Uso de Técnicas Neurais para o Reconhecimento de Comandos a Voz

Suelaine dos Santos Diniz *
*Antonio Carlos Gay Thomé***

No artigo anterior (parte I) abordou-se o problema do Reconhecimento Automático da Voz —RAV— e apresentou-se a fundamentação teórica sobre o processamento dos sinais de voz. Neste artigo será discutida a possibilidade da utilização de técnicas neurais na construção de modelos para o Reconhecimento de Palavra Isolada —RPI. No próximo artigo (parte III) serão apresentados os modelos neurais concebidos para o estudo de caso realizado e os resultados alcançados.

PARTE II

MODELOS NEURAIS E O RECONHECIMENTO AUTOMÁTICO DA VOZ

INTRODUÇÃO

O RAV é um problema complexo e de difícil solução face a diversas características próprias tais como: a não estacionariedade do sinal de voz, particularidades do idioma falado, qualidade dos equipamentos de captura utilizados e níveis de ruído do ambiente.

Face a tais dificuldades no tratamento do problema, observa-se hoje uma forte tendência na busca de modelos não convencionais, geralmente empre-gando técnicas de redes neurais[1,2,3,4,5,6,7], cadeias de Markov[8,9] e uso de wavelets. O emprego de sistemas híbridos é uma outra variante de estudo.

Dentre as técnicas acima mencionadas, aquela baseada no emprego de redes neurais tem recebido especial ateção

* MC / IME
** Cel. R/1 – Ph.D. / Purdue

graças a existência na literatura, de inúmeros modelos diferentes a serem estudados e avaliados.

O PARADIGMA DOS MODELOS NEURAIS

Redes neurais são sistemas computacionais formados pela integração de inúmeros elementos de processamento (EP), funcionalmente muito simples, altamente interconectados e trabalhando massivamente em paralelo. Originalmente concebidas com base no estudo do cérebro humano e de suas conexões sinápticas, são **radicalmente** diferentes de todos os demais modelos computacionais. Uma rede neural pode ser vista, a grosso modo, como uma função matemática que mapeia um conjunto de padrões de entrada num conjunto de padrões de saída desejados. A rede neural é eminentemente numérica.

O sistema nervoso se constitui do cérebro, da medula espinhal e dos nervos. O cérebro e a medula espinhal formam o sistema nervoso central (SNC) – centro de controle e coordenação do corpo. Bilhões de longos neurônios, a maioria agrupados em nervos, formam o sistema nervoso periférico, transmitindo impulsos nervosos entre o SNC e as demais regiões do corpo. Cada neurônio possui três partes: corpo celular – composto por um núcleo e um citoplasma onde os estímulos de informação são integrados e onde a maioria do metabolismo celular é realizado –, dendritos – recebem os impulsos provenientes dos axônios de outros neurônios e os levam ao corpo celular, para a integração das informações, reiniciando novo ciclo – e axônio – encarregado da transmissão dos impulsos da célula para outros neurônios.

O cérebro humano é o maior órgão do sistema nervoso central, funciona como o centro deste sistema e controla todas as atividades do corpo, voluntárias e involuntárias. É também responsável por todas as operações do pensamento, incluindo: memória, emoções e a linguagem. Pesa cerca de 1,5 Kg no adulto humano e contém cerca de 10 bilhões de células nervosas, denominadas neurônios, que se comunicam entre si através de ligações eletro-químicas denominadas sinapses, densamente interligadas em uma malha. O cérebro comporta uma rede de conexões de 10 bilhões de neurônios com uma densidade de aproximadamente 10^4 sinapses por neurônio[1].

Um neurônio é capaz de produzir até 10000 sinapses, ou seja, até 10000 conexões com neurônios adjacentes. Cada neurônio, basicamente, capta estímulos nos dendritos ou dendrônios, os processa em seu corpo celular e, dependendo do seu estado de ativação, resultante dos estímulos recebidos, gera e transmite um estímulo pelos axônios ou cilindro-eixo para que atinja outros neurônios ou outros tipos de células, num processo altamente paralelo. O axônio, considerado uma fibra nervosa, é envolvido por uma bainha de mielina, externamente a qual pode existir outra, a bainha de Schwann ou neurilema.

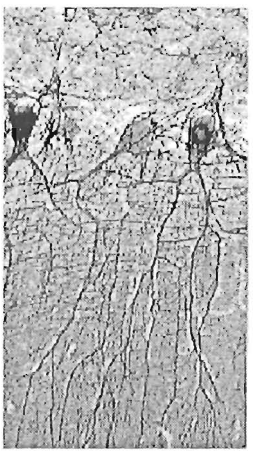


FIGURA 1 –
Neurônios do cérebro.

A figura 1 mostra os neurônios do cérebro, onde as células escuras são células de Purkinje, que estão entre as maiores células nervosas do corpo humano. Na figura 2 pode-se ver a estrutura de um neurônio motor e na figura 3 são apresentados, segundo o número de prolongamentos, os três tipos de neurônios conhecidos até o momento.

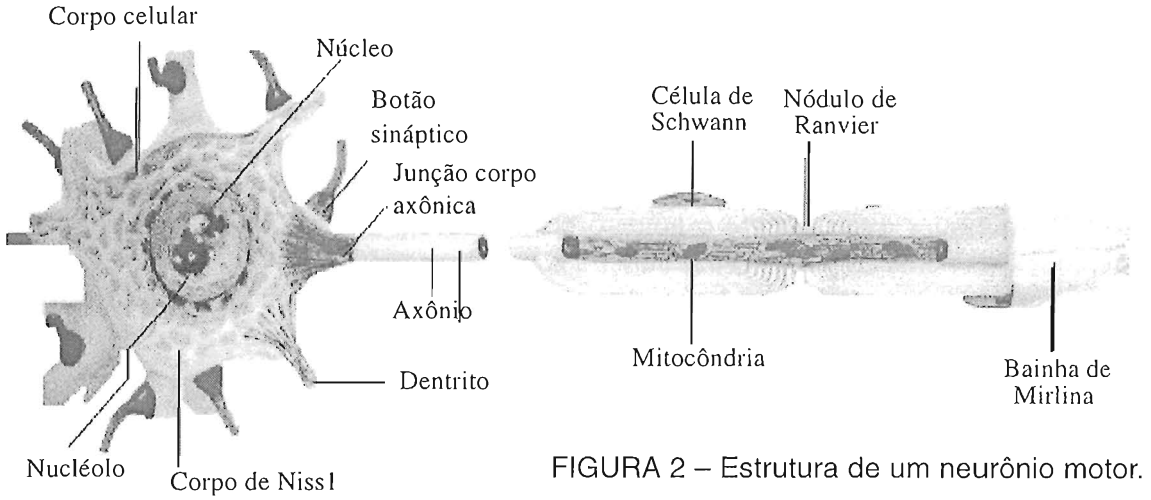


FIGURA 2 – Estrutura de um neurônio motor.

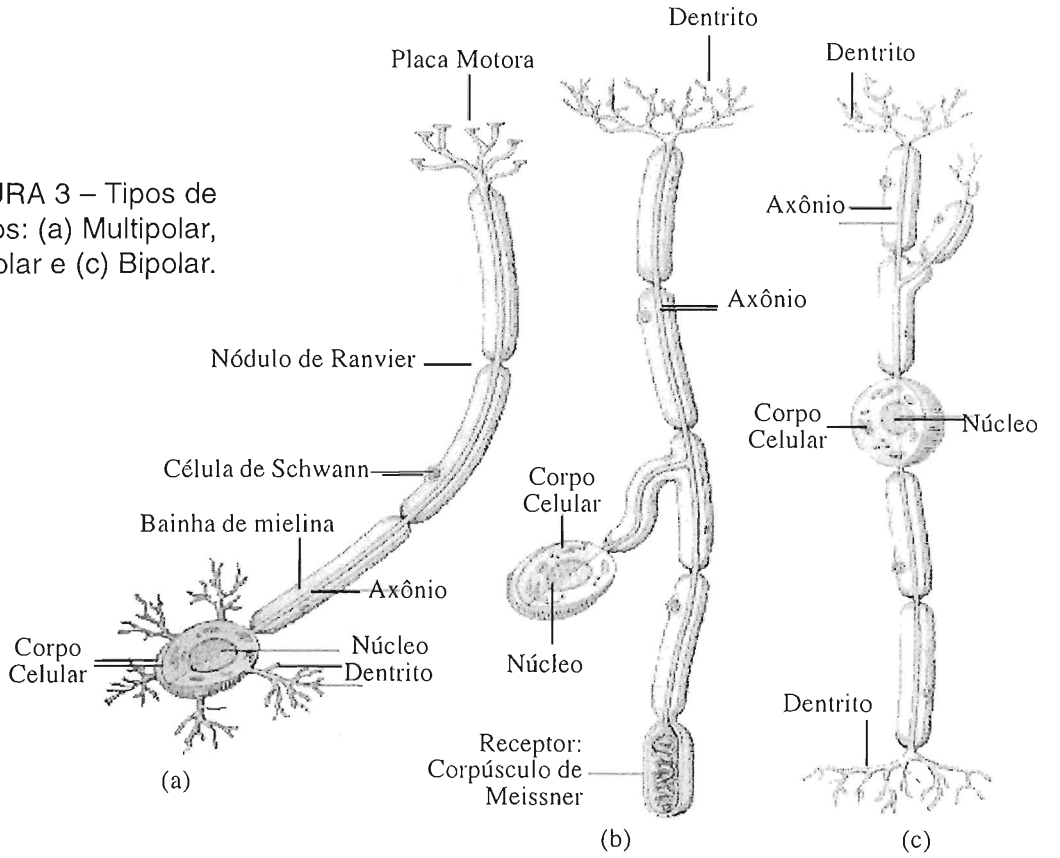


FIGURA 3 – Tipos de neurônios: (a) Multipolar, (b) Unipolar e (c) Bipolar.

Os neurônios multipolares, recebem estímulos de vários dendritos, retransmitem estes estímulos por apenas um axônio e seu corpo celular geralmente está situado no final do neurônio. Os neurônios unipolares possuem uma estrutura contendo fibras nervosas em espiral, conhecidas como "Corpúsculo de Meissner", que penetram na epiderme ou na língua, e são fortemente relacionadas com o sentido do tato, e seu corpo celular geralmente está situado em um ramo da célula nervosa. Os neurônios bipolares se caracterizam por seus impulsos nervosos que podem ser recebidos e transmitidos em apenas uma única direção, porque só possuem um dendrito e um axônio, e seu corpo celular geralmente está situado no meio da linha principal do neurônio.

A estrutura biológica do cérebro para a classe animal e insetos é apresentada na figura 4, onde no eixo x está presente o número de interconexões e no eixo y a quantidade de interconexões por segundo que cada classe é capaz de exercitar.

Inspirada na estrutura biológica do cérebro e no comportamento das células nervosas, a comunidade científica concebeu um modelo computacional formado a partir de elementos simples de processamento que, de forma análoga aos neurônios biológicos, realizam pequenas tarefas específicas, estão massivamente interligados e operam em paralelo. Tais modelos, embora concepcionalmente ilimitados, encontram restrições de implementação face o volume de processamento requerido e o atual estágio da tecnologia de "*hardware*" dos computadores.

MODELOS NEURAIS ESTUDADOS

A topologia de uma rede neural é bastante flexível, e diversos modelos têm sido sugerido na literatura em função dos problemas e aplicações a que se destinam, dentre eles têm-se o modelo *multilayer percepton* –MLP– com algoritmo de treinamento do tipo retropropagação do erro – "*backpropagation*" –, o modelo "*radial basis*" e o modelo paralelo auto-organizável –pshnn.

Uma rede neural pode possuir uma ou mais camadas, sendo que cada camada tem uma matriz de pesos W, um vetor de polarizações B ("*bias*"), uma função de ativação "a(.)", e uma função de propagação "p(.)". Funções estas que podem ser iguais ou não em todas as camadas. A última camada é chamada de camada de saída e todas as outras são comumente chamadas de camadas escondidas ou camadas intermediárias, podendo ainda ser a primeira delas chamada de camada de entrada. A única função da camada de entrada é armazenar a informação de entrada para ser passada para a camada seguinte de neurônios.

Não existe nenhuma regra que defina o número de camadas intermediárias, que pode variar entre 0 e n. O número de neurônios da camada intermediária é livre, não obedecendo nenhuma regra específica. Cabe ressaltar que um número grande de camadas intermediárias pode acabar fazendo com que a rede memorize os dados em vez de generalizá-los (TAFNER, 1996).

Dois principais tipos de estrutura compõem o universo de modelos de RNA's: as do tipo unidirecional ("*feedforward*") e as do tipo recorrente.

• Feedforward – neste tipo de estrutura todas as conexões entre neurônios diferentes – inter-camada – obedecem a direção entrada⟶saída, não havendo conexões entre neurô-nios de uma mesma camada – intra-camada. Esta estrutura é *"fully conected"*, pois todas as saídas dos neurônios de uma camada são conectadas com todos os neurônios da camada posterior. A figura 5 apresenta uma estrutura (*"feedforward"*) com quatro camadas.

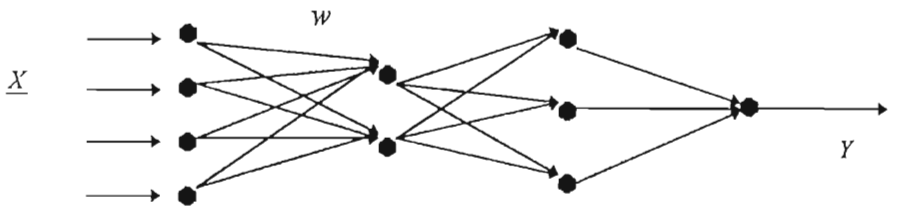


Figura 5 – Estrutura "*feedforward*", com 1 camada de entrada, 2 escondidas e 1 saída.

• Recorrente – são redes com realimentação, onde um neurônio pode ser direta ou indiretamente retroalimentado pela sua saída. Cada camada pode conter conexões entre os elementos de processamento da mesma camada (estímulos laterais), das camadas anteriores e das camadas posteriores. Na estrutura recorrente não existe um sentido único para o fluxo dos sinais entre neurônios ou entre camadas. A figura 6 apresenta uma estrutura recorrente de 1 camada.

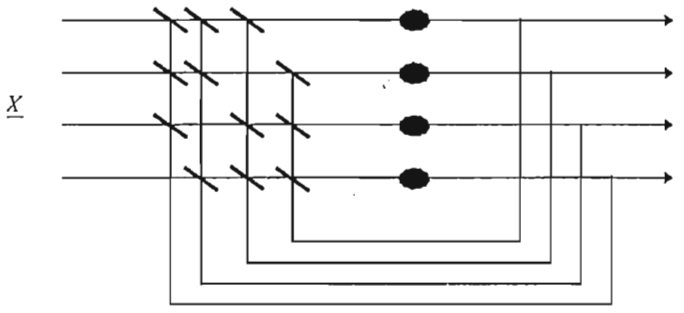


Figura 6 – Estrutura recorrente.

A operação de uma rede neural constitui-se de 3 etapas: treinamento – ajuste dos parâmetros do modelo –, teste – validação dos parâmetros do modelo – e produção – utilização do modelo.

Na etapa de treinamento é escolhido o algoritmo de aprendizado juntamente com os parâmetros de aprendizado. O aprendizado é o processo pelo qual a rede adapta seus parâmetros de forma a satisfazer os requisitos de mapeamento estabelecidos. Quanto a dinâmica de atualização dos parâmetros de treinamento, pode-se citar:

• Batch – os parâmetros são ajustados somente ao final de cada ciclo (*"epoch"*) – processamento de todo o conjunto de observações –, ou seja, a cada *"epoch"* os parâmetros da rede são ajustados. Nesta dinâmica, o treinamento é menos influenciado pela ordem de apresentação dos padrões e é menos suscetível à oscilações, porém a velocidade de aprendizado (convergência) geralmente é mais baixa.

• Incremental – os parâmetros são ajustados ao final do processamento de cada observação, ou seja a cada observação ajusta-se os parâmetros da rede. Nesta dinâmica, a ordem da apresentação dos padrões é importante para a velocidade de aprendizado da rede e, em alguns casos, deve-se reorganizar esta ordem, de forma a acelerar o treinamento.

O aprendizado geralmente se constitui no ajuste do conjunto de pesos de modo à executar uma tarefa específica, e acontece, basicamente, de duas formas distintas:

• supervisionado – utiliza um conjunto de pares (entrada - saída), em que para cada padrão de entrada é especificado um padrão de saída desejado (resposta desejada). O aprendizado ocorre no momento em que a saída gerada pela rede, a partir dos cálculos efetuados com o padrão de entrada e os pesos correntes, for diferente da saída desejada, a rede deverá então, segundo alguns critérios, ajustar seus pesos de forma a reduzir o erro. Essa dinâmica é repetida para todo conjunto de dados (entradas e saídas) inúmeras vezes, até que a taxa de erro esteja dentro de uma faixa considerada satisfatória.

• não-supervisionado – este tipo de aprendizado também é conhecido como aprendizado auto-supervisionado, e classifica os padrões similares sem utilizar pares (entrada - saída), isto é, no treinamento da rede são usados apenas valores de entrada. A rede trabalha essas entradas e se organiza de modo a classificá-las mediante algum critério de semelhança. Esse tipo de rede utiliza os neurônios como classificadores, e os dados de entrada como elementos de classificação.

Os parâmetros usados para aprendizado e armazenamento do conhecimento dependem do modelo de rede adotado. Quaisquer que sejam estes parâmetros, os métodos de ajustes dos mesmos são chamados de regras de aprendizado, que implementam na prática, um procedimento matemático de otimização que busca minimizar ou maximizar uma determinada função objetivo.

MODELO *"BACKPROPAGATION"*

O algoritmo *"backpropagation"* foi criado por Rumelhart, Hinton & Williams em 1986[1,2,3] a partir da generalização da regra de aprendizado "*Widrow-Hoff*", que fora introduzida por Bernard Widrow & Marcian Hoff em 1960-1962 para redes do tipo "*feedfoward perceptron*". A regra de aprendizado "*Widrow-Hoff*" também é conhecida como "Regra Delta" – LMS (minimização do erro médio quadrático) – que ajusta os pesos das conexões entre os neurônios da rede de acordo com o erro, ou seja, esta regra tem como objetivo encontrar um conjunto de pesos e polarizações que minimizem a função erro,

$$E = \frac{1}{2} \sum_{p=1}^{R} \sum_{i=1}^{S} (y_{p,i} - y_{p,i}) \qquad (1)$$

onde R = número de padrões ou vetores de entrada;
S = número de neurônios de saída – dimensão do vetor de saída;
$y_{p,i}$ = saída desejada no i-ésimo neurônio, quando o p-ésimo padrão é apresentado;
$y_{p,i}$ = saída obtida pela rede no i-ésimo neurônio, quando o p-ésimo padrão é apresentado.

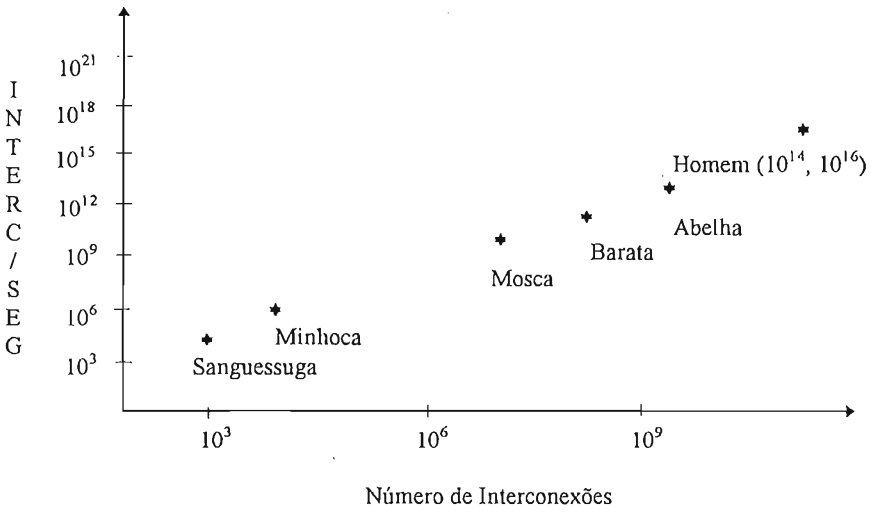


Figura 4 – Atividades cerebrais da classe animal e insetos.

A alteração dos pesos $W_{i,j}$ da regra de "*Widrow-Hoff*" é calculada da seguinte maneira:

$$\Delta W_{i,j} = -\eta \frac{\partial E}{\partial W_{i,j}} \tag{2}$$

onde, η = parâmetro da taxa de aprendizado e $\frac{\partial E}{\partial W_{i,j}}$ é a derivada parcial do erro em relação ao peso da respectiva conexão – gradiente. A principal restrição na minimização do erro no sentido do gradiente descendente é que a função de transferência do neurônio tem que ser monotônica e diferenciável em qualquer ponto.

O algoritmo *"backpropagation"* (BP) refere-se a uma regra de aprendizagem que consiste no ajuste dos pesos e polarizações da rede através da retropropagação do erro encontrado na saída. A minimização é conseguida realizando-se continuamente – a cada iteração – a atualização dos pesos e das polarizações da rede no sentido oposto ao do gradiente da função no ponto corrente, ou seja, proporcionalmente ao negativo da derivada do erro quadrático em relação aos pesos correntes. Trata-se portanto, de um algoritmo de treinamento

supervisionado, determinístico, de computação local, e que implementa o método do gradiente descendente na soma dos quadrados dos erros.

A topologia da arquitetura da rede que utiliza esta regra de aprendizagem é formada, geralmente, por uma ou mais camadas escondidas (intermediárias) de neurônios não-lineares (com função de propagação sigmoidal) e uma camada de saída de neurônios lineares. Devido a grande difusão da arquitetura da rede a que esta regra de aprendizagem se aplica, é comum referir-se a ela com o nome da própria regra de aprendizagem, ou seja, rede BP.

O algoritmo BP para um treinamento incremental pode ser descrito pelos seguintes passos:

Passo 1: Inicializar os pesos, as polarizações e os demais parâmetros de treinamento;

Passo 2: Apresentar à rede um padrão de entrada do conjunto de treinamento e computar a sua saída;

Passo 3: Calcular o erro para os neurônios da camada de saída, subtraindo a saída desejada da saída calculada;

$$e_k = (y_k - \hat{y}_k) \tag{3}$$

onde: y é a saída desejada e $\hat{y}$ a saída real (saída gerada pela rede).

Passo 4: Calcular o ajuste nos pesos da camada da saída com a fórmula:

$$\Delta w_i^o(k+1) = -\frac{\partial e_k^2}{\partial w_i^o(k+1)} = \eta.(y_k - \hat{y}_k).p'(neto(k)).a'(w_i^o(k), O_i^1(k)) \tag{4}$$

onde: p é uma função contínua derivável, comumente uma sigmoidal, η é a taxa de aprendizagem, net é o estado de ativação e O_i^1 é a entrada.

Passo 5: Retropropagar o erro para as camadas escondidas. Como não existe uma saída desejada para os neurônios das camadas escondidas, deve-se calcular o erro destes a partir do erro dos neurônios pertencentes à camada de saída e das conexões que os interligam. Têm-se assim, a seguinte equação para calcular o ajuste dos pesos para a primeira camada escondida mais próxima à saída.

$$\Delta w_{j,i}^1(k+1) = -\frac{\partial e_k^2}{\partial w_{j,i}^1(k+1)} = \eta.(y_k - \hat{y}_k)p'(net^o).\frac{\partial a}{\partial O_j^1(k)}(w_j^o(k), O_j^1(k)).p'(net_j^1).$$

$$.\frac{\partial a}{\partial w_{j,i}^1(k)}(w_{j,i}^1(k), x_i(k)) \tag{5}$$

Passo 6: Calcular o erro acumulado da rede. Nesta etapa, deve ser verificado se o erro total sobre todos os padrões de entrada pode ser considerado desprezível,

isto é, se caiu abaixo de um limiar de aceitação. Se assim for o caso, o algoritmo deve parar, caso contrário, deve-se voltar ao passo 2.

Múltiplas camadas de neurônios não-lineares permitem à rede BP aprender relações lineares e não-lineares existentes entre os valores de entrada e saída. Nesse caso, se a camada de saída for composta por neurônios lineares, a rede pode produzir valores fora do intervalo [0,1], no entanto, se for desejável restringir a saída da rede a valores no intervalo [0,1], por exemplo, a camada de saída poderá também consistir de neurônios sigmoidais.

Rede BP com polarizações, com no mínimo uma camada intermediária de neurônios lineares na saída, são teoricamente capazes de realizar a aproximação de qualquer função matemática, sendo ainda bastante utilizadas na associação e classificação de padrões.

Enquanto redes lineares possuem apenas um mínimo em suas superfícies de erro, redes não-lineares – como a BP – podem apresentar vários mínimos locais em diferentes níveis, tal como numa cadeia de montanhas onde pode-se encontrar vários vales em altitudes diferentes em relação ao nível do mínimo global (se existir). O processo de minimização do erro em uma rede neural, atua no sentido de encontrar o mais baixo de todos os vales, ou seja, o mínimo global. Nem sempre isto é possível pois, dependendo do ponto da superfície de erro – caracterizado pelos valores iniciais dos pesos e polarizações – de onde o treinamento é iniciado, o método do gradiente descendente poderá convergir para um dos mínimos locais ao invés do mínimo global. Caso um mínimo local encontrado não seja satisfatório para a aplicação desejada, uma regra prática consiste em se treinar outras redes – com as mesmas entradas e saídas –, variando-se os pesos e adicionalmente pode-se variar também outros parâmetros relevantes.

A figura 7 apresenta a configuração de uma rede BP de 4 camadas (1 de entrada, 2 escondidas e 1 de saída), onde a matriz de entrada tem dimensão m e a matriz de saída tem dimendão n. Segundo um critério heurístico escolhe-se a topologia da rede que nesta exemplificação é: n1 neurônios para a 1ª camada escondida, n2 neurônios para a 2ª camada escondida e n neurônios para a camada de saída.

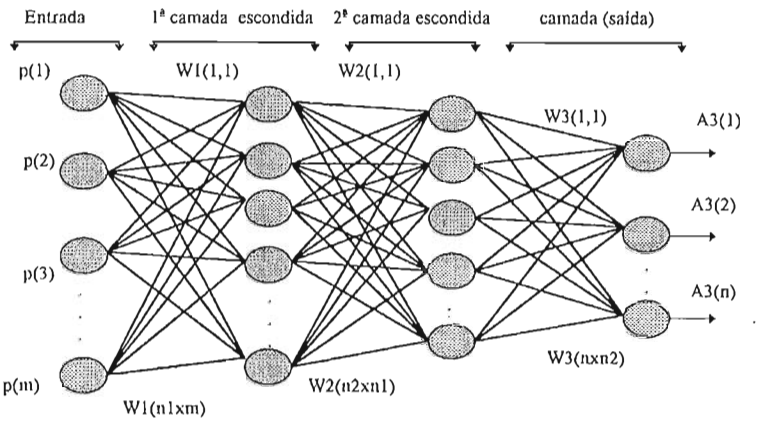


FIGURA 7 – Arquitetura da rede *"backpropagation"* utilizada no reconhecimento dos 10 comandos isolados proposto nesta tese.

MODELO *"RADIAL BASIS"*

Broomhead & Lowe (1988) foram os primeiros à explorarem o uso da função *"radial basis"* no projeto de redes neurais[2]. Outras contribuições de teoria, projeto, e aplicação da função *"radial basis"* em redes incluem os artigos de Moody & Darken (1989)[4], Renals (1989), e Poggio & Girosi (1990a). O artigo de Poggio & Girosi enfatiza o uso da teoria de regularização aplicada a esta classe de redes neurais como um método aperfeiçoado de generalização à novos dados. A rede *"radial basis"* é um tipo de rede neural artificial para aplicação em problemas de aprendizado supervisionado – regressão, classificação e predição de série temporal.

A arquitetura básica de uma rede função *"radial basis"* consiste basicamente de três camadas completamente diferentes, sendo a 1ª camada de entrada, a 2ª camada escondida de dimensão bastante alta – camada gaussiana –, que possui finalidade diferente do *"perceptron"* multi-camada, e a 3ª camada de saída constituída de uma camada linear com os pesos ajustados pelo método do mínimo quadrado, onde a transformação do espaço dimensional da camada de entrada para a camada escondida é não linear, enquanto que a transformação do espaço dimensional da camada escondida para a camada de saída é linear, conforme mostra a figura 8.

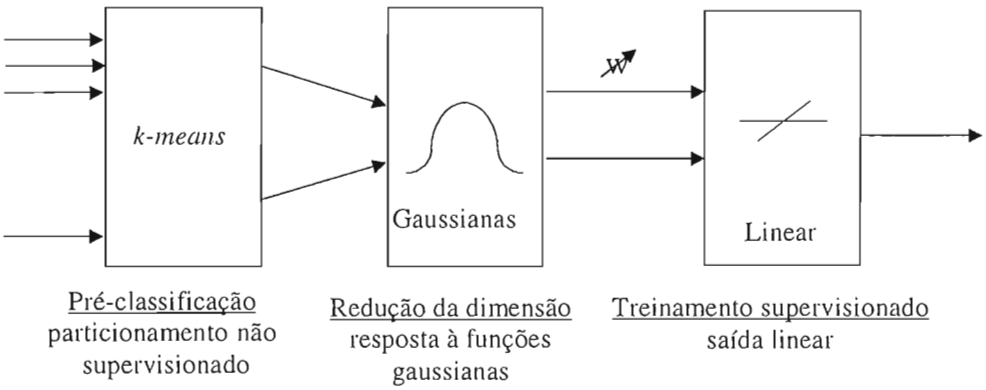


Figura 8 – Arquitetura básica da implementação da rede "*radial basis*" onde $\not{w}$ simboliza peso fixo e w simboliza peso ajustável.

A partir da arquitetura básica apresentada – figura 8 –, optou-se, para os trabalhos aqui desenvolvidos, em fazer um estudo de duas configurações ligeiramente distintas destas. Na 1ª configuração introduziu-se algumas pequenas alterações com vistas a se conseguir um melhor desempenho no treinamento e reconhecimento dos comandos à voz – figura 9 –, onde a camada linear de saída foi substituída por uma não linear (sigmóide), e uma 2ª camada não linear – 2º estágio – foi adicionada. A 2ª configuração é uma simplificação da 1ª e consiste de apenas uma única camada não linear (figura 10).

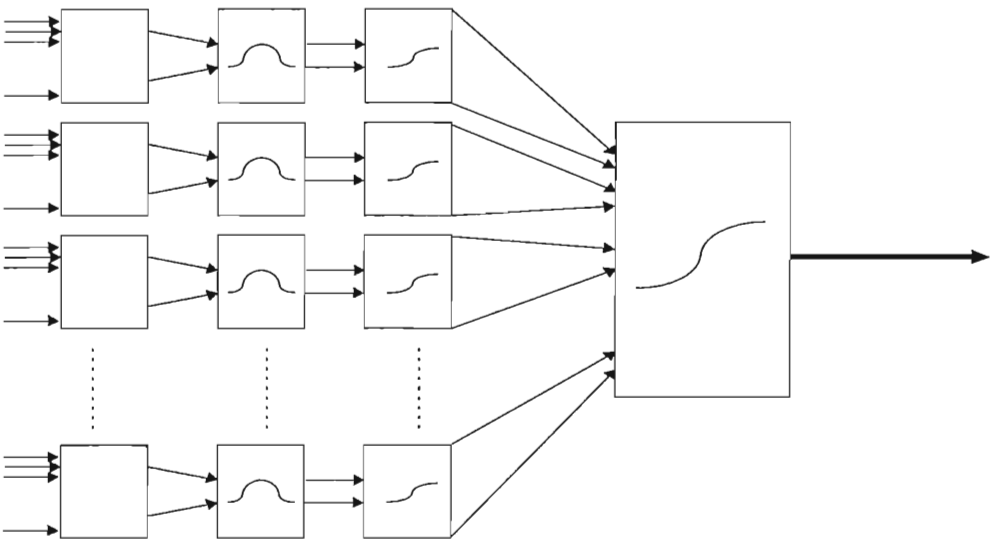


Figura 9 – Arquitetura número 1 da rede *"radial basis"*.

A construção das duas configurações foi baseada na rede tipo *"radial basis"* (RBF) e pode também ser dividida em três blocos, onde no primeiro bloco é realizado um particionamento não supervisionado dos dados, utilizando-se um algoritmo de clusterização do tipo *k-means*. No segundo bloco é realizada uma redução da dimensionalidade do problema pelo emprego de uma camada de n funções gaussianas, definidas a partir do resultado da pré-classificação. O número de gaussianas corresponde ao número de *"clusters"* – grupos – usados no *k-means*, e seus respectivos centros são escolhidos coincidentes aos centros de massa destes mesmos grupos. O terceiro bloco é responsável pelo treinamento de uma rede MLP – saída não linear – utilizando a regra de aprendizado *"backpropagation"*, este treinamento é feito de forma supervisionada.

Na pré-classificação não supervisionada optou-se pelo uso do algoritmo *k-means*, que proporciona um rápido particionamento. Este algoritmo é baseado na distância euclidiana entre as amostras da massa de dados, e necessita que se forneça previamente o número C de classes em que se deseja particionar o hiper-espaço. Uma vez separados os elementos, é calculada a distância euclidiana de cada elemento a cada centróide – centro do *"cluster"* –, se existir entre tais distâncias uma que seja menor do que a distância ao seu centro, será atribuído então o elemento àquele *"cluster"* que apresentou menor distância. Novos centróides são calculados, iniciando assim o ciclo de trocas, quando não ocorrerem mais trocas, a execução acaba, e o particionamento dos dados está feito. Neste trabalho avaliou-se vários números de "*clusters*" e o que mostrou melhor desempenho foi aquele igual ao número de comandos do vocabulário, que no estudo de cada caso realizado e a ser descrito no artigo 3, é 10.

A camada de gaussianas é criada a partir dos centros de massa e espalhamentos dos diversos grupos identificados pelo algoritmo de particionamento. A função da camada de

gaussianas é fornecer um valor escalar correspondente ao decaimento exponencial de cada observação em relação ao centro de massa das mesmas. Esta função é dada pela seguinte expressão:

$$f_{\underline{x}} = \frac{1}{(2\pi)^{N/2} \prod \sigma_j} \exp\left\{ -\frac{1}{2} \sum_{j=1}^{n} \left(\frac{x_j - c_j}{\sigma_j} \right)^2 \right\} \qquad (6)$$

onde o somatório dentro da exponencial é oriundo do produto das exponenciais referente a mistura de gaussianas[5].

Devido ao fato dos valores retornados terem sido muito baixos em função do espaço dimensional do problema em questão ser muito elevado, decidiu-se descartar da fórmula a constante de multiplicação, e como mesmo assim, os valores de $f_{\underline{x}}$ continuavam baixos, resolveu-se aplicar um fator de normalização – empírico – conforme a expressão:

$$f_{\underline{x}} = \exp\left[-\sum_{j=1}^{n} (x_j - c_j)^2 / 1000\sigma_j^2 \right] \qquad (7)$$

onde: x_j é o valor da j-ésima dimensão do vetor de entrada;

c_j é o valor da j-ésima dimensão do centro de massa da gaussiana;

σ_j^2 é o valor da variância para j-ésima dimensão da gaussiana.

Os dados para treinamento da rede foram dispostos em uma matriz sendo m a dimensão dos vetores amostra e n a sua quantidade, onde cada coluna representa a locução de um comando. A dimensão m é composta por *"x"* características do sinal de voz, extraídas em *"y"* janelas temporais, ou seja: $m = x.y$. Os dados foram subdivididos de forma a gerar X matrizes de y linhas por n colunas, sendo que cada uma destas matrizes X_i é composta pelas y extrações temporais da característica x_i em questão. Cada uma das matrizes foi então submetida à arquitetura básica, em uma tentativa de classificar as amostras utilizando-se apenas as informações de uma única característica. É claro que o alcance da precisão desejada nesta fase seria uma tarefa quase impossível, desta forma, as saídas obtidas neste estágio são submetidas a uma nova rede MLP – 2º estágio (2ª camada não linear) –, que se encarrega em realizar a classificação final a partir da combinação dos resultados parciais alcançados no primeiro estágio. Este modelo proposto é apresentado na figura 10, onde c_1 representa a primeira característica e c_n a n-ésima característica.

A segunda configuração nada mais é que uma simplificação da primeira, onde a camada MLP é eliminada do módulo básico, ficando assim, uma única rede MLP para fazer a classificação. A disposição dos dados e a sua ordenação permanecem idênticas às apresentadas na primeira configuração. Esta arquitetura é apresentada na figura 10.

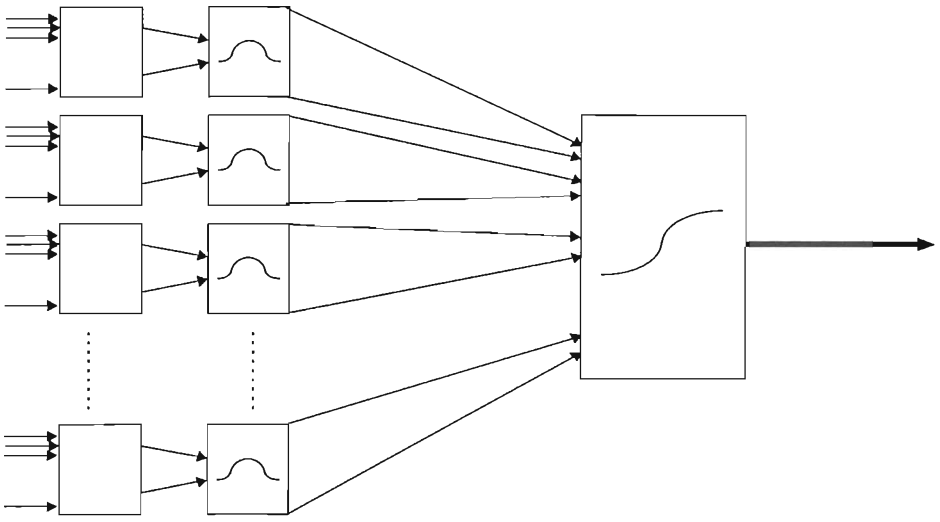


Figura 10 – Arquitetura número 2 da rede *"radial basis"*.

A 1ª configuração apresenta a vantagem em relação a 2ª, de poder ser utilizada como um discriminante não linear – PCA –, pois através dos resultados obtidos no 1º estágio pode-se verificar o quanto uma característica é relevante para distinguir uma palavra da outra. A vantagem da 2ª configuração está no tempo necessário para o treinmento, geralmente menor.

MODELO PARALELO AUTO-ORGANIZÁVEL (PSHNN)

PSHNN's são redes multi-estágio onde os diferentes estágios são treinados de forma sequenciada e utilizados em paralelo após o ajuste final do modelo. Sugerido por Ersoy & Hong[6], o modelo PSHNN implementa, na realidade, um comitê de redes neurais onde, na teoria, cada um dos membros do comitê pode ser um tipo particular e diferente dos demais.

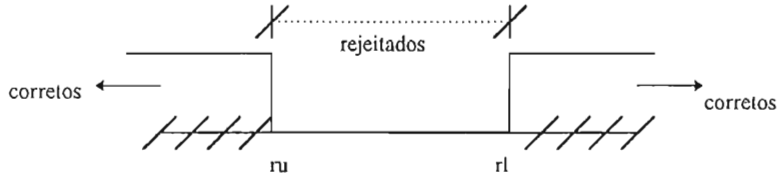
O modelo é dito hierárquico porque embora composto por vários membros, a resposta dada a um determinado estímulo de entrada é fornecida por apenas um dos membros, selecionado segundo a hierarquia existente. Ou seja, a resposta será fornecida pelo 3º membro da hierarquia do comitê somente se ela não puder ser fornecida pelo 1º ou 2º membro do comitê.

A figura 11 mostra a estrutura – diagrama em bloco – da rede PSHNN utilizada no treinamento. O treinamento do modelo PSHNN envolve os seguintes passos:

Passo 1: Treinar uma rede MLP com algoritmo *"backpropagation"* por um máximo de n iterações (*"epoch"*);

Passo 2: Verificar a saída para cada vetor de entrada. Se alcançar o erro desejado, o treinamento da rede é finalizado;

Passo 3: Determinar os limiares de aceitação dos vetores de entrada da rede, limiar superior – ru – e limiar inferior – rl . Se todos os elementos escolhidos estão fora do limite de rejeição, isto é, nenhum elemento foi rejeitado, termina o treinamento; caso contrário continue;



Passo 4: Selecionar os elementos que estão dentro do limite de rejeição, onde há possibilidade de ter elementos classificados corretamente e elementos não classificados. Aplicar uma transformação no conjunto de elementos rejeitados, selecionar uma nova rede e retornar ao passo 1.

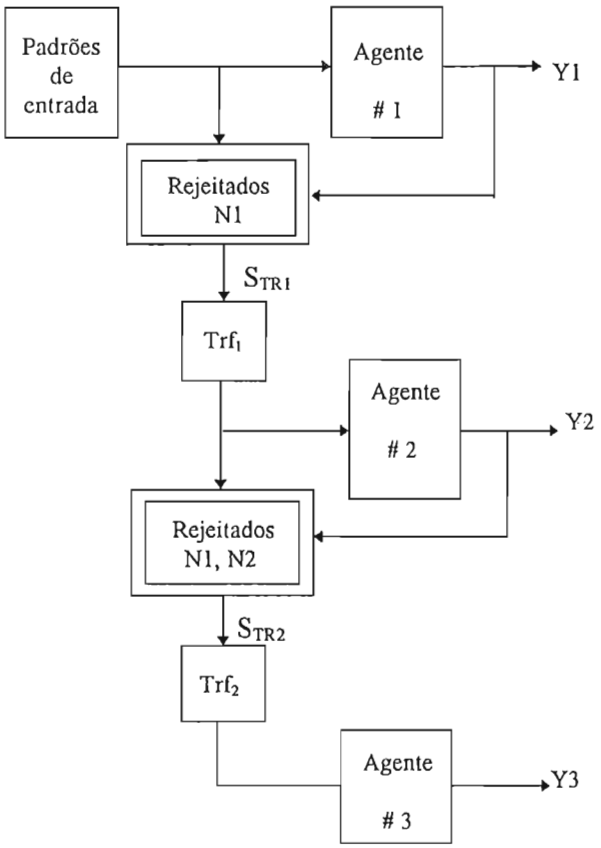


Figura 11 – Diagrama em bloco do procedimento do treinamento paralelo com a rede PSHNN, quando o número de estágios da rede é igual a 3.

Depois de treinar a rede até um máximo de iterações, é necessário gerar os limiares superior e inferior para posterior seleção – aceitação – dos vetores de entrada. Para o cálculo destes limiares segue-se os seguintes passos:

Passo 1: Binarizar o vetor de saída da rede;

Passo 2: Gerar um vetor contendo um limiar superior e um limiar inferior de valores 0.5 cada, para posterior ajuste;

Passo 3: Comparar o vetor de saída (binarizado) com o vetor de saída desejado (*"target"*). Se forem iguais, assume-se que os limiares superior e inferior são os atribuídos no passo 2, e se forem diferentes, seleciona-se as posições onde ocorreram tais diferenças e faz-se o ajuste dos limiares de forma a rejeitar o vetor de entrada.

A binarização dos vetores de saída da rede para comparar com o vetor *"target"* é obtida seguindo-se o seguinte critério:

$$Y_i = \begin{cases} 1, & se\ y_i \geq 0.5 \\ 0, & se\ y_i < 0.5 \end{cases} \qquad (8)$$

onde: y_i representa a i-ésima posição do vetor de saída gerado pela rede e Y_i a i-ésima posição deste mesmo vetor binarizada.

O ajuste dos limiares superior e inferior é feito selecionando as posições onde a saída da rede binarizada é diferente do *"target"* e buscando o valor do vetor de saída da rede – sem binarização – na posição referida. Se este valor for maior que ou igual a 0.5, faz-se então o ajuste no limiar superior, sempre pelo maior valor, caso contrário, faz-se o ajuste no limiar inferior, sempre pelo menor valor. Estes limiares podem ser usados durante o teste no sentido a não classificar àqueles vetores que tenham ficado dentro dos limites de rejeição.

Com os limiares já ajustados faz-se a seleção dos vetores de entrada para o próximo agente do modelo, pegando todos aqueles que tenham ficado dentro dos limites de rejeição. É interessante observar que o conjunto de todos os elementos que estão dentro do limite de rejeição conterá todos aqueles elementos incorretos e poderá conter alguns corretos cujo grau de confiança não tenha ficado evidente. O conjunto desses vetores rejeitados serão a entrada do próximo estágio da rede, depois de serem processados por uma técnica de transformação – linear ou não linear.

A figura 12 mostra a estrutura – diagrama em bloco – da rede PSHNN utilizada no teste. O teste do modelo PSHNN envolve os seguintes passos:

Passo 0: $\longrightarrow$ 1;

Passo 1: Gerar o vetor de saída para o estágio N_i da rede.

Passo 2: Verificar se na saída da rede existe algum elemento "rejeitado"; isto é dentro dos limites fixados por ocasião do treinamento daquele agente, se indicar: caso seja o último estágio da rede classifique-o como rejeitado; e se não for o último, passe o vetor teste pela transformada obtida no treinamento, faça i $\longleftarrow$ i+1 e retorne ao passo 1; caso a saída não indique nenhum elemento rejeitado, classifique este como o vetor saída.

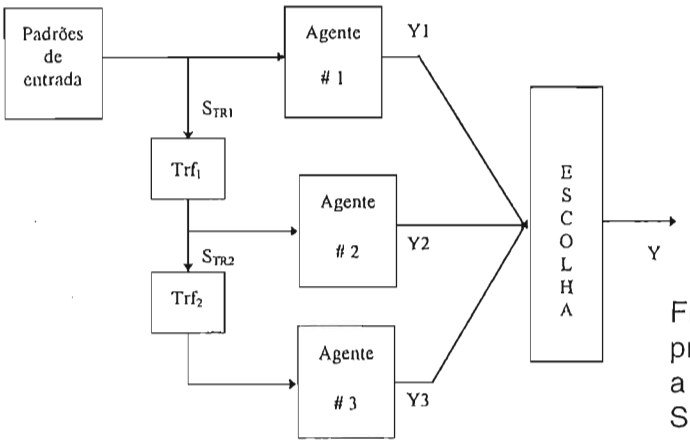


FIGURA 12 – Diagrama em bloco do procedimento do teste paralelo com a rede PSHNN, quando o número de SNN's é igual a 3.

Após esta breve apresentação de alguns modelos neurais (parte II), assim como do RAV (parte I), será apresentado no artigo seguinte (parte III) o uso de tais modelos de rede no reconhecimento de palavra isolada –RPI–, onde será feito um estudo de cada caso realizado e uma avaliação dos resultados alcançados.

REFERÊNCIAS BIBLIOGRÁFICAS

1. ZURADA, Jacek M., Introdution to Artificial Neural Systens, West Publishing Company, 1992.
2. HAYKIN, Simon, Neural Networks – A Comprehensive Foundation, Prentice Hall, 1994.
3. McCLELLAND, J. L. & RUMELHART, D. E., Exploratins in Parallel Distribuited Processing - A Handbook of Models, Programs, and Exercises, The Massachussetts Institute of Technology, 1988.
4. MOODY, J. & DARKEN, C. J., Fast Learning in Networks of Locally-Tuned Processing Units, Neural Computation, vol.1, pp. 281-294, 1989.
5. HAYKIN, Simon, Neural Networks – A Comprehensive Foundation, Prentice Hall, 1994.
6. ERSOY, K. O. & HONG, D., Parallel, Self-Organizing, Hierarquical Neural Networks, School of Eletrical Engineering Purdue University - West Lafayette, Indiana 47907, Setembro 1989.
7. DUDA, O. Richard & HART, E. Peter, Pattern Classification and Scene Analysis, Wiley-Interscience, 1973, pp.114-118.
8. RABINER, Lawrence & JUANG, Biing-Hwang, Fundamentals of Speech Recognation, Prentice-Hall, Inc., 1993.
9. DELLER, John R. JR, PROAKIS, John G. & HANSEN, John H. L., Discret Time Processing of Speech Signals, Macmillan Publishing Company, New York, 1993.