try {
1740             for (int i=0;i<nPontos;i++) {
1741               if (campoP[i].getText() != null){
1742                 PCS[i]=Double.parseDouble(campoP[i].getText());
1743                 System.out.println("PCS"+ (i+1) + " = " + PCS[i]);
1744               }
1745               else {
1746                 PCS[i]=9600d;
1747               }
1748             }
1749        }
1750        catch (Exception e) {
1751             System.out.println(
             "Usuário colocou valor de PCS não valido, então uso do
             default.");
1752             for (int i=0;i<nPontos;i++) {
1753               PCS[i]=9600d;
1754             }
1755        }
1756
1757
1758        //atribuição de classe ou qualidade às medidas ou vectores de
            dados ou de entrada
1759        avalia_d[0]=0;
1760        for (int i=1; i<=nMaximoDados; i=i+nPontos) {
1761          for (int j=0; j<nPontos; j++) {
1762            if ( Double.parseDouble(qualy[j+i][4])<0.995d*PCS[j] ) {
1763              avalia_d[j+i]=4;
1764            }
1765            else {
1766              if ( Double.parseDouble(qualy[j+i][3])<0.8d*QDC[j] ) {
1767                avalia_d[j+i]=3;
1768              }
1769              else {
1770                if ( Double.parseDouble(qualy[j+i][3])>=1.05d*QDC[j] ) {
1771                  avalia_d[j+i]=1;
1772                }
1773                else {
1774                  avalia_d[j+i]=2;
1775                }
1776              }
1777            }
1778          }//fim do for intermediário para atribuição de classe para cada
             vector de entrada
1779        }//fim do for para atribuição de classe para cada vector de
            entrada
1780
1781        try {
1782           String pring = new String("Sincrano");
1783           FileOutputStream fos = new FileOutputStream(
             "C:/Documents and Settings/osiz/Meus
             documentos/DHTML_Java_Projects/AtWork/K73_367Hex31b16dim/volPcs
             /classificatorium.txt");
1784           PrintStream outputFile = new PrintStream(fos);
1785           for (int i = 1; i<=nMaximoDados; i++) {
1786               pring = "";
1787               for (int j = 0; j <= dim+1; j++) {
1788                 if (j==0){pring = pring + String.valueOf(i); pring =
                   pring + ";"; pring = pring + avalia_d[i]; pring = pring
                   + ";";}
1789                 if (j>0 && j<dim+1) { pring = pring + String.valueOf(x_d
                   [i][j]); pring = pring + ";"; }
1790                 if (j==dim+1) {pring = pring + avalia_d[i];//repete
                   coluna avalia_d[i] para melhor visualização
1791               }
1792               outputFile.println(pring);
1793           }
```

```
1794            outputFile.close();
1795            fos.close();
1796         }
1797         catch (IOException io) {
1798            System.err.println("Erro ao escrever o arquivo");
1799            io.printStackTrace();
1800         }
1801
1802
1803      }//fim do método mapeiaQualidade
1804
1805
1806
1807      //calcula a que classe pertence o novo dado
1808      public void calcPrev() {
1809
1810         double vencedorPrev;
1811         double dist=0.0d;
1812         double x_V[][];//os dados para validação e previsão
1813         int venc[];//i=0; int venc2=0; int venc3=0; int venc4=0; int
             venc5=0; int venc6=0;
1814         String[] aviso = {" ","10% maior","00% resto","zero% !"};
1815         int decimoPesos = (int)Math.round((nPesos*1d)/10d);
1816         int aux;
1817
1818
1819         x_V = new double[nEntradas+1][dim+1];
1820         venc = new int[8];
1821
1822
1823         //valores default
1824         for (int k=0; k <= nEntradas; k++){
1825                alarme[k]=aviso[0];
1826         ///
1827
1828
1829         for (int k=0; k <= nEntradas; k++){
1830                pr[k]=0;
1831         ////
1832
1833         for (int i=0; i<=nEntradas; i++) {
1834            for (int j=0; j<=dim; j++) {
1835                x_V[i][j] = 0d;
1836            }
1837         /////
1838
1839         for (int i=0; i<=7; i++) {
1840            venc[i]=0;
1841         //////
1842
1843         //atribuição dos valores
1844         for (int i=1; i<=prev.size(); i++) {
1845
1846           for (int j=1; j<=dim; j++) {
1847             if (j==1) {
1848                x_V[i][j]=( ( ((Point)prev.elementAt(i-1)).x ) / 350d ) -
                   1d;
1849             }
1850             else { if (j==2) {
1851                         x_V[i][j]=( ( ((Point)prev.elementAt(i-1) ).y)-
                            1750d ) / 400d ) - 1d;
1852                   }
1853                   else { if (j==3) {
1854                            while (j<=dim) {
1855                              x_V[i][j]=prevDimAux[j-3][i-1];
1856                              j++;
1857                            }
1858                          }
1859                   }
1860             }
1861           }
1862
1863         ///fim do for para atribuição de valores
```

```
1864
1865      for (int k=1; k <= prev.size(); k++){//3
1866         vencedorPrev = 2000.0d;
1867         for (int i=1; i <= nPesos; i++){//2
1868
1869          dist=0.0d;
1870          for (int j=1; j <= dim; j++){//1
1871            dist += (x_V[k][j] - w[i][j])*(x_V[k][j] - w[i][j]);
1872          }//1
1873          try {
1874               dist = Math.sqrt(dist);
1875          }
1876          catch (Exception e) {
1877               System.out.println("Distancia negativa!!!");
1878               break;
1879          }
1880          if (vencedorPrev > dist) {vencedorPrev = dist; pr[k]=i;}
1881
1882         }//2
1883       }//3
1884
1885      if (decimoPesos>=1) {
1886       aux=qNeuro[0];
1887       for(int m=nPesos; m>=1; m--) {
1888         if(qNeuro[m]>aux) {
1889           venc[1]=m; aux=qNeuro[m];
1890         }
1891       }//
1892      }
1893
1894      if (decimoPesos>=2) {
1895       aux=qNeuro[0];
1896       for(int m=nPesos; m>=1; m--) {
1897         if(qNeuro[m]>aux && m!=venc[1]) {
1898           venc[2]=m; aux=qNeuro[m];
1899         }
1900       }///
1901      }
1902
1903      if (decimoPesos>=3) {
1904       aux=qNeuro[0];
1905       for(int m=nPesos; m>=1; m--) {
1906         if(qNeuro[m]>aux && m!=venc[1] && m!=venc[2]) {
1907           venc[3]=m; aux=qNeuro[m];
1908         }
1909       }////
1910      }
1911
1912      if (decimoPesos>=4) {
1913       aux=qNeuro[0];
1914       for(int m=nPesos; m>=1; m--) {
1915         if(qNeuro[m]>aux && m!=venc[1] && m!=venc[2] && m!=venc[3]) {
1916           venc[4]=m; aux=qNeuro[m];
1917         }
1918       }/////
1919      }
1920
1921      if (decimoPesos>=5) {
1922       aux=qNeuro[0];
1923       for(int m=nPesos; m>=1; m--) {
1924         if(qNeuro[m]>aux && m!=venc[1] && m!=venc[2] && m!=venc[3] &&
           m!=venc[4]) {
1925           venc[5]=m; aux=qNeuro[m];
1926         }
1927       }//////
1928      }
1929
1930      if (decimoPesos>=6) {
1931       aux=qNeuro[0];
1932       for(int m=nPesos; m>=1; m--) {
1933         if(qNeuro[m]>aux && m!=venc[1] && m!=venc[2] && m!=venc[3] &&
           m!=venc[4] && m!=venc[5]) {
1934           venc[6]=m; aux=qNeuro[m];
```

```
1935          }
1936         }///////
1937        }
1938
1939        if (decimoPesos>=7) {
1940         aux=qNeuro[0];
1941         for(int m=nPesos; m>=1; m--) {
1942           if(qNeuro[m]>aux && m!=venc[1] && m!=venc[2] && m!=venc[3] &&
             m!=venc[4] && m!=venc[5] && m!=venc[6]) {
1943             venc[7]=m; aux=qNeuro[m];
1944           }
1945         }///////
1946        }
1947
1948        for (int k=1; k <= prev.size(); k++){
1949           if (pr[k]==venc[1]||pr[k]==venc[2]||pr[k]==venc[3]||pr[k]==venc
             [4]||pr[k]==venc[5]||pr[k]==venc[6]||pr[k]==venc[7]) {
1950              alarme[k]=aviso[1];
1951              System.out.println("Entrada "+ k +" mais próxima à unidade "
                  + pr[k] +", que esta entre os "+ alarme[k] +
                  " de dados recebidos.");
1952           }
1953           else {
1954              if (qNeuro[pr[k]]==0) {
1955                 alarme[k]=aviso[3];
1956                 System.out.println("Entrada "+ k +
                     " mais próxima à unidade "+ pr[k] +", que esta entre os "
                     + alarme[k] +" de dados recebidos.");
1957              }
1958              else {
1959                 alarme[k]=aviso[2];
1960                 System.out.println("Entrada "+ k +
                     " mais próxima à unidade "+ pr[k] +", que esta entre os "
                     + alarme[k] +" de dados recebidos.");
1961              }
1962           }
1963        }
1964
1965
1966     }/*fim do método calcPrev*/
1967
1968
1969
1970     //calcula a que classe pertence o novo dado
1971     public void calcClassifDado() {
1972
1973
1974       double vencedorPrev;
1975       double dist=0.0d;
1976       double x_C[];//os dados para comparação e classificação
1977       int venc[];//1=0; int venc2=0; int venc3=0; int venc4=0; int
          venc5=0; int venc6=0;
1978       String[] aviso = {" ","10% maior","90% resto","zero! !"};
1979       int decimoPesos = (int)Math.round((nPesos*1d)/10d);
1980       int aux;
1981       int proximado;
1982       int lineaData;
1983       String alarme;
1984
1985
1986       x_C = new double[dim+1];
1987       venc = new int[8];
1988
1989
1990       alarme=aviso[0];
1991
1992
1993       proximado=0;
1994
1995
1996          for (int j=0; j<=dim; j++) {
1997              x_C[j] = 0d;
1998          }
```

```
1999
2000
2001      for (int i=0; i<=7; i++) {
2002        venc[i]=0;
2003      }
2004
2005      lineaData = Integer.parseInt(selecao.getText());
2006
2007      //atribuição dos valores
2008        for (int j=1; j<=dim; j++) {
2009             x_C[j]=x_d[lineaData][j];
2010        }
2011
2012
2013         vencedorPrev = 2000.0d;
2014         for (int i=1; i <= nPesos; i++){//2
2015
2016          dist=0.0d;
2017          for (int j=1; j <= dim; j++){//1
2018            dist += (x_C[j] - w[i][j])*(x_C[j] - w[i][j]);
2019          }//1
2020          try {
2021               dist = Math.sqrt(dist);
2022          }
2023          catch (Exception e) {
2024               System.out.println("Distância negativa!!!");
2025               break;
2026          }
2027          if (vencedorPrev > dist) {vencedorPrev = dist; proximado=i;}
2028
2029         }//2
2030
2031      if (decimoPesos>=1) {
2032       aux=qNeuro[0];
2033       for(int m=nPesos; m>=1; m--) {
2034         if(qNeuro[m]>aux) {
2035           venc[1]=m; aux=qNeuro[m];
2036         }
2037       }//
2038      }
2039
2040      if (decimoPesos>=2) {
2041       aux=qNeuro[0];
2042       for(int m=nPesos; m>=1; m--) {
2043         if(qNeuro[m]>aux && m!=venc[1]) {
2044           venc[2]=m; aux=qNeuro[m];
2045         }
2046       }///
2047      }
2048
2049      if (decimoPesos>=3) {
2050       aux=qNeuro[0];
2051       for(int m=nPesos; m>=1; m--) {
2052         if(qNeuro[m]>aux && m!=venc[1] && m!=venc[2]) {
2053           venc[3]=m; aux=qNeuro[m];
2054         }
2055       }////
2056      }
2057
2058      if (decimoPesos>=4) {
2059       aux=qNeuro[0];
2060       for(int m=nPesos; m>=1; m--) {
2061         if(qNeuro[m]>aux && m!=venc[1] && m!=venc[2] && m!=venc[3]) {
2062           venc[4]=m; aux=qNeuro[m];
2063         }
2064       }/////
2065      }
2066
2067      if (decimoPesos>=5) {
2068       aux=qNeuro[0];
2069       for(int m=nPesos; m>=1; m--) {
2070         if(qNeuro[m]>aux && m!=venc[1] && m!=venc[2] && m!=venc[3] &&
          m!=venc[4]) {
```

```
2071            venc[5]=m; aux=qNeuro[m];
2072          }
2073         }//////
2074        }
2075
2076        if (decimoPesos>=6) {
2077         aux=qNeuro[0];
2078         for(int m=nPesos; m>=1; m--) {
2079           if(qNeuro[m]>aux && m!=venc[1] && m!=venc[2] && m!=venc[3] &&
             m!=venc[4] && m!=venc[5]) {
2080             venc[6]=m; aux=qNeuro[m];
2081           }
2082         ///////
2083        }
2084
2085        if (decimoPesos>=7) {
2086         aux=qNeuro[0];
2087         for(int m=nPesos; m>=1; m--) {
2088           if(qNeuro[m]>aux && m!=venc[1] && m!=venc[2] && m!=venc[3] &&
             m!=venc[4] && m!=venc[5] && m!=venc[6]) {
2089             venc[7]=m; aux=qNeuro[m];
2090           }
2091         }///////
2092        }
2093
2094
2095           if (proximado==venc[1]||proximado==venc[2]||proximado==venc[3]||
             proximado==venc[4]||proximado==venc[5]||proximado==venc[6]||
             proximado==venc[7]) {
2096              alarme=aviso[1];
2097              System.out.println("Entrada "+ lineaData +
                 " mais próxima à unidade "+ proximado +
                 ", que esta entre os "+ alarme +" de dados recebidos.");
2098              System.out.println("");
2099           }
2100           else {
2101              if (qNeuro[proximado]==0) {
2102                 alarme=aviso[3];
2103                 System.out.println("Entrada "+ lineaData +
                    " mais próxima à unidade "+ proximado +
                    ", que esta entre os "+ alarme +" de dados recebidos.");
2104                 System.out.println("");
2105              }
2106              else {
2107                 alarme=aviso[2];
2108                 System.out.println("Entrada "+ lineaData +
                    " mais próxima à unidade "+ proximado +
                    ", que esta entre os "+ alarme +" de dados recebidos.");
2109                 System.out.println("");
2110              }
2111           }
2112
2113
2114       }/*fim do método calcClassifDado*/
2115
2116
2117
2118       public void leInformacao()  throws Exception{
2119
2120          int nGrupos=0;
2121          try {//a
2122              BufferedReader in = new BufferedReader(new FileReader(
                 "C:/Documents and Settings/rslz/Meus
                 documentos/DHTML_Java_Projects/AtWork/K73_3G7Hex3lbl6dimVolP
                 os/datones.txt"));
2123              String ss;
2124              int lin=1;
2125              int col=1;
2126                 while ((ss = in.readLine()) != null) {
2127                   StringTokenizer st = new StringTokenizer(ss,";");
2128                   if(!st.hasMoreTokens()) { throw new Exception(
                      "Erro de sintaxe no arquivo");}
2129                   if (col==1) {
```

```
2130                while (col<=10) {
2131                  entrada[lin][col]=st.nextToken();
2132                  col++;
2133                }
2134                if (col==11) { col=1; lin++;}
2135              }
2136            }
2137          nMaximoDados=(lin-1);
2138          //trecho deste método que descobre quantas estações de
              medição ha no arquivo texto datones
2139          for (int i=2; i<=10; i++) {
2140            if (entrada[i][2].equals(entrada[i][2])) {
2141              nPontos=i-1; i=11;
2142            }
2143          }
2144          //fim do trecho de descoberta do número de pontos (ou
              estações) de medição
2145          nTuplas=nMaximoDados/nPontos;
2146      }//final try a
2147      catch (IOException e) {//a
2148          for (int i=0; i<=nMaximoDados; i++) {
2149               for (int j=0; j<=10; j++) {
2150                  entrada[i][j] = "Não conseguiu ler TXT datones";
2151               }
2152          }
2153       }//final catch a
2154
2155      for (int i=1; i<=nMaximoDados; i++) {
2156           nGrupos++;
2157      }
2158      if (nEntradas>nGrupos) { nEntradas=nGrupos; }
2159
2160      for (int i=1; i<=nMaximoDados; i++) {
2161          for (int j=1; j<=10; j++) {
2162              System.out.print(entrada[i][j]);
2163              System.out.print(";");
2164              if (j==10) {System.out.println("");}
2165          }
2166      }
2167
2168  }//final do método leInformação
2169
2170
2171  private void filtratiTextus( ) {
2172    try {
2173       String pring = new String("Fulano");
2174       FileOutputStream fos = new FileOutputStream(
          "D:/Documents and Settings/csiz/Meus
          documentos/DHDML_Java_Projects/AtWork/X73_367Hex3lbl6dimVolPos
          /filtrati.txt");
2175       PrintStream outputFile = new PrintStream(fos);
2176       for (int i = 1; i<=nMaximoDados; i++) {
2177           pring = "";
2178           for (int j = 0; j <= dim+1; j++) {
2179             if (j==0){pring = pring + String.valueOf(i); pring =
                  pring + ";";}
2180             if (j>0 && j<dim+1) { pring = pring + String.valueOf(x_d
                  [i][j]); pring = pring + ";"; }
2181             if (j==dim+1) {pring = pring +
                  String.valueOf(x_d[i][0]);}
2182           }
2183           outputFile.println(pring);
2184       }
2185       outputFile.close();
2186       fos.close();
2187    }
2188    catch (IOException io) {
2189       System.err.println("Erro ao escrever o arquivo");
2190       io.printStackTrace();
2191    }
2192  }//final do método filtratiTextus
2193
2194
```

```
2195
2196    public void leFiltrato() throws Exception{
2197
2198       int nGrupos=0;
2199       try {//a
2200           BufferedReader in = new BufferedReader(new FileReader(
           "D:/Documents and Settings/csiz/Meus
           documentos/DHTML_Java_Projects/AtWork/K73_367Hex3lbl6dimVolP
           os/filtrati.txt"));
2201           String ss;
2202           int lin=1;
2203           int col=1;
2204              while ((ss = in.readLine()) != null) {
2205                StringTokenizer st = new StringTokenizer(ss,";");
2206                if(!st.hasMoreTokens()) { throw new Exception(
                "Erro de sintaxe no arquivo");}
2207                if (col==1) {
2208                  while (col<=8) {
2209                    qualy[lin][col]=st.nextToken();
2210                    col++;
2211                  }
2212                  if (col==9) { col=1; lin++;}
2213                }
2214              }
2215           nMaximoDados=(lin-1);
2216           etiqueta4.setText(String.valueOf(nMaximoDados));
2217           nTuplas=nMaximoDados/nPontos;
2218       }//final try a
2219       catch (IOException e) {//a
2220           for (int i=0; i<=nMaximoDados; i++) {
2221                for (int j=0; j<=8; j++) {
2222                   qualy[i][j] = "Não conseguiu ler TXT";
2223                }
2224           }
2225       }//final catch a
2226
2227       for (int i=1; i<=nMaximoDados; i++) {
2228         if (Float.parseFloat(entrada[i][2]) == 0.1f) {
2229           nGrupos++;
2230         }
2231       }
2232
2233    }//final do método leFiltrato
2234
2235
2236
2237    public void extractatMediaDP() {
2238
2239        double multiplicaCasa;
2240        int indicaCasa=0;
2241
2242        for (int j=0; j<=dim; j++) {
2243            for (int k=0; k<=nPontos; k++) {
2244              media[k][j]=0d;
2245              desvioPadraoSignif[k][j]=0d;
2246              desvioPadraoNat[k][j]=0d;
2247              dPAlgarismoSignificativo[k][j]=0;
2248            }
2249        }
2250        for (int j=2; j<=dim; j++) {
2251          for (int k=0; k<=(nPontos-1); k++) {
2252            for (int i=1; i<=nEntradas; i=i+nPontos) {
2253              media[k+1][j]+=x_d[i+k][j];
2254            }
2255          }
2256        }
2257        for (int j=2; j<=dim; j++) {
2258          for (int k=0; k<=(nPontos-1); k++) {
2259            media[k+1][j]=media[k+1][j]/(nEntradas/nPontos);
2260          }
2261        }
2262        for (int j=2; j<=dim; j++) {
2263          for (int k=0; k<=(nPontos-1); k++) {
```

```
2264            for (int i=1; i<=nEntradas; i=i+nPontos) {
2265              desvioPadraoSignif[k+1][j]+=(x_d[i+k][j]-media[k+1][j])*(
                  x_d[i+k][j]-media[k+1][j]);
2266            }
2267            if (desvioPadraoSignif[k+1][j]<=0.000001d) {
                desvioPadraoSignif[k+1][j]=0d;}
2268          }
2269        }
2270        for (int j=2; j<=dim; j++) {
2271          for (int k=0; k<=(nPontos-1); k++) {
2272
                desvioPadraoSignif[k+1][j]=
                Math.sqrt(desvioPadraoSignif[k+1][j]/(nEntradas/nPontos));
2273            desvioPadraoNat[k+1][j]=desvioPadraoSignif[k+1][j];
2274          }
2275        }
2276        for (int j=2; j<=dim; j++) {
2277          for (int k=0; k<=(nPontos-1); k++) {
2278            multiplicaCasa=1d;
2279            indicaCasa=0;
2280            while (indicaCasa==0){
2281              if (desvioPadraoSignif[k+1][j]<1d && desvioPadraoSignif[k
                  +1][j]>0d) {
2282                multiplicaCasa*=10d;
2283                indicaCasa=(int)Math.rint(desvioPadraoSignif[k+1][j]*
                    multiplicaCasa);
2284              }
2285              else {
2286                indicaCasa=1;
2287                multiplicaCasa=1d;
2288              }
2289
2290            dPAlgarismoSignificativo[k+1][j]=(int)Math.rint((
                desvioPadraoSignif[k+1][j]+(1d/((int)multiplicaCasa*10)))*(
                int)multiplicaCasa);
2291            desvioPadraoSignif[k+1][j]=((int)Math.rint(
                desvioPadraoSignif[k+1][j]*multiplicaCasa))*(1d/(int)
                multiplicaCasa);
2292            media[k+1][j]=((int)Math.rint(media[k+1][j]*(int)
                multiplicaCasa))*(1d/(int)multiplicaCasa);
2293          }
2294        }
2295
2296    }//final do método extractatMediaDP
2297
2298
2299
2300    public void fixaBackground( ) {
2301        switch (cicliccBG) {
2302          case 1 : setBackground(new Color(250,247,153)); break;
2303          case 2 : setBackground(Color.white); break;
2304          case 3 : setBackground(Color.black); break;
2305          default : setBackground(new Color(250,247,153)); break;
2306        }
2307    }//final do método fixaBackground
2308
2309
2310
2311
2312  }/*final da classe Kohonen73*/
```

REFERÊNCIAS BIBLIOGRÁFICAS

1 AMERICAN GAS ASSOCIATION. **Compressibility factor of natural gas and related hydrocarbon gases**. Report n. 8, 1994.

2 BAJS, T.; ORIANI, L.; RICOTTI, M. E.; BARROSO, A. C. Transient analysis of the IRIS reactor. In: INTERNATIONAL CONFERENCE NUCLEAR ENERGY FOR NEW EUROPE, Sept. 9-12, 2002, Kranjska Gora, Slovenia.

3 BALLONE, G. J. **Cérebro e violência**. http://www.psiqweb.med.br/forense/cerebro.html, 2002.

4 BAPTISTA FILHO, B. D. **Redes neurais artificiais na engenharia nuclear** – parte 2. Revista Brasileira de Pesquisa e Desenvolvimento, vol. 4, no.2, págs.220-225, 2002.

5 BAPTISTA FILHO, B. D.; BARROSO, A. C. O. Identification of IRIS reactor transients with Self-Organized Maps. In: INTERNATIONAL CONFERENCE ON GLOBAL ENVIRONMENT AND ADVANCED NUCLEAR POWER PLANTS (GENES4/ANP2003), 2003, Kyoto.

6 BAPTISTA FILHO, B. D.; BARROSO, A. C. O. Self-Organized Maps in the identification of IRIS reactor transients. In: THE INTERNATIONAL CONFERENCE ON INTEGRATED MODELING & ANALYSIS IN APPLIED CONTROL & AUTOMATION (IMAACA 2004), Oct. 28-30, 2004, Genoa. (Paper IMAACA_Nu-05).

7 BARROSO, A. C. O.; BAPTISTA FILHO, B. D.; ARONE, I. D.; MACEDO, L. A.; SAMPAIO, P. A. B.; MORAES, M. IRIS pressurizer design. In: INTERNATIONAL CONGRESS ON ADVANCES IN NUCLEAR POWER PLANTS (ICAPP'03), May 4-7, 2003, Córdoba. (Paper 3227).

8 BRAILOWSKY, S. **Las sustâncias de los sueños: Neuropsicofarmacología** – primera parte: el sistema nervioso central (SNC). http://omega.ilce.edu.mx:3000/sites/ciencia/volumen3/ciencia3/130/html/sec_5.html, 1995.

9 CARDOSO, S. H.; MELLO, L. C.; SABBATINI, R. M. E. **Como funcionam as células nervosas** - parte II: o potencial de ação. http://www.epub.org.br/cm/n10/fundamentos/pot2.htm, 2000.

10 CARELLI, M. D. *et al.* IRIS, International New Generation Reactor. **Proceedings of the 8th International Conference on Nuclear Engineering** (ICONE8). Baltimore: ICONE, 2000. 1 CD-ROM.

11 CARELLI, M. D.; CONWAY, L. E.; FIORINI, G. L.; LOMBARDI, C. V.; RICOTTI, M. E.; ORIANI, L.; BERRA, F.; TODREAS, N. E. Safety by Design: A New Approach to Accident Management in the IRIS Reactor. In: IAEA INTERNATIONAL SEMINAR ON STATUS AND PROSPECTS FOR SMALL AND MEDIUM SIZED REACTORS, May 27-31, 2001, Cairo. (IAEA-SR-218-36).

12 DEITEL, H.M.; DEITEL, P.J. **Java, como programar**. Porto Alegre, RS: Bookman, 2001.

13 ELETRONUCLEAR. **Energia nuclear no Brasil**. http://www.eletronuclear.gov.br/novo/sys/interna.asp?IdSecao=50&secao_mae=2, 2003.

14 GÁS&ENERGIA. **Reservas nacionais**. http://www.gasenergia.com.br/portalge/port/gn/reservas_nac.jsp, 2005.

15 GAS NET. **Gás natural**. http://www.gasnet.com.br/gasnet_br/oque_gn/gas_completo.html, 1999.

16 GERSHO, A.; GRAY, R.M. **Vector quantization and signal compression**. Norwell, MA: Kluwer, 1992.

17 HAYKIN, S., **Neural networks: a comprehensive foundation**. Jersey City, N. J.: Prentice Hall, 1999.

18 JOHNSON, G. **Nos palácios da memória**. Tradução de C. A. Gimenez. São Paulo, SP: Siciliano, 1994.

19 JOSEPH, R. **Functional neuroanatomy**. http://brainmind.com, 2005.

20 KOHONEN, T. Self-organized formation of topologically correct feature maps. **Biological Cybernetics**, v. 43, pp. 59-69, 1982. (Reprinted by Anderson and Rosenfeld, 1988).

21 KOHONEN, T. The self-organizing map. **Proceedings of the Institute of Electrical and Electronics Engineers**, v.78, pp. 1464-1480, 1990.

22 KOVÁCS, Z. L. **Redes neurais artificiais**: fundamentos e aplicações. S. Paulo, SP: Collegium Cognitio, 1998.

23 KRÖSE, B; VAN DER SMAGT, P. **An introduction to neural networks**. http://www.avaye.com/files/articles/nnintro/nn_intro.pdf, 1996.

24 LO, Z. P.; FUJITA, M.; BAVARIAN, B. Analysis of neighborhood interaction in Kohonen neural networks. **6th International Parallel Processing Symposium** (IPPS'91) **Proceedings,** pp. 246-249. Los Alamitos, CA: IEEE, 1991.

25 LO, Z. P.; YU, Y.; BAVARIAN, B. Analysis of the convergence properties of topology preserving neural networks. **IEEE Transactions on Neural Networks**, v. 4, pp. 207-220, 1993.

26 LUTTRELL, S. P. Hierarchical vector quantization. **IEE Proceedings** (London), v. 136, pp. 405-413, 1989. Part I.

27 MORRISON, M. **Java 2**. Rio de Janeiro, RJ: Ciência Moderna, 2000.

28 NADEL, L.; COOPER, A.; CULICOVER, P.; HARNISH, M. (editors). **Neural connections, mental computation**. Cambridge, MA: MIT Press, 1989.

29 NASCIMENTO JÚNIOR, C. L.; YONEYAMA, T. **Inteligência artificial em controle e automação**. São Paulo, SP: Edgard Blucher - FAPESP, 2000.

30 NEURODIMENSION. **Resources**. http://www.nd.com/resouces, 2005.

31 OPENSHAW, S.; TURTON, I. A parallel Kohonen algorithm for the classification of large spatial datasets. **Computer & Geosciences**, v. 22, n. 9, pp. 1019-1026, 1996.

32 PACKER, C. A framework for the organizational assumptions underlying safety culture. **Annals of the International Safety Conference on Safety Culture in Nuclear Installations**. Rio de Janeiro: IAEA, 2002.

33 PAPLIŃSKI, A. P. **Neural networks**, L. 10. http://www.csse.monash.edu.au/courseware/cse5301/04/Lnts/L10.pdf, 2002.

34 PETROBRAS. **Cálculo de poder calorífico, fator de compressibilidade, índice de Wobbe e densidade relativa de combustíveis gasosos**. NORMALIZAÇÃO TÉCNICA PETROBRAS N-2535, 2001.

35 PETROVIC, B. International collaboration and neutronic analyses in support of the IRIS Project. **Transactions of American Nuclear Society**, v. 83, pp. 186-187, 2000.

36 RITTER, H.; MATINETZ, T.; SCHULTEN, K. **Neural computation and self-organizing maps**: an introduction. Reading, MA: Addison-Wesley, 1992.

37 ROSENBLATT, F. The Perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, pp. 386-408, 1958.

38 SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de banco de dados**. São Paulo, SP: Makron Books, 1999.

39 STEINER, J. E. (coordenador). **Introdução à astronomia e astrofísica**. Cap. I por R. Boczko. São Paulo, SP: IAGUSP, 1978.

40 WILLSHAW, D. J.; VON DER MARLSBURG, C. How patterned neural connections can be set up by self-organization. **Proceedings of the Royal Society of London**, Series B, v. 194, pp. 431-445, 1976.

Instituto de Pesquisas Energéticas e Nucleares
Diretoria de Ensino & Informação Científica e Tecnológica
Av. Prof. Lineu Prestes, 2242 Cidade Universitária CEP: 05508-000
Fone/Fax(0XX11) 3816 - 9148
SÃO PAULO - São Paulo - Brasil
http: //www.ipen.br

O Ipen é uma autarquia vinculada à Secretaria de Ciência, Tecnologia e Desenvolvimento Econômico e Turismo do Estado de São Paulo, gerida técnica, administrativa e financeiramente pela Comissão Nacional de Energia Nuclear, orgão do Ministério da Ciência e Tecnologia, e associada à Universidade de São Paulo.