Aprendizado Não-Supervisionado em Redes Neuronais

Luiz Adauto F. C. Pessoa

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profa. Sueli B. T. Mendes, Ph. D.
(presidente)

Profa. Dóris Ferraz de Aragon, D. Sc.

Prof. Valmir Carneiro Barbosa, Ph. D.

RIO DE JANEIRO, RJ – BRASIL

JULHO DE 1990

PESSOA, LUIZ ADAUTO FILIZOLA CARNEIRO

Aprendizado Não-Supervisionado em Redes Neuronais [Rio de Janeiro] 1990

X, 129 p., 29.7 cm, (COPPE/UFRJ, M. Sc., ENGENHARIA DE SISTEMAS E COMPUTAÇÃO, 1990)

TESE – Universidade Federal do Rio de Janeiro, COPPE

1 – Redes Neuronais 2 – Aprendizado 3 – Algoritmos Não-Supervisionados

I. COPPE/UFRJ II. Título(Série).

À Joísa

Agradecimentos

À minha orientadora Sueli Mendes meus sinceros agradecimentos por ter me introduzido na área de redes neuronais e pela ajuda e amizade ao longo dos anos.

Ao professor Valmir pela sua ajuda e amizade, sempre.

Ao Pedro Paulo, e em especial à Cláudia, pelo apoio no dia-a-dia, pela leitura, correção e melhoria de muitos capítulos da tese, e principalmente pelas suas amizades.

Ao Juarez, uma verdadeira "legenda" viva do Tex, pela ajuda com este sistema, pelas inúmeras correções de "erros" da tese, e pela sua amizade.

Ao meu amigo Ricardo Bianchini por sua amizade.

À Cláudia e à Denise da secretaria de Sistemas, pela inestimável ajuda em tantas situações, e pelas suas amizades.

A tantos amigos e colegas da COPPE pela companhia ao longo do meu mestrado.

Por último, à Joísa, por incontáveis motivos.

Resumo da Tese apresentada à COPPE como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.)

Aprendizado Não-Supervisionado em Redes Neuronais

Luiz Adauto F. C. Pessoa

Julho de 1990

Orientadora: Sueli B. T. Mendes

Programa: Engenharia de Sistemas e Computação

Neste trabalho estudamos os principais aspectos do aprendizado em redes neuronais. Damos epecial ênfase ao aprendizado não-supervisionado, apresentando com bastante detalhe alguns importantes algoritmos desta classe: O aprendizado competitivo, a auto-organização de mapas de características, e os sistemas ART. Foi feita, também, uma avaliação experimental dos algoritmos acima em uma aplicação particular: A categorização de letras. Finalmente, traçamos conclusões e indicamos possíveis tendências da área de aprendizado em redes neuronais.

Abstract of Thesis presented to COPPE as partial fulfillment of the requirements for the degree of Master of Science (M. Sc.)

Unsupervised Learning in Neural Networks

Luiz Adauto F. C. Pessoa

July, 1990

Thesis Supervisor: Sueli B. T. Mendes

Department: Programa de Engenharia de Sistemas e Computação

In this work we study the main concepts of learning in neural networks. Special emphasis is given to non-supervised learning, and we study some important algorithms in this class: Competitive learning, Self-organization of feature maps, and ART systems. An evaluation of the above algorithms in a particular application, the categorization of letters, was done. Finally, we conclude and mention possible trends in the area of learning in neural networks.

Índice

Capítulo I

Introdução

I.1 O Que É Aprendizado?

Aprender: 1. Tomar conhecimento de. 2. Reter na memória, mediante o estudo, a observação ou a experiência. 3. Tornar-se apto ou capaz de alguma coisa (FERREIRA, 1975, p. 119).

Aprendizado, ou aprender, é um destes termos de difícil conceituação. Entretanto, as inúmeras definições existentes possuem um núcleo, ou idéia comum, que está presente até na simples definição acima do dicionário Aurélio: o aprendizado envolve *mudança*, associada a *aperfeiçoamento.* Por exemplo, uma definição mais técnica (SHEPHERD, 1988) afirma que *o aprendizado é uma mudança adaptativa no comportamento causada por experiência.* Apesar dos termos acima possuirem significado preciso para o neurobiólogo, as idéias de mudança e aperfeiçoamento estão claramente presentes. Uma definição operacional, bastante usual em Inteligência Artificial (IA), define o aprendizado como a habilidade de realizar novas tarefas que não podiam ser realizadas anteriormente, ou realizar melhor tarefas antigas, como resultado de mudanças produzidas no processo de aprendizado (CARBONELL, 1989). Mais uma vez, as idéias chave estão presentes.

O problema é ainda mais grave se observarmos que aprendizado é um termo de uso corriqueiro no dia-a-dia de todos. É preciso deixar claro de início, então, que apesar desta ser uma tese acerca de aprendizado, no presente contexto este termo possui um significado bastante claro, preciso, e também, restrito. Es-

tamos interessados em um tipo de aprendizado particular: o *aprendizado em redes neuronais* (ou aprendizado conexionista). Mais especificamente, vamos nos concentrar em uma das possíveis formas de aprendizado em redes neuronais: *o aprendizado não-supervisionado.*

Um outro aspecto que precisa ser enfatizado é que até recentemente o tipo de aprendizado que ocorre em redes neuronais era tido como bastante primitivo. Por exemplo, Fischler e Firschein consideram esta forma de aprendizado como *aprendizado de parâmetros*, e situam-na em um plano bastante inferior ao que eles chamam de *aprendizado de conceitos.* E concluem negativamente (FISCHLER e FIRSCHEIN, 1987, p. 140): "It should be recognized, however, that adjustment of weights ... is not the critical issue in building a learning system. What we require is that the corresponding recognition rule generalizes distinctions represented by the training set so as to be applicable to new members of a class. Since the weight-adjustment learning algorithms do not address this question, even sucessful generalization on a few selected problems provides very little assurance that this type of approach can be sucessful on a wide range of real problems."

Com a verdadeira explosão das pesquisas sobre redes neuronais, o paradigma de aprendizado conexionista vem ganhando respeitabilidade. De fato, em uma edição especial recente do importante periódico "Artificial Intelligence" de 1989 dedicado ao aprendizado em máquina, CARBONELL (1989) destaca quatro paradigmas principais de aprendizado, dentre os quais o conexionista.

Apesar deste reconhecimento mais recente, o infindável debate *Conexionismo* × *Simbolismo* também atua nesta área particular. Para citar apenas um exemplo importante, FODOR e PYLYSHYN (1988) atacam duramente a concepção conexionista de aprendizado.

O estudo e o debate sobre as "essências" do que é aprendizado são, sem dúvida, extremamente importantes. No entanto, os objetivos desta tese são por demais modestos para abranger tais questões. Estamos interessados apenas em *algumas* questões — as quais julgamos importantes — do aprendizado em redes neuronais.

I.2 Uma Introdução às Redes Neuronais

Nesta seção daremos uma breve introdução às redes neuronais – baseada em RUMELHART et al. (1986a). Apresentaremos os componentes básicos destas redes e como estes se interrelacionam. Alguns textos introdutórios recomendados são (CARPENTER, 1989), (KOHONEN, 1987), (LEVINE, 1983), e o recente livro de SIMPSON (1990).

Uma rede neuronal é composta por uma série de *unidades de processamento*. Estas unidades possuem ligações entre si determinando o *padrão de interconexão*. As ligações possuem pesos, os quais determinam a influência de uma unidade na outra. Por exemplo, w_{ij} significa que a unidade i está ligada à unidade j, e que esta ligação possui peso w_{ij} (de i para j). Os pesos das ligações entre unidades são características dinâmicas (pelo menos na fase de aprendizado) da rede, e podem sofrer modificações de acordo com a experiência do sistema. A maneira pela qual os pesos das ligações se alteram, e desta forma o padrão de interconexão muda, é chamada de *regra de aprendizado*.

Em um determinado instante de tempo t, o sistema pode ser caracterizado por um estado, o *estado de ativação*, o qual é determinado pelo valor de ativação individual, $a_i(t)$, de cada unidade i da rede. O valor $a_i(t)$ é dado pela *regra de ativação*, a qual combina as entradas que chegam a uma unidade com o seu próprio valor de ativação, produzindo, então, um novo valor para $a_i(t)$. A ativação de uma unidade é propagada pelo sistema de acordo com uma função f que determina o valor de saída da unidade i no instante t, $o_i(t)$. A função f é chamada *função de saída*. É preciso ainda, uma regra que combine as saídas com os pesos das ligações para produzir a entrada efetiva para cada unidade. Esta é a chamada *regra de propagação*. E, finalmente, há um *ambiente* onde o sistema deve operar.

Resumindo, podemos dizer que uma rede neuronal pode ser descrita por oito elementos principais:

- *um conjunto de unidades de processamento;*

- *um estado de ativação;*
- *uma função de saída;*
- *um padrão de interconexão;*
- *uma regra de propagação;*
- *uma regra de ativação;*
- *uma regra de aprendizado; e*
- *um ambiente onde o sistema deve funcionar.*

Apresentaremos, agora, mais detalhadamente cada um destes aspectos.

Unidades de processamento. Os "significados" das unidades (ou grupo de unidades) da rede são o único meio de representação existente. Dependendo da aplicação, as unidades podem representar objetos variados, tais como "pixels" (pontos), letras, palavras, ou outros conceitos mais complexos.

Podemos dizer que há, basicamente, dois tipos de representação possíveis. Em um extremo, encontramos sistemas onde uma única unidade representa um conceito (uma-unidade-um-conceito). Neste caso, se a unidade correspondente estiver ativada, podemos dizer que o conceito foi "estimulado". Este tipo de representação é dito *local.* No extremo oposto, o padrão de ativação de um conjunto de unidades como um todo é que possui significado. Ou seja, os valores de ativação de diversas unidades determinarão o conceito que está sendo representado. Dizemos que estes sistemas empregam representação *distribuída.*

Deve ser enfatizado que todo o processamento do sistema é realizado pelas unidades. Não há um executivo central. Há somente unidades simples, cada qual fazendo o seu trabalho, qual seja, o de receber estímulos das unidades vizinhas e computar um valor de saída para ser enviado às outras unidades. Diversas unidades podem estar realizando suas computações ao mesmo tempo, paralelamente.

Estado de ativação. O estado do sistema em um instante qualquer t pode ser representado por um vetor $\mathbf{a}(t)$. O padrão de ativação de todas as unidades

da rede, ou seja, o estado de ativação do sistema, é que especifica o que o que está sendo representando no instante t. O "processamento" do sistema pode ser visto como a evolução, através do tempo, dos valores de ativação das unidades. Neste sentido, a evolução do sistema pode ser vista como a trajetória descrita pelo vetor **a** ao longo do tempo.

A ativação de uma unidade é dada por um valor numérico. A escolha de que tipos de valores empregar dependerá domodelo em particular. Muitos modelos utilizam valores contínuos, e as unidades podem assumir qualquer valor real. É comum restringir esses valores a algum intervalo, por exemplo $[0, 1]$. Mas existem também ativações discretas, mais comumente $\{0, 1\}$ e $\{-1, +1\}$.

Função de saída. Cada unidade interfere com suas vizinhas através de um valor transmitido para essas unidades, o qual é determinado pela ativação da unidade estimuladora. O valor exato é dado por uma função de saída f. Mais formalmente, $o_i(t) = f(a_i(t))$. A escolha da função f é de extrema importância para o funcionamento do sistema.

Padrão de interconexão. Toda unidade possui uma série de ligações. É conveniente representar o padrão de interconexão por uma matriz de pesos, W, onde um elemento, por exemplo, w_{ij}, indica a intensidade e o sentido da ligação entre as unidades i e j. A convenção é que w_{ij} representa a influência da unidade i na unidade j. Se a unidade i excitar a unidade j, o peso w_{ij} será um número positivo, se i inibir j um número negativo, e se i não possuir conexão direta com j valerá 0. O valor absoluto de w_{ij} dá a intensidade da ligação, ou seja, o quanto uma unidade interfere numa outra.

Regra de propagação. As saídas das unidades devem ser combinadas com os pesos das ligações para produzir a *entrada líquida* (net) nas unidades ligadas. Isto é realizado pela regra de propagação, a qual é normalmente bastante simples. Tipicamente, a entrada líquida é expressa como a soma ponderada (pelos pesos) dos estímulos chegando à unidade. Assim,

$$net_j = \sum_i w_{ij} o_i,$$

onde net_j é a entrada líquida da unidade j. Salvo menção explícita, assumiremos

que a entrada líquida é dada pela expressão acima.

Regra de ativação. É preciso uma regra para calcular o valor da ativação de uma unidade no instante t. Para tanto emprega-se uma função F, a qual combina a entrada líquida da unidade com a sua ativação corrente para calcular a nova ativação. No caso mais simples $a_i(t+1) = net_i(t)$. Ou seja, a nova ativação da unidade será igual à sua entrada líquida. No caso mais geral, no entanto, o novo valor de ativação será da forma $a_i(t+1) = F(a_i(t), net_i(t))$. A função F é a própria regra de ativação.

Regra de aprendizado. Uma das características mais interessante das redes neuronais é a capacidade de aprendizado ou adaptação, a qual é obtida através da mudança dos pesos das ligações da rede.

Ambiente. É importante estabelecer precisamente a natureza do ambiente onde a rede deve funcionar. Mais especificamente, é preciso estabelecer os possíveis padrões de entrada e as respectivas probabilidades destes serem apresentados à rede.

Na situação mais geral, o ambiente é representado como uma função estocástica variável no tempo sobre o espaço de padrões de entrada. Ou seja, a cada instante há uma certa probabilidade de um padrão ser apresentado à rede, sendo que esta probabilidade pode depender da história do sistema.

Geralmente, a representação do ambiente é bastante direta, consistindo apenas do conjunto de padrões e suas respectivas probabilidades (as quais são fixas).

Além desses oitos elementos, definiremos, também, o padrão de entrada sendo apresentado à rede. Chamaremos por S_k o k-ésimo padrão, o qual estabelece a ativação $a_i(0)$ nas unidades de entrada i. Muitas vezes, estaremos interessados nas saídas da camada de entrada, ou seja, no vetor $\mathbf{o}(0) = (o_1(0), \ldots, o_n(0))$, onde n é o número de unidades de entrada.

I.3 O Nosso Caminho

A área de aprendizado em redes neuronais é muito vasta. Há inúmeras concepções de como fazer uma rede neuronal aprender; umas mais biológicas, outras mais matemáticas, etc. A diversidade de abordagens é extremamente enriquecedora, mas ao mesmo tempo dificulta bastante o estudo deste campo.

Torna-se necessário, então, restringir, ou delimitar, o objeto de estudo. Uma das principais formas de aprendizado em redes neuronais é a não-supervisionada. Mesmo esta forma particular é por demais vasta para uma análise pormenorizada. Decidimos, portanto, estudar uma subclasse dos algoritmos não-supervisionados, a qual designaremos por algoritmos de *categorização*, examinando detalhadamente alguns representantes importantes desta subclasse. Estes algoritmos constituem um dos campos da área de aprendizado em redes neuronais que mais tem crescido nos últimos anos, apresentando avanços consideráveis.

Para não parecermos muito exagerados, ou muito limitados, quanto à delimitação da tese, citamos uma passagem de ABBOTT (1990, p. 105) ao rever o campo de aprendizado em redes neuronais: "Any reviewer hoping to cover this field in a reasonable amount of time and space must do so with a severely restricted viewpoint."

Pressupomos que o leitor possui uma razoável familiaridade com redes neuronais. Portanto, foi incluída neste primeiro capítulo apenas uma brevíssima introdução a estas redes. A principal função desta introdução é padronizar os termos a serem utilizados ao longo da tese.

No capítulo 2 introduzimos os principais conceitos relacionados ao aprendizado em redes e apresentamos algumas possíveis classificações deste tipo de aprendizado.

O capítulo 3 apresenta alguns conceitos úteis para o estudo do aprendizado em redes neuronais: funções discriminantes e algoritmos de conglomeração ("clustering").

O capítulo 4 é uma rápida discussão sobre o aprendizado supervisionado. Como grande parte do sucesso recente de redes neuronais se deve a um algoritmo supervisionado particular — "backpropagation" — achou-se necessário incluir este tipo de aprendizado nesta tese.

Os capítulos 5 a 8 correspondem ao núcleo desta tese. Neles são discutidos três algoritmos de aprendizado bastante importantes: *Aprendizado competitivo* (capítulo 5), *Auto-organização de mapas de características* (capítulo 6), e *aprendizado em sistemas ART* (capítulo 8). No capítulo 7 discutimos algumas questões acerca da instabilidade do aprendizado competitivo, servindo também, para introduzir alguns conceitos empregados nos sistemas ART. O conceito de estabilidade/instabilidade é extremamente importante para o estudo do aprendizado em redes neuronais e vem, recentemente, recebendo bastante atenção.

Mais do que apenas algoritmos de aprendizado importantes, a auto-organização de mapas de caraterísticas e os sistemas ART são considerados, atualmente, modelos de redes neuronais extremamente importantes. Os trabalhos relaciondos a estes modelos têm se multiplicado rapidamente.

O capítulo 9 é dedicado ao teste e avaliação dos três algoritmos não-supervisionados estudados.

Por fim, o capítulo 10 conclui a tese traçando algumas considerações finais sobre o aprendizado em redes neuronais.

Textos introdutórios acerca do aprendizado em redes neuronais são (HINTON, 1989), (LIPPMANN, 1987), (MATHEUS e HOHENSEE, 1987) e (SIMPSON, 1990).

Capítulo II

Aprendizado em redes neuronais

II.1 Um Pouco de História

A psicologia do aprendizado acumulou ao longo deste século uma enorme base de dados com ricas informações sobre várias formas de aprendizado (e.g., condicionamento clássico). Ao mesmo tempo, entretanto, a neurobiologia do aprendizado evoluiu lentamente até meados da década de 70 — a partir desta época o crescimento nesta área acelerou-se consideravelmente. Neste ínterim, poucos pontos de contato foram estabelecidos entre a psicologia e a neurobiologia do aprendizado.

Na tentativa de reduzir a distância entre estas duas disciplinas, Hebb foi um dos pesquisadores que mais se destacou na elaboração de teorias neuronais correspondentes a paradigmas de modificação comportamental. Seu importante princípio de que a alteração da eficiência sináptica é a base do aprendizado e da memória merece ser repetido aqui (HEBB, 1949, p. 50): "when an axon of a cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

Apesar deste princípio ser bastante geral (e vago), seu sucesso particular na explicação do condicionamento clássico (ver SUTTON e BARTO, 1981) foi um dos motivos para tê-lo feito tão influente.

A Figura (II.1) ilustra os efeitos do aprendizado nas conexões, ou

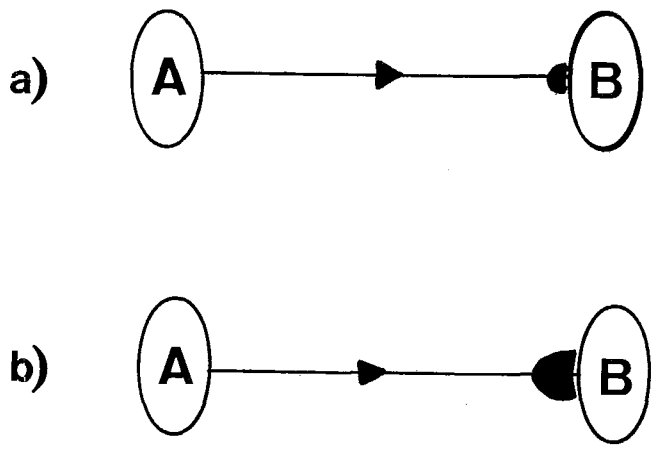

Figura II.1: a) Conexão antes. b) Conexão depois (reforçada)

sinapses, entre neurônios segundo o princípio de Hebb. A conjunção temporal das ativações dos neurônios A e B reforça a conexão entre eles. Esta alteração é, supostamente, o substrato neuronal de alguma alteração comportamental.

Segundo Hebb, uma determinada conexão só é alterada se as células pré- e pós-sinápticas estão excitadas (ativas). A ativação da célula pós-sináptica é tipicamente causada pela estimulação excitatória de vários outros neurônios pré-sinápticos, e assim, todas as células pré-sinápticas que estiverem contribuindo para esta ativação terão as suas conexões reforçadas. A coincidência temporal importante do princípio de Hebb é, na verdade, entre a ativação forte de certos neurônios pré-sinápticos. Aqueles cuja ativação forte coincide com a ativação de muitos outros têm a sua subseqüente influência aumentada.

É importante destacar que a concepção de alteração sináptica como base para o aprendizado é bem mais antiga do que o postulado de Hebb — Hebb é apenas o seu defensor (antigo) mais conhecido. Por exemplo, Freud no final do século passado defendeu tal idéia. De fato, esta concepção é anterior até ao consenso de que a sinapse existe.

O postulado de Hebb, e suas inúmeras variações, foi utilizado em muitos modelos matemáticos de aprendizado comportamental e mais recentemente

nos modelos de redes neuronais. Entretanto, foi apenas em 1986 que se obteve evidências inequívocas da existência de alterações Hebbianas *associadas* ao aprendizado em uma região do sistema nervoso conhecida como hipocampo (BROWN et al., 1988). Mesmo assim, como um dos grupos que reportou a existência de sinapses Hebbianas em 1986 afirma, as evidências *ligando* as alterações sinápticas ao aprendizado não são ainda totalmente conclusivas (BROWN et al., 1988).

A grande maioria dos pesquisadores de redes neuronais assume que a alteração das conexões da rede é o mecanismo básico para fazer uma rede aprender, e estudam as características e potencialidades de mecanismos particulares.

II.2 Formulação Matemática do Princípio de Hebb

Apesar de Hebb não ter formalizado o seu postulado, a seguinte equação matemática tem sido amplamente utilizada para descrevê-lo:

$$w_{ij}(t+1) = w_{ij}(t) + \eta a_i(t) a_j(t), \tag{II.1}$$

onde η é uma constante positiva denominada *taxa de aprendizado*. Muitas variações da regra de Hebb — como é chamada a equação (II.1) – foram estudadas (ver SIMPSON, 1990), sendo comum empregar-se o valor das saídas das unidades como em

$$w_{ij}(t+1) = w_{ij}(t) + \eta o_i(t) o_j(t).$$

Observe que o postulado de Hebb, e a equação (II.1) no caso das ativações serem positivas, só permite incrementos nos pesos. Devido a motivações biológicas — e também práticas — é comum alterar-se a regra de Hebb permitindo-se a incorporação de um termo de decaimento, ou "esquecimento". Estas regras com decaimento, ditas não-Hebbianas ou anti-Hebbianas, possuem muitas formulações semelhantes. Uma regra típica é dada pela equação diferencial

$$\frac{dw_{ij}}{dt} = -Aw_{ij} + a_i a_j, \tag{II.2}$$

onde A é uma constante de decaimento. Uma outra formulação obedece à equação

$$\frac{dw_{ij}}{dt} = a_j(a_i - Aw_{ij}). \tag{II.3}$$

Neste caso, os pesos só são alterados quando $a_j \neq 0$. Esta última equação é bastante importante. Quando a_j é binário só há alteração de pesos se a_j vale 1. Nesta situação (i.e., $a_j = 1$), (II.3) assume a seguinte forma discreta

$$w_{ij}(t+1) = w_{ij}(t) + a_i(t) - Aw_{ij}(t),$$

a qual é comumente substituída pela equação

$$w_{ij}(t+1) = w_{ij}(t) + \eta(a_i(t) - w_{ij}(t)). \tag{II.4}$$

Uma forma interessante de considerar a alteração de pesos em uma rede neuronal é expressa na equação abaixo:

$$w_{ij}(t+1) = w_{ij}(t) + \eta r_{ij}(t).$$

Ou seja, o peso de uma determinada ligação é alterado de acordo com um sinal de reforço r. Cada algoritmo de aprendizado particular estabelece como r deve ser calculado. Por exemplo, no aprendizado de Hebb o sinal de reforço é o produto das ativações pré- e pós-sinápticas.

Mencionamos no capítulo 1 que a regra de aprendizado é um dos elementos que especificam, ou determinam, o comportamento de um dado modelo de rede neuronal. Através de equações, tais como (II.1–4), estas regras estipulam como os pesos das conexões do sistema devem ser alterados ao longo do tempo.

II.3 A Importância do Aprendizado

Como já vimos, os pesos das ligações entre unidades em uma rede neuronal podem ser especificados através de uma matriz de interconexões W. O valor desta matriz é que determinará o comportamento específico de uma rede particular. Em última instância, pode-se dizer que são os pesos da rede que comandam o seu funcionamento.

Desta forma, se sabemos o que a rede deve fazer (e.g., associar padrões) é preciso determinar alguma matriz W que permita que a rede realmente o faça. Até mesmo em situações onde não podemos especificar exatamente a função

da rede, o importante é que se determine uma matriz W adequada. Os algoritmos de aprendizado tratados nesta tese são meios específicos de calcular W.

É importante ressaltar que estes algoritmos — ou outras formas particulares de obter W — são extremamente importantes, uma vez que para qualquer problema não trivial a matriz W não pode ser especificada diretamente. De fato, um dos principais motivos para o recente crescimento do interesse em redes neuronais se deve à elaboração de algoritmos de aprendizado bastante poderosos.

II.4 Algumas Classificações do Aprendizado em Redes Neuronais

É possível classificar os mecanismos de aprendizado em redes neuronais de diversas formas distintas. Apresentaremos nesta seção três destas classificações salientando as interdependências existentes entre elas.

II.4.1 Paradigmas de Aprendizado

RUMELHART e ZIPSER (1985) sugerem que uma classificação interessante é relacionada ao paradigma no qual o modelo deve funcionar. Estes, distinguem quatro paradigmas comumente encontrados:

1-*Auto-Associador.* Neste paradigma um conjunto de padrões é inicialmente "armazenado" no sistema para posterior recuperação através do uso de atributos particulares dos padrões armazenados. Auto-associadores são capazes de funcionar como uma *memória associativa.* Em outras palavras, ao apresentarmos apenas um pedaço de algum padrão armazenado, o sistema é capaz de recuperar o padrão completo ("pattern completion"). Além disso, eles são capazes de ao serem ativados com uma distorção de um padrão aprendido recuperarem o padrão em sua forma original ("pattern rectification"). Isto permite uma forma elementar de *generalização*, já que padrões semelhantes ao original são mapeados no padrão original. Auto-associadores podem ser treinados com diversas regras de aprendizado, tais como a regra de Hebb e a regra delta (ver capítulo 4). Maiores detalhes acerca

destas questões podem ser obtidos em (MCCLELLAND e RUMELHART, 1988, capítulo 6); ver também (MCCLELLAND e RUMELHART, 1985) e (PESSOA e REIS, 1990) para a utilização deste paradigma na representação de conhecimento.

A arquitetura de um auto-associador é muito simples: Há um único conjunto de unidades completamente conectado, ou seja, uma determinada unidade liga-se a todas as demais. O processamento na rede se dá através do cálculo repetido dos valores de ativação das unidades até que estes valores estabilizem.

2-*Associador de Padrões.* Um associador de padrões nada mais é do que um dispositivo que aprende associações entre padrões de entrada e de saída. Por exemplo, podemos ajustar os pesos das conexões de uma rede para que toda vez que um padrão de atividade f_1 surgir no grupo de unidades de entrada, um padrão associado, g_1, surja nas unidades de saída. Além disso, várias associações $f_i \rightarrow g_i$ podem ser capturadas no mesmo conjunto de ligações. Mais uma vez podemos dizer que o sistema é capaz de generalizar, já que padrões de entrada semelhantes são mapeados em padrões de saída parecidos.

A arquitetura típica de um associador de padrões consiste em um conjunto de camadas de unidades. Há três tipos de unidades: unidades de entrada, escondidas, e de saída. Dado um padrão de entrada, a atividade "flui" seqüencialmente — de camada em camada — da camada de entrada até a camada de saída. Para maiores informações veja (MCCLELLAND e RUMELHART, 1988, capítulo 4) e (ANDERSON, 1983).

3-*Classificador.* O funcionamento de um classificador pode ser enxergado como separando estímulos do ambiente (padrões de entrada) em classes distintas. O objetivo é que o sistema aprenda a classificar corretamente estímulos para que no futuro, quando um estímulo particular ou uma versão um pouco distorcida deste for apresentada, o sistema o classifique adequadamente. Este paradigma pode ser considerado como uma variação da associação de padrões; basta considerar que as unidades de saída do associador codificam as categorias de interesse. No entanto, como RUMELHART e ZIPSER (1985) enfatizam, seu objetivo, a classificação de padrões, é suficientemente distinto para considerá-lo como um paradigma

autônomo.

4-*Detetor de Regularidades.* O sistema é apresentado a um conjunto de estímulos é deve *descobrir* regularidades (características interessantes) na população de entrada. Em outras palavras, o sistema deve descobrir regularidades estatísticas dos padrões de entrada capazes de categorizá-los. Deve ser enfatizado que o sistema não possui, como no caso dos classificadores, um conjunto de classes a priori determinado para separar os padrões de entrada. Ao invés disso, ele próprio deve determinar categorias relevantes.

Uma outra classificação análoga possível sugerida por RUMELHART et al. (1986a) distingue apenas dois tipos de aprendizado: aprendizado associativo, e dectetor de regularidades. Neste caso, os três primeiros paradigmas mencionados acima foram agrupados sob o rótulo aprendizado associativo. No aprendizado associativo, normalmente os dois padrões (o de entrada e o de saída) a serem associados são fornecidos e a rede deve aprender a mapeá-los (no caso do auto-associador apenas um padrão é fornecido). No dectetor de regularidades o padrão de saída não é fornecido; ele deve ser descoberto.

II.4.2 Níveis de Supervisionamento

Uma outra maneira de classificar os mecanismos de aprendizado é relacionada aos níveis de supervisionamento empregados. Podemos distinguir, basicamente, três formas de aprendizado:

1-*Aprendizado Supervisionado.* Neste tipo de aprendizado, os *resultados* a serem obtidos pela rede são fornecidos integralmente. Por exemplo, em um associador de padrões os resultados, ou seja, os padrões de saída, são dados ao sistema para que a rede aprenda a associar os estímulos corretamente. O aprendizado supervisionado é chamado também de aprendizado com *professor* — o qual fornece os padrões de saída da rede.

2-*Aprendizado por Reforço.* Apenas um sinal externo é fornecido ao sistema, o qual pode ser entendido como uma medida relativa da adequação das

suas ações recentes. A idéia é que a rede muda seus pesos (aprende) de acordo com estes sinais. Por exemplo, se o sinal de reforço é negativo — correspondendo a uma punição — os pesos das ligações envolvidas devem ser decrementados, enquanto que um sinal positivo — correspondendo a uma recompensa — deve aumentar estes pesos. Este tipo de aprendizado é chamado também de aprendizado com um *crítico* — correspondendo ao agente externo (à rede) que julga as ações do sistema.

3-*Aprendizado Não-Supervisionado.* Nenhum tipo de sinal ou informação é fornecido ao sistema. O sistema aprende por si só propriedades interessantes dos estímulos de entrada.

Na prática, muitos algoritmos de aprendizado não podem ser classificados diretamente em um destes três grupos. É comum encontrarmos mecanismos particulares que misturam elementos dos tipos de aprendizado acima. Por exemplo, há algoritmos que misturam o aprendizado não-supervisionado e o supervisionado — e.g., possuem duas fases: uma não-supervisionada e outra supervisionada.

Pode-se imaginar que os aprendizados supervisionado e não-supervisionado são dois extremos quanto ao nível de informação (supervisionamento) externo fornecido à rede. O aprendizado por reforço seria, então, uma situação intermediária, podendo-se imaginar reforços mais ou menos informativos.

Os algoritmos supervisionados e não-supervisionados predominam na literatura de redes neuronais. Entretanto, nos últimos anos o aprendizado por reforço tem recebido mais atenção. Nesta tese não trataremos desta classe de algoritmos.

II.4.3 Regras de Correlação e Regras de Correção de Erros

É possível, ainda, classificarmos os algoritmos de aprendizado segundo a seguinte divisão:

1-*Regras de Correlação.* Como já vimos, o princípio de Hebb pode ser descrito formalmente segundo a seguite equação:

$$\Delta w_{ij} = \eta a_i a_j. \tag{II.5}$$

Este tipo de regra utiliza apenas informações locais para decidir como deve mudar os pesos envolvidos. Como MATHEUS e HOHENSEE (1987) indicam, a maioria das regras de aprendizado faz uso da correlação das ativações, mas o que faz uma determinada regra ser classificada como de correlação é que estas ativações são o único meio para alterar os pesos.

A regra de Hebb, ou outras regras de correlação (e.g., regras anti-Hebbianas), podem ser usadas tanto de uma forma supervisionada quanto não-supervisionada. (Na literatura há algumas pessoas que não fazem esta distinção; consideram a regra de Hebb sempre não-supervisionada.) No caso supervisionado, sabemos quais valores de ativação as unidades do sistema devem assumir. Assim, aplica-se a equação (II.5) para estes valores conhecidos.

Apesar de simples, a equação (II.5) supervisionada é capaz de implementar o armazenamento associativo de informações. Apresentações repetidas de pares de padrões (f_i, g_i) fazem com que a rede aprenda a gerar g_i quando apresentada à apenas f_i. (Mais precisamente, isto ocorre se os padrões f_i são ortogonais e possuem norma unitária.) Este fato gerou bastante interesse nos primeiros pesquisadores de redes neuronais. Para maiores detalhes ver (JORDAN, 1986) e (CARPENTER, 1989).

No caso não-supervisionado, não sabemos qual deve ser a saída do sistema. Após uma inicialização aleatória dos pesos, deixa-se simplesmente o sistema gerar uma saída dado que uma entrada foi fornecida. Agora, de posse deste valor, aplica-se a regra (II.5).

2-*Regras de Correção de Erros.* As regras de aprendizado baseadas na correção de erros são sempre utilizadas de uma forma supervisionada — entretanto, algumas regras supervisionadas, e.g., regra de Hebb supervisionada, não empregam correção de erros.

Nestes mecanismos de aprendizado, a idéia é permitir que a rede obtenha sua própria saída em resposta a um padrão qualquer, e comparar este resultado com o valor desejado. Caso a resposta da rede esteja correta não é necessário alterar os pesos, mas no caso de haver discrepância entre estes valores, a diferença

entre estes é usada para alterá-los.

Considerando t_j a saída desejada, ou correta, da unidade j e o_j a saída real (i.e., obtida pela rede) desta mesma unidade, uma regra de correção de erros típica alteraria os pesos w_{ij} de acordo com a difernça $t_j - o_j$. A equação exata da alteração de pesos vai depender do algoritmo particular considerado.

O objetivo básico das regras de correção de erros é reduzir ao máximo, possivelmente até anular, as diversas diferenças, ou erros, $t_j - o_j$ (para todos os padrões). Considerando E uma medida global do erro do sistema, onde E é tipicamente uma função dos valores $t_j - o_j$, o objetivo dos algoritmos de correção de erros é minimizar E. Alterando os pesos da rede segundo a equação

$$\Delta w_{ij} = -k\frac{\partial E}{\partial w_{ij}} \tag{II.6}$$

estaremos justamente minimizando a contribuição de cada peso para o erro global do sistema. Observe que a alteração dos pesos da equação (II.6) ocorre na direção de maior decrescimento da função E, devido ao sinal de sua derivada.

Como as regras de correção de erros tentam obter uma solução global — i.e., um mínimo de E — realizando pequenas alterações na direção de maior melhoria local, elas são, basicamente, o método gradiente descendente aplicado ao problema de aprendizado.

II.5 "Instars" e "Outstars"

Dois conceitos de extrema importância na área de aprendizado não-supervisionado são os módulos de aprendizado "instar" e "outstar". A Figura (II.2) mostra a geometria convergente do instar e divergente do outstar.

Em um instar, o vetor peso convergente $(w_{1j}, \ldots, w_{nj})$ aproxima-se do vetor $(o_1, \ldots, o_n)$ — correspondente ao padrão de ativação da camada de entrada — quando um padrão de entrada S_k é apresentado. Supondo que os vetores $\mathbf{w}$ e $\mathbf{o}$ são normalizados, isto faz com que a expressão $\sum_i w_{ij} o_i$ tenda ao seu valor máximo durante o aprendizado. Como sabemos, esta expressão é utilizada para calcular a

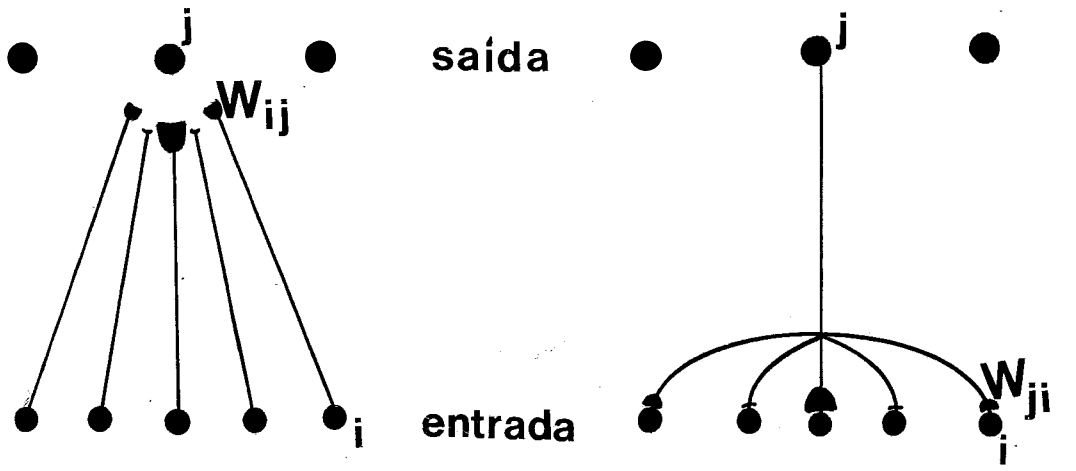

Figura II.2: a) instar. b) outstar.

entrada líquida na unidade j, net_j. Subseqüentes apresentações do padrão S_k irão gerar, então, atividades cada vez maiores na unidade j. Dizemos que j codifica o padrão S_k.

Uma regra de aprendizado para o instar é dada por

$$\frac{dw_{ij}}{dt} = \eta(o_i - w_{ij})o_j. \tag{II.7}$$

Suponha, por exemplo, que a unidade de saída j deve representar uma dada categoria. De acordo com (II.7), o vetor peso $(w_{1j}, \ldots, w_{nj})$ converge para o vetor $(o_1, \ldots, o_n)$ quando a unidade j está ativa, mas não é alterado quando uma outra representação de categoria está. Em outras palavras, a representação da categoria representada por j só é refinada quando ela está ativa. (Observe que a equação (II.4) é, basicamente, a equação do instar — binária — em uma forma discreta. Vemos, assim, que o conceito de instar possui ligação direta com a regra de Hebb, ou mais precisamente, com regras não-Hebbianas.)

Enquanto o instar permite uma categorização adaptativa, o outstar possibilita o aprendizado espacial de padrões. Suponha que o vetor peso divergente $(w_{j1}, \ldots, w_{jn})$ aproxima-se do padrão (espacial) de ativação $(o_1, \ldots, o_n)$ enquanto

o padrão de entrada S_k está presente. Ativações subseqüentes da unidade j transmitirão para a camada de entrada o sinal $o_j(w_{j1}, \ldots, w_{jn})$, o qual é diretamente proporcional ao padrão de ativação **o**.

Uma regra de aprendizado para o outstar é:

$$\frac{dw_{ji}}{dt} = \eta(o_i - w_{ji})o_j,$$

ou seja, quando a unidade de saída j está ativa o vetor $(w_{j1}, \ldots, w_{jn})$ converge para o padrão de atividade na entrada.

Instars e outstars foram introduzidos por Grossberg na década de 60 (ver CARPENTER, 1989) e são a base do aprendizado em muitos modelos de redes neuronais (ver, e.g., HECTH-NIELSEN, 1987). Além disso, como veremos nos capítulos a seguir, o conceito de instar é fundamental nos algoritmos não-supervisionados de categorização.

II.6 O Problema da Saturação Sináptica

Se considerarmos a regra de aprendizado da equação (II.1) vemos que os pesos podem crescer indefinidamente. De forma a controlar melhor o comportamento do sistema, é comum que as regras de aprendizado estabeleçam limites superiores (e inferiores, quando aplicável) para os valores dos pesos.

Mesmo supondo que os pesos possuem um limite máximo ainda há problemas. No caso da equação (II.1), por exemplo, seria possível que todos os pesos da rede alcançassem o valor máximo e permanecessem nele. Esta situação, conhecida como *saturação sináptica*, é altamente indesejável pois neste caso toda a informação na rede é perdida. Uma possível solução é fazer com que os pesos também decresçam em algumas circunstâncias. A saturação de alguns pesos da rede pode ser interessante, mas a saturação de todos claramente corromperá o seu funcionamento.

As regras não-Hebbianas (II.2–3) são uma maneira de contornar o problema da saturação sináptica. Outra forma bastante comum envolve o princípio

da *conservação da quantidade total de peso*, ou princípio da *normalização sináptica.* Este princípio estabelece que a soma de todos os pesos para uma determinada unidade j seja constante ao longo do aprendizado, ou seja, $\sum_i w_{ij} = c$, onde c é uma constante. Assim, após a determinação dos novos pesos realiza-se uma normalização:

$$w_{ij}(t+1) = w^*_{ij} / \sum_k w^*_{kj},$$

onde w^*_{ij} é o peso obtido pela regra de aprendizado sendo empregada. Claramente, a normalização faz com que se alguns pesos aumentam outros diminuam. Pode-se entender esta alteração sináptica como uma competição entre conexões por porções da soma total de pesos.

O princípio da normalização sináptica apresenta alguns problemas (ver SUTTON e BARTO, 1981), entre eles a plausibilidade biológica. Alguns pesquisadores defendem, portanto, a *normalização de ativação* como um meio de evitar a saturação sináptica. Se a ativação na rede for normalizada e os pesos tenderem a esta ativação (como no instar e no outstar), alguns pesos aumentarão enquanto que outros diminuirão.

Na prática, a saturação sináptica só é importante para algoritmos de aprendizado não-supervisionados (e, possivelmente, com reforço), já que os pesos podem, em certos casos, crescer indefinidamente. No caso supervisionado isto não ocorre pois as alterações nos pesos são uma função de $t_j - o_j$; quando $t_j - o_j = 0$ os pesos não mudam mais.

II.7 O Problema da Instabilidade do Aprendizado

Suponha que um algoritmo de aprendizado obteve uma matriz W capaz de resolver um problema específico. Se após isto a matriz for alterada o suficiente para que a rede não mais seja capaz de resolver o problema diz-se que o aprendizado é *instável.* Por exemplo, um categorizador de padrões passa a não mais ser capaz de categorizar corretamente os estímulos.

Apesar do aprendizado supervisionado ser teoricamente instável (ver

GROSSBERG, 1988), o problema da instabilidade não é tão relevante nesta forma de aprendizado. Isto porque tipicamente utilizam-se sistemas supervisionados em duas fases: fase de aprendizado e fase de uso. Ou seja, a rede é primeiramente treinada para ser depois utilizada. Obviamente que o problema da instabilidade é diluído já que os pesos são alterados até resolverem o problema e não mais mexidos.

Por outro lado, em sistemas que devem funcionar continuamente, como alguns sistemas de aprendizado por reforço e aprendizado não-supervisionado, o problema da instabilidade é bastante sério já que a rede pode "desaprender" fatos. (Muitos sistemas destes tipos também são utilizados em duas fases.)

Ao estudarmos e avaliarmos as formas de aprendizado não-supervisionado abordadas nesta tese, a questão da estabilidade/instabilidade será sempre uma questão importante. Em particular, estudaremos os problemas de instabilidade do aprendizado competitivo e como os sistemas ART os resolvem.

Capítulo III

Alguns Conceitos Úteis

A teoria de reconhecimento de padrões provê alguns conceitos bastante úteis para o estudo do aprendizado em redes neuronais. Neste capítulo discutiremos, brevemente, apenas os conceitos mais relevantes para esta tese. Uma discussão mais detalhada pode ser encontrada nos ótimos textos de TOU e GONZALEZ (1974) e NILSSON (1965).

III.1 Funções Discriminantes

A tarefa de um sistema classificador de padrões é a de, dado um padrão, determinar a *categoria*, ou classe, deste. Observe que este é exatamente o comportamento de uma rede neuronal classificadora de padrões. Mais precisamente, como um padrão pode ser representado como um ponto em um espaço d-dimensional Euclidiano (E^d), um classificador de padrões é, então, um dispositivo que mapeia pontos do espaço de atributos no espaço de categorias.

Para cada categoria i denotemos por $\mathbf{R}_i$ o conjunto de pontos em E^d que são mapeados no número i. A Figura (III.1) mostra os conjuntos de pontos $\mathbf{R}_1$, $\mathbf{R}_2$, e $\mathbf{R}_3$ em um exemplo particular. Observe que cada conjunto de pontos é separado dos demais por superfícies (neste caso curvas) chamadas *superfícies de decisão*. Em geral, as superfícies de decisão dividem E^d em um conjunto de regiões chamadas *regiões de decisão*. Neste contexto, um classificador de padrões deve ser capaz de dividir o espaço de atributos em regiões de decisão adequadas para o

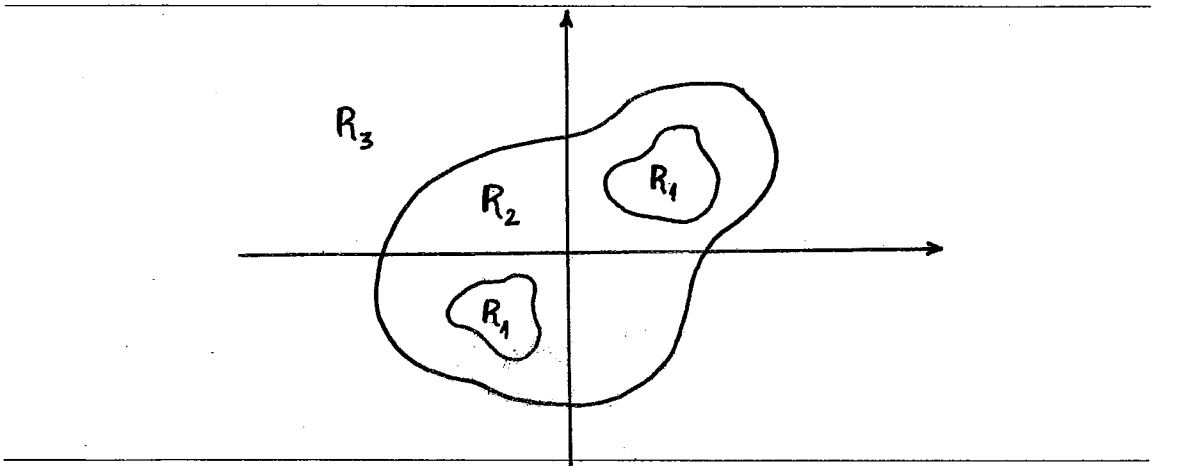

Figura III.1: Superfícies e Regiões de decisão

problema sendo tratado. É importante, então, determinar e especificar estas regiões.

As superfícies de decisão associadas a um classificador de padrões qualquer podem ser definidas por um conjunto de funções. Estas funções, denotadas por $g_i(\mathbf{x})$, onde $\mathbf{x}$ é um vetor padrão, devem ser escolhidas de tal forma que para todo $\mathbf{x}$ em $\mathbf{R}_i$

$$g_i(\mathbf{x}) > g_j(\mathbf{x}), \quad j \neq i.$$

Estas funções são ditas *funções discriminantes* (ou *funções de decisão*). A definição das funções $g(\mathbf{x})$ garante que se um determinado padrão $\mathbf{x}$ pertence à categoria i então $g_i(\mathbf{x})$ fornecerá o valor máximo, permitindo a classificação inequívoca de $\mathbf{x}$. Assumindo-se que as funções discriminantes são contínuas ao longo das superfícies de decisão, então

$$g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0 \tag{III.1}$$

determina a superfície de decisão que separa $\mathbf{R}_i$ e $\mathbf{R}_j$.

Segue-se naturalmente que a especificação das funções discriminantes é central na elaboração de classificadores de padrões, sejam eles neuronais ou não. Em particular, métodos de descoberta de funções discriminantes através do *treino*, ou *aprendizado*, de um dispositivo classificador de padrões são bastante importantes.

Em muitos tipos de aplicação não é necessário considerar todos os tipos possíveis de funções discriminantes. É possível considerar *famílias* particulares. Através do uso de parâmetros pode-se determinar membros particulares de

uma família mais geral. É possível tornar explícita a dependência de uma função discriminante com relação aos seus parâmetros escrevendo-a da forma

$$g(\mathbf{x}) = g(\mathbf{x}; w_1, w_2, \ldots, w_m),$$

onde os w_i são parâmetros reais – os parâmetros w_i são, na literatura de reconhecimento de padrões, comumente chamados de pesos. O conjunto de funções que pode ser obtido variando-se os valores dos parâmetros é dito uma família de funções.

O treino de um dispositivo classificador de padrões, restrito a uma família de funções discriminantes pode, então, ser obtido através do ajuste dos valores dos parâmetros.

Uma família de funções muito importante é a de *discriminantes lineares*, cujas funções são da forma

$$g(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + w_{d+1}, \qquad \text{(III.2)}$$

onde $(x_1, x_2, \ldots, x_d) = \mathbf{x}$. A superfície de decisão que separa $\mathbf{R}_i$ e $\mathbf{R}_j$ — ver equação (III.1) — no caso das funções discriminantes lineares, é dada por

$$(w_{i1} - w_{j1})x_1 + (w_{i2} - w_{j2})x_2 + \cdots + (w_{id} - w_{jd})x_d + (w_{i,d+1} - w_{j,d+1}) = 0. \qquad \text{(III.3)}$$

Observe que quando d é igual a 2 a equação (III.3) define uma reta; para $d = 3$ define um plano e $d > 3$ um hiperplano.

Dizemos que os conjuntos de categorias $\mathbf{R}_1, \mathbf{R}_2, \ldots \mathbf{R}_n$, são *linearmente separáveis* se e somente se há um conjunto de funções discriminantes lineares g_1, $g_2 \ldots$, g_n para este problema. Na situação onde há apenas as categorias 1 e 2, os conjuntos $\mathbf{R}_1$ e $\mathbf{R}_2$ são linearmente separáveis se há uma reta (no caso mais geral um hiperplano) separando estes conjuntos; ver a Figura (III.2). Quando há várias categorias é preciso, no pior caso, que haja um hiperplano separando cada par de regiões envolvidas.

O segundo membro de (III.2) possui forma quase igual à expressão típica de *net* – pode-se, obviamente, escrever (III.2) da mesma forma que *net*; basta utilizar um vetor aumentado $\mathbf{x}^* = (x_1, x_2, \ldots, x_d, 1)$. Desta forma, numa rede onde

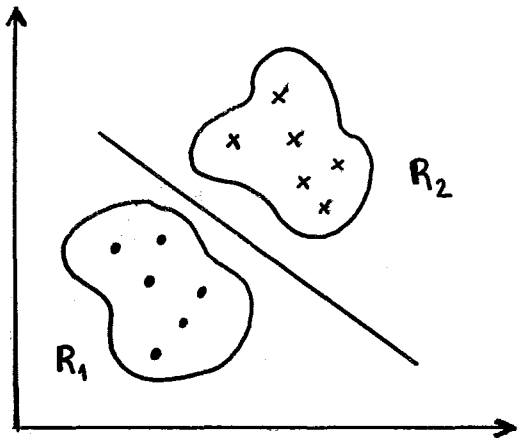

Figura III.2: Duas categorias linearmente separáveis

$a_i(t+1) = net_i(t)$ podemos dizer que a regra de ativação implementa, basicamente, uma função discriminante linear. Além disso, o aprendizado de uma rede neuronal deste tipo pode ser visto como o ajuste de uma função discriminante linear.

Em situações mais gerais as superfícies de decisão não são lineares, podendo se tornar bastante complexas. Uma maneira conveniente de generalizar o conceito de funções discriminantes lineares é considerar funções da forma

$$g(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_d f_d(\mathbf{x}) + w_{d+1}.$$

A equação acima representa uma variedade enorme de funções dependendo da escolha das funções f_i. Apesar de sua potencial complexidade é possível tratar a equação acima como se fosse linear (TOU e GONZALEZ, 1974). Basta supor que há um meio de calcular os diversos $f_i(\mathbf{x})$ e usar estes valores como os termos lineares da equação (III.2). (Basicamente, é por esta razão que problemas que não podem ser resolvidos por redes neuronais de duas camadas podem ser resolvidos por redes de múltiplas camadas.)

As idéias discutidas nesta seção são particularmente relevantes para os algoritmos de aprendizado supervisionados, já que podemos enxergar o comportamento destes como métodos para obter funções discriminantes – por vezes complexas, i.e., não-lineares.

III.2 Algoritmos de Conglomeração

Um outro conceito de reconhecimento de padrões – ou mais precisamente, de estatística – é o de algoritmos de conglomeração ("clustering"). Veremos ao longo da tese que este conceito é bastante útil no estudo dos algoritmos de categorização.

Dizemos que padrões formam conglomerados ("clusters") se há regiões no espaço de atributos com padrões bastante similares e ao mesmo tempo freqüentes. Uma maneira de formular o problema de conglomeração é tomar $\mathbf{X} = \{\mathbf{x}_i \mid i = 1, \ldots, n\}$ como um conjunto finito que deve ser particionado em subconjuntos disjuntos $\mathbf{X}_j$ de tal forma que, com respeito a essa divisão, alguma função descrevendo as distâncias entre elementos assume um valor extremo. Esta função deve descrever a qualidade dos conglomerados no sentido de que as distâncias entre todos $\mathbf{x}_i$ de um mesmo $\mathbf{X}_j$ são as menores possíveis enquanto que as distâncias entre elementos de $\mathbf{X}_j$ diferentes são grandes. No caso mais simples emprega-se a distância Euclidiana como uma medida de distância entre elementos vetoriais.

No parágrafo acima, introduzimos o conceito de conglomeração citando a distância Euclidiana entre elementos como forma de particionar os padrões de entrada. Para identificar os conglomerados, um algoritmo de conglomeração emprega, na realidade, uma *medida de similaridade* entre padrões, a qual é, no caso mais comum, simplesmente a distância Euclidiana. Entretanto, esta medida pode variar dependendo do método particular. Diversas medidas de similaridade métricas são possíveis. Algumas medidas não métricas são também comumente usadas. Por exemplo, a medida

$$s(\mathbf{x}, \mathbf{z}) = \frac{\mathbf{x}\mathbf{z}}{\|\mathbf{x}\|\|\mathbf{z}\|} = \cos(\mathbf{x}, \mathbf{z})$$

é útil quando os conglomerados se encontram ao longo de eixos. No caso de nos restringirmos a padrões binários ela possui uma interessante interpretação não geométrica: $s(\mathbf{x}, \mathbf{z})$ fornece uma medida do número de atributos compartilhados por $\mathbf{x}$ e $\mathbf{z}$ (TOU e GONZALEZ, 1974).

Uma outra medida de similaridade bastante insteressante é a cor-

relação (não-normalizada) entre dois vetores $\mathbf{x}$ e $\mathbf{y}$ que é dada por

$$\sum_i x_i y_i.$$

Mais uma vez encontramos a expressão de *net*. A entrada líquida na unidade i, net_i, nada mais é do que um valor da similaridade entre o vetor de entrada e o vetor peso em questão.

Além do estabelecimento de uma medida de similaridade, o estabelecimento dos conglomerados requer ainda algum critério para particionar os padrões. Muitos algoritmos utilizam critérios experimentais – i.e., funcionam adequadamente na prática –, mas há também os que empregam como critério a minimização (ou maximização) de um determinado *critério de desempenho*. Como exemplo, podemos citar o seguinte índice (TOU e GONZALEZ, 1974):

$$J = \sum_{j=1}^{N_c} \sum_{\mathbf{x} \in \mathbf{S}_j} \|\mathbf{x} - \mathbf{m}_j\|^2,$$

onde N_c é o número de conglomerados, $\mathbf{S}_j$ é o conjunto de padrões pertencentes ao conglomerado j e

$$\mathbf{m}_j = \frac{1}{N_j} \sum_{\mathbf{x} \in \mathbf{S}_j} \mathbf{x}$$

é o vetor média do conjunto S_j; N_j é o número de elementos em $\mathbf{S}_j$. O índice J representa a soma total dos quadrados dos erros (diferenças) entre os exemplares de um conglomerado e a média destes. Desta forma, um método que empregue o índice (critério) acima deve obter uma categorização dos padrões $\mathbf{x}$ que minimize J.

Descreveremos, agora, um algoritmo de conglomeração bastante simples. Suponha que há um conjunto de padrões $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Faça o centro de conglomerado ou *protótipo*, $\mathbf{z}_1$, ser igual a $\mathbf{x}_1$ (na realidade pode-se escolher qualquer $\mathbf{x}_i$). Calcule a distância D_{21} de $\mathbf{x}_2$ a $\mathbf{z}_1$. Se esta distância exceder um valor T pré-estabelecido crie um novo protótipo $\mathbf{z}_2 = \mathbf{x}_2$. Caso contrário, associe $\mathbf{x}_2$ ao conglomerado de $\mathbf{z}_1$. Suponha que $D_{21} > T$, (i.e., $\mathbf{z}_2$ foi criado). No próximo passo, as distâncias D_{31} e D_{32} de $\mathbf{x}_3$ para $\mathbf{z}_1$ e $\mathbf{z}_2$ são calculadas. Se ambos D_{31} e D_{32} forem maiores que T, cria-se um novo protótipo $\mathbf{z}_3 = \mathbf{x}_3$. Caso contrário, $\mathbf{x}_3$ será associado ao conglomerado do protótipo mais próximo. Não é difícil observar que o algoritmo prossegue calculando a distância entre os novos padrões e todos os protótipos criados.

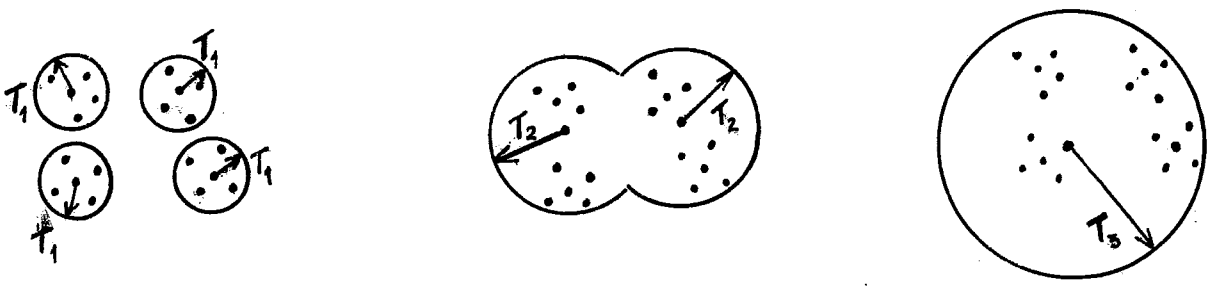

Figura III.3: Possíveis resultados do algoritmo

Se nenhuma distância for menor que T cria-se um novo protótipo e, caso contrário, o padrão é associado ao conglomerado do protótipo mais próximo.

Os resultados obtidos pelo algoritmo acima dependerão de uma série de fatores: a) o primeiro protótipo escolhido; b) a ordem em que os padrões são submetidos ao algoritmo; c) o valor de T; e d) as propriedades geométricas dos padrões. Estes efeitos estão ilustrados na Figura (III.3), onde diferentes conglomerações são obtidas para um mesmo grupo de padrões, através da variação de alguns parâmetros.

Este algoritmo é extremamente simples, mas ilustra alguns conceitos importantes de métodos de conglomeração. Algoritmos mais sofisticados utilizam formas mais elaboradas de criar e atualizar seus protótipos e medidas de similaridade mais complexas.

Cumpre ressaltar que o sucesso de um algoritmo de conglomeração, em uma certa classe de aplicações, dependerá da escolha criteriosa da medida de similaridade e do método de particionamento (formação de conglomerados) dos padrões. Critérios adequados em um tipo de aplicação podem não o ser em outro tipo.

Algoritmos de conglomeração podem ser vistos como métodos para o reconhecimento *não-supervisionado* de padrões (TOU e GONZALEZ, 1974), onde o termo não-supervisionado significa que não há informações sobre as classes dos padrões. Se considerarmos protótipos como uma forma de representação de classes, o processo de conglomeração pode ser visto como um meio de identificar, ou descobrir, as classes, ou categorias, em um dado conjunto de padrões.

Veremos posteriormente que os algoritmos de categorização em redes neuronais podem ser vistos como algoritmos de conglomeração.

Capítulo IV

Aprendizado Supervisionado

Neste capítulo discutiremos três algoritmos supervisionados: Aprendizado em Perceptrons, regra delta, e backpropagation. Como esta forma de aprendizado é a mais difundida e, também, a mais empregada em redes neuronais, dedicaremos alguma atenção a ela.

IV.1 Perceptrons

Por razões históricas, o primeiro sistema a ser estudado será o *Perceptron.* Originalmente proposto por Rosenblatt ao final da década de 50, o Perceptron obteve uma enorme repercussão na década seguinte. No entanto, alguns estudos (MINSKY e PAPERT, 1969) mostraram que estes modelos eram, apesar de interessantes, extremamente limitados (e.g., não eram capazes de computar a função lógica XOR). Mais precisamente, um Perceptron só é capaz de resolver problemas linearmente separáveis. Infelizmente, estas críticas enfraqueceram não só as pesquisas com modelos do tipo Perceptron mas também com outros tipos de redes neuronais. Para maiores informações acerca da história dos Perceptrons ver (PAPERT, 1988), (MINSKY e PAPERT, 1986) e (RUMELHART e ZIPSER, 1985).

Existem muitas variações do modelo do Perceptron (ver, e.g., ARBIB, 1987, capítulo 4), mas talvez a versão mais estudada seja a ilustrada na Figura (IV.1). O sistema possui três camadas de unidades. A primeira camada possui unidades de entrada — originalmente unidades R (de retina) —, a segunda uni-

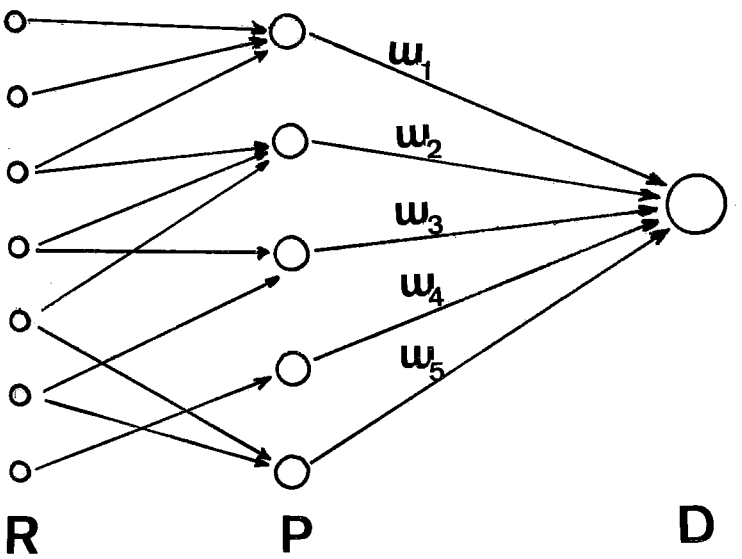

Figura IV.1: Perceptron

dades chamadas de predicados (P) e a última unidades de decisão (D). Na versão mais tradicional todas as unidades são binárias — no entanto, o sistema funciona também para entradas contínuas (LIPPMANN, 1987). As unidades do tipo P e D empregam função de saída com limiar. Os pesos das ligações $R \rightarrow P$ são fixos ao longo de todo o funcionamento do sistema, enquanto que as ligações $P \rightarrow D$ são variáveis; só existem ligações $R \rightarrow P$ e $P \rightarrow D$. Ou seja, há apenas *uma* camada de pesos variáveis. Assim, na prática (e no algoritmo abaixo) considera-se que o sistema possui apenas duas camadas de unidades. No caso mais geral existem diversas unidades de decisão.

Um Perceptron é, basicamente, um *classificador de padrões.* Quando há apenas uma unidade de decisão, o sistema deve separar os estímulos de entrada em duas categorias distintas. Dada uma entrada, o sistema responde com a saída 1 se o estímulo pertence a uma determinada classe e 0 caso contrário. O fato que gerou interesse nos Perceptrons foi a capacidade destes aprenderem um conjunto de pesos capaz de classificar a entrada em duas classes distinta ou, mais precisamente, que um Perceptron é um dispositivo capaz de aprender um discriminante linear.

O algoritmo proposto por Rosenblatt está descrito abaixo. O passo mais importante é, obviamente, o Passo 4; nele os pesos são ajustados de uma forma bastante simples. Como em todo algoritmo de correção de erros, os pesos são

alterados de acordo com a discrepância do resultado desejado e o efetivamente gerado pela rede — só ocorre alteração do peso no caso do sistema fazer a classificação incorreta, i.e., $t_d(t) \neq o_d(t)$. O peso é incrementado de η quando a rede gerou uma saída 0 e a saída desejada era 1, ou seja, os pesos não eram suficientemente fortes para gerar uma saída 1. Este incremento só é efetivado quando a unidade correspondente está ativa; isto garante que ela realmente contribuiu para o erro. A idéia de alterar os pesos apenas nos casos onde a unidade pré-sinaptica está ativa é bastante importante e é empregada na grande maioria dos algoritmos de aprendizado — sejam eles supervisionados ou não. Um raciocínio análogo, mas inverso, permite concluirmos que no caso complementar de discrepância o peso é decrementado de η. Algumas variações deste algoritmo modificam também o valor do limiar θ; ver (MCCLELLAND e RUMELHART, 1988, capítulo 5).

- **Algoritmo de Aprendizado do Perceptron**

  - **Passo1. Inicialização**

    Os pesos e o limiar são inicializados. $w_{id}(0)$ e θ recebem valores aleatórios pequenos diferentes de 0. O índice i varia nas unidades do tipo predicado e θ é o limiar da unidade de decisão, d.

  - **Passo 2. Entrada**

    Apresentação de entrada e saída desejada. Entrada: O padrão (binário) S_k é apresentado gerando o vetor $(o_1, \ldots, o_n)$ da camada de entrada; Saída desejada: $t_d(t)$

  - **Passo 3. Cálculo de saída efetiva**

    Saída efetiva o_d é dada por

    $$o_d = f(net_d - \theta),$$

    onde f é a função limiar, ou seja, o_d valerá 1 se a entrada na unidade exceder θ, 0 caso contrário.

– **Passo 4. Adaptação**

Os pesos w_{id} são alterados:

$$w_{id}(t+1) = w_{id}(t) + \eta\,(t_d(t) - o_d(t))\,o_i(t).$$

– **Passo 5. Loop**

Vá para o Passo 2. (Pode-se interromper o algoritmo quando os pesos não mais se alteram ou quando um contador exceder um certo valor.)

Este algoritmo, extremamente simples, é a base de muitos outros algoritmos supervisionados. Apesar de sua simplicidade ele é capaz de aprender classificações bastante interessantes, fato este que despertou um enorme interesse na década de 60. Pode-se dizer, até, que o algoritmo do Perceptron é bastante poderoso já que ele é capaz de aprender *tudo* o que um sistema como o da Figura (IV.1) pode implementar. Ou seja, se existe uma configuração de pesos que permite implementar uma determinada função, o algoritmo a descobrirá. Este fato é garantido formalmente pelo *Teorema da Convergência do Perceptron*; ver, e.g., (ARBIB, 1987). Em outras palavras, o sistema da Figura (IV.1) consegue implementar qualquer função linearmente separável, e o algoritmo de aprendizado do Perceptron é capaz de aprender este problema.

IV.2 A Regra Delta

Uma regra de aprendizado bastante conhecida é a chamada *regra delta* ou *regra de Widrow-Hoff* elaborada no início da década de 60. A arquitetura de rede para qual esta regra se aplica possui duas camadas de unidades, uma camada de entrada e uma de saída, e os pesos das ligações são variáveis. Este tipo de arquitetura tipicamente se aplica à associação de padrões. As unidades de saída da rede devem utilizar uma função de saída diferenciável (ver HINTON, 1989); no caso típico a função é linear.

A idéia da regra delta é similar à do Perceptron. Ou seja, comparar a saída obtida pela rede com o valor desejado (correto) desta. Considere o_{pj} a saída da unidade j, t_{pj} a saída desejada desta mesma unidade, e E_p o erro do sistema

quando da apresentação do padrão p. Então a função

$$E = \sum_p E_p = \sum_p \sum_j (t_{pj} - o_{pj})^2 \qquad \text{(IV.1)},$$

onde $E_p = \sum_j (t_{pj} - o_{pj})^2$, oferece uma medida de erro do funcionamento do sistema. Observe que se o sistema funciona perfeitamente, E será igual a zero e quanto mais erros a rede cometer maior será E. O objetivo da regra delta é minimizar E, ou seja, descobrir um conjunto de pesos que minimize esta função, e ela realiza isto usando, basicamente, o método do gradiente.

As mudanças nos pesos devem ser proporcionais a menos a derivada do erro, medido com respeito ao padrão corrente e em relação a cada peso. Ou seja, os pesos devem variar de acordo com a seguinte equação:

$$\Delta_p w_{ij} = -k \frac{\partial E_p}{w_{ij}}, \qquad \text{(IV.2)}$$

onde k é uma constante de proporcionalidade, e $\Delta_p w_{ij}$ é a variação de w_{ij} quando o padrão p é apresentado. Utilizando-se (IV.1) e (IV.2) pode-se obter

$$\Delta_p w_{ij} = \eta (t_{pj} - o_{pj}) o_{pi}.$$

Observe que esta regra é, basicamente, a regra do Perceptron.

Quando o mínimo de E é zero a rede resolve o problema perfeitamente, e se ele é maior do que zero a rede obtém o melhor desempenho possível (no sentido de mínimos quadrados). Além disso, para que o sistema de fato minimize E é preciso que esta função não possua mínimos locais, pois neste caso a minimização pode ficar "presa" em um desses mínimos. Em redes de duas camadas, com unidades de saída lineares, a superfície de erro é bem comportada e há apenas um mínimo (HINTON, 1989).

IV.3 Backpropagation

Provavelmente, o algoritmo de aprendizado mais difundido é o algoritmo de *back-propagation*, e a arquitetura em camadas ("feedfoward"), para a qual backpropagation se aplica, é sem dúvida o modelo de rede neuronal mais empregado (HECTH-

NIELSEN, 1989). Nada mais natural, então, do que dedicar algumas páginas a este mecanismo de aprendizado.

Aparentemente, backpropagation foi independentemente descoberto, em meados da década de 80, por Parker, le Cun, e por Rumelhart, Hinton, e Williams, do grupo PDP. Entretanto, descobriu-se recentemente que um trabalho de Werbos de 1974 e outro de Bryson e Ho de 1969 já apresentavam versões deste algoritmo. Ao que tudo indica, todas estas descobertas se deram independentemente.

A importância de backpropagation reside no fato de que redes (com unidades binárias com função limiar, i.e., do tipo Perceptron) de apenas uma camada de pesos só serem capazes de implementar funções linearmente separáveis. Isto, como se sabe, é uma grande limitação, já que funções simples (e.g., a função XOR) não podem ser obtidas. Redes com múltiplas camadas são capazes de implementar (ou mais precisamente, aproximar) funções multidimensionais bastante gerais, e o algoritmo de backpropagation permite que a rede aprenda estes mapeamentos. Vê-se, claramente, que esta forma de aprendizado é extremamente importante.

O algoritmo de backpropagation consiste numa generalização da regra delta (por esta razão alguns o chamam de *Regra Delta Generalizada*). A regra delta não é aplicável a redes de múltiplas camadas por duas razões principais. A primeira é que esta regra normalmente emprega unidades de saída lineares. No entanto, qualquer rede com apenas unidades lineares possui uma rede equivalente com apenas uma camada de pesos; ver, e.g., (RUMELHART et al., 1986a) para a demostração deste fato elementar. Desta forma, não faz sentido, a princípio, utilizar redes com várias camadas de unidades lineares. Um motivo mais importante provém do mecanismo de operação do algoritmo. Na regra delta, os pesos são ajustados de acordo com as diferenças entre os valores obtidos e os desejados. Numa rede de múltiplas camadas, os valores das unidades das camadas intermediárias não são conhecidos, já que no aprendizado supervisionado tradicional apenas os padrões de entrada e de saída são fornecidos. O problema é que, apesar de ser possível definir uma função de erro do sistema, não é, a princípio, claro como determinar como cada conexão contribui para o erro já que há várias conexões intermediárias (entre a entrada e a saída) e os erros podem se somar. Como não se sabe o que as unidades

escondidas devem fazer, não é imediato como computar o sinal de erro para estas unidades. Como, então, ajustar os pesos das ligações para minimizar uma medida de erro do sistema? Backpropagation fornece um mecanismo capaz de solucionar este problema.

Como na regra delta, a mudança de pesos deve ser da forma

$$\Delta_p w_{ij} = -k\frac{\partial E_p}{\partial w_{ij}}.$$

Realizando esta derivação obtém-se (ver RUMELHART et al., 1986b)

$$\Delta_p w_{ij} = \eta \delta_{pj} o_{pi} \tag{IV.3}$$

onde

$$\delta_{pj} = \begin{cases} (t_{pj} - o_{pj}) f_j'(net_{pj}) & \text{se a unidade } j \text{ for de saída} \\ f_j'(net_{pj}) \sum_k \delta_{pk} w_{jk} & \text{se a unidade } j \text{ for escondida} \end{cases},$$

onde f é a função de ativação e f' é a sua derivada. Pela equação (IV.3), vê-se que para unidades de saída usa-se a diferença entre os valores obtido e desejado — justamente como na regra delta. Já as unidades escondidas precisam utilizar os valores δ previamente calculados; o cálculo de δ para estas unidades ocorre de uma forma recorrente.

Podemos, agora, entender as duas fases do algoritmo de backpropagation. (i) Fase "forward": ativa-se a entrada e espera-se a ativação da saída. Calcula-se δ_{pj} para as unidades de saída. (ii) Fase "backward": a partir deste cálculo, propagam-se os sinais de erro para camadas anteriores, realizando-se as mudanças nos pesos. A natureza da mudança de pesos na fase "backward" dá o nome ao algoritmo.

Observe que a regra de aprendizado descrita em (IV.3) possui a mesma forma da equação de adaptação da regra delta: um determinado peso deve ser alterado de acordo com o produto de um sinal de erro, δ_{pj}, multiplicado pela ativação da unidade ligada, o_{pi}.

O algoritmo de backpropagation possui uma grande importância pois permite a utilização de redes neuronais em tarefas bastante complexas (e interessantes). (Backpropagation é capaz de gerar regiões de decisão extremamente complexas

— ver, e.g., algumas regiões formadas com estímulos de voz (LIPPMANN, 1988).) Entretanto, vários estudos mostraram que esta forma de aprendizado possui problemas sérios: (i) é muito lento (FAHLMAN, 1988); (ii) quando aplicado a redes maiores o algoritmo não se comporta tão bem (FANTY, 1988); e (iii) o algoritmo pode fornecer uma configuração de pesos que corresponde a um mínimo local da função de erro. Isto ocorre porque o algoritmo utiliza o método do gradiente e, a princípio, a superfície de erro possui uma forma qualquer. Na prática, utilizando-se representações suficientemente distribuídas o sistema se comporta bem (MCCLELLAND e RUMELHART, 1988).

Devido à sua enorme repercussão, backpropagation foi experimentado em um enorme número de aplicações, não sendo possível, assim, mencionar nem as mais importantes. Talvez a aplicação mais conhecida seja o sistema de SEJNOWSKI e ROSENBERG (1987) denominado NETtalk. Com o treino, NETtalk aprende a pronunciar palavras em inglês corretamente, sendo capaz de generalizar para palavras novas — i.e., não apresentadas ao longo do aprendizado. A entrada do sistema consiste do texto a ser falado e a saída (após pós-processamento) é uma voz sintetizada. Uma outra aplicação bastante interessante é o sistema de TESAURO e SEJNOWSKI (1989) para aprender a jogar gamão.

A quantidade de estudos teóricos também é muito grande e vários avanços têm sido obtidos; um dos mais importantes talvez seja a definição de versões mais rápidas. Recentemente, importantes trabalhos (e.g., HECTH-NIELSEN, 1989) têm investigado as capacidades computacionais de redes neuronais de três camadas, mostrando que estes sistemas são capazes de aproximar uma grande classe de funções. No entanto, é uma questão em aberto se backpropagation, ou qualquer outro método, consegue aprender os pesos necessários. Em outras palavras, teoricamente foi determinado que redes de três camadas de unidades podem resolver uma grande classe de problemas mas não se sabe se os algoritmos existentes são capazes de aprender a resolver tais problemas.

Capítulo V

Aprendizado Competitivo

V.1 Introdução

A partir deste capítulo estudaremos o aprendizado não-supervisionado em redes neuronais, em particular os algoritmos de categorização. Os primeiros pesquisadores a estudar os algoritmos de categorização foram Grossberg, von der Malsburg e Fukushima no início da década de 70. Posteriormente, muitos trabalhos foram produzidos trazendo grandes avanços para esta área.

Discutiremos, agora, uma forma de aprendizado em redes neuronais conhecida como *aprendizado competitivo* (RUMELHART e ZIPSER, 1985). Esta apresentação servirá para ilustrar as principais características dos algoritmos de categorização, dentre as quais podemos destacar a capacidade de unidades individuais de se especializarem e se comportarem como "detectores de características" ou classificadores de padrões sem o auxílio externo, ou seja, sem qualquer indicação de quais são as características ou categorias a serem descobertas. A formulação básica deste tipo de aprendizado, apesar de simples, é bastante interessante pois permite que uma rede descubra por si só características importantes, ou particulares, capazes de serem usadas para classificar um conjunto de padrões.

Apesar de muitos outros pesquisadores — e.g., GROSSBERG (1976a) e BIENENSTOCK et al. (1982) — terem estudado mecanismos de aprendizado muito semelhantes aos a serem discutidos aqui, concentraremos nossa exposição na formulação de RUMELHART e ZIPSER (1985).

Outros algoritmos de categorização podem ser vistos como variações, às vezes bastante sofisticadas, do aprendizado competitivo. A idéia básica desta forma de aprendizado é permitir que as unidades de uma rede inicialmente desestruturada *compitam* de alguma forma para obter o direito de responder a determinados estímulos. A desestruturação é no sentido de que os parâmetros da rede (e.g., pesos) são especificados aleatoriamente, permitindo que as unidades exibam respostas ligeiramente distintas quando estimuladas por um conjunto de padrões de entrada.

A arquitetura básica de um sistema de aprendizado competitivo pode ser vista na Figura (V.1). A rede é composta por duas camadas de unidades binárias, sendo que uma camada é de entrada e outra de saída. As unidades da camada de saída funcionam como unidades "winner-take-all" (wta), ou seja, apenas a unidade que recebe a maior excitação (i.e., maior *net*) possuirá ativação 1; as demais terão ativação 0. Dado um padrão à rede, a camada de saída categoriza-o através da ativação de apenas uma unidade. As unidades de saída representam, então, categorias particulares.

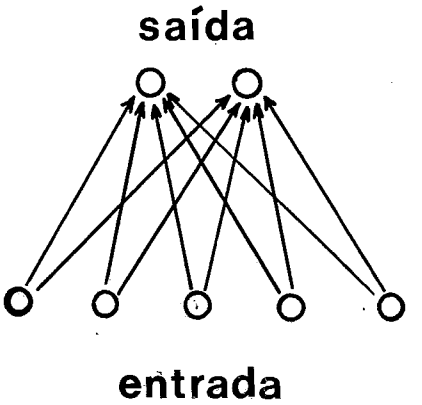

Figura V.1: Arquitetura básica de um sistema competitivo

A arquitetura de um sistema mais geral pode ser vista na Figura (V.2). Há n camadas de unidades, sendo que a camada 1 corresponde à entrada do sistema. As $n-1$ camadas restantes possuem um comportamento idêntico, sendo que cada uma possui conjuntos de unidades wta. O processamento do sistema

pode ser entendido como uma representação sucessiva do estímulo de entrada, onde cada camada detecta as regularidades do estímulo proveniente da camada anterior. Observe que a existência de conjuntos de unidades wta na formulação mais geral permite que a rede adote representações da entrada mais e mais refinadas nas camadas sucessivas. Na formulação básica, a camada de saída, ou de categorias, adota uma representação que não permite mais refinamento, e desta forma não faz sentido adicionar outras camadas. No texto a seguir, salvo menção explícita, estaremos nos referindo sempre à formulação básica do aprendizado competitivo.

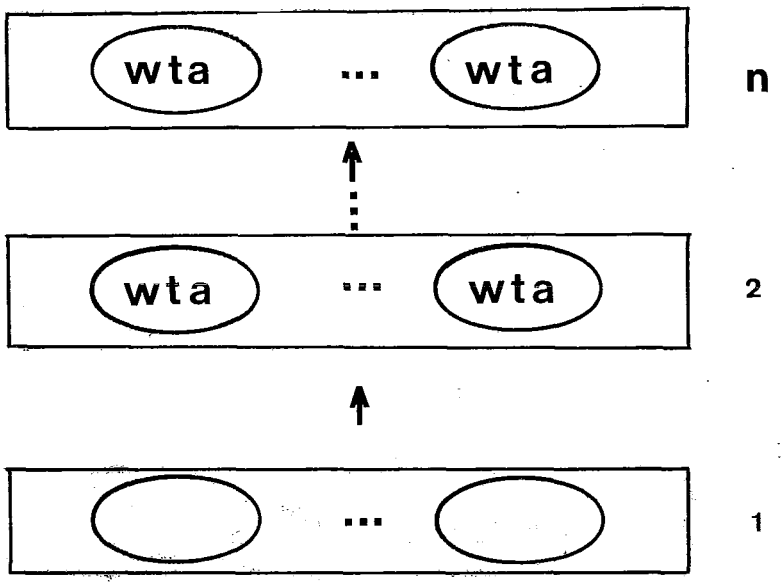

Figura V.2: Arquitetura mais geral

As unidades de saída da rede competem para aumentar os pesos de suas conexões e, assim, reforçar as suas chances de responder a certos padrões. A competição das unidades de saída consiste exatamente em determinar a unidade que recebe maior excitação em suas entradas. Apenas a unidade vencedora adquire o direito de alterar os pesos de suas conexões.

V.2 O Algoritmo de Aprendizado Competitivo

Como outros algoritmos não-supervisionados, o aprendizado competitivo precisa resolver o problema da saturação sináptica. A solução adotada no algoritmo de RUMELHART e ZIPSER (1985) a ser apresentado a seguir, envolve a normalização inicial dos pesos e a normalização da ativação. Ou seja, inicialmente, para cada unidade de saída j, $\sum_i w_{ij} = 1$, e ao longo do aprendizado considera-se a ativação de uma unidade de entrada i como $o_i / \sum_j o_j$.

Podemos, agora, descrever mais precisamente o algoritmo de aprendizado competitivo. Abaixo destacamos os principais passos.

- **Aprendizado Competitivo**

  - **Passo 1. Inicialização**

    Os pesos das ligações recebem valores aleatórios pequenos de forma que para cada unidade de saída j, $\sum_i w_{ij} = 1$.

  - **Passo 2. Entrada**

    Um padrão binário S_k é fornecido à rede.

  - **Passo 3. Competição**

    As unidades de saída competem e apenas uma vence. A unidade de saída vencedora j é tal que net_j é máximo. A unidade vencedora receberá ativação 1 e as demais ativação 0, i.e., $o_j = 1$ e $o_k = 0$, $k \neq j$.

  - **Passo 4. Adaptação**

    Alteração dos pesos das ligações da unidade vencedora. Para todas as unidades i ligadas à unidade vencedora j faça

    $$w_{ij}(t+1) = w_{ij}(t) + g\left(\frac{o_i}{n_k} - w_{ij}(t)\right)$$

    onde g é a constante de aprendizado, e n_k é o número de unidades ativas em S_k, i.e., $n_k = \sum_j o_j$.

  - **Passo 5. Loop**

    Vá para o Passo 2.

A equação de adaptação do Passo 4 é, basicamente, uma regra de Hebb com esquecimento discreta (como visto no capítulo 2) combinada com uma normalização da ativação de entrada. Ao dividir o_i por n_k, garante-se que as alterações nos pesos serão proporcionais à ativação relativa de cada unidade e garante-se, também, a não violação da condição do Passo 1. Se não levarmos em conta a normalização da ativação, temos que

$$\Delta w_{ij} = g(o_i - w_{ij})$$

se j é a unidade vencedora. Ao vencer, a unidade j altera seus pesos de forma a diminuir a diferença entre o_i e w_{ij}. Ou seja, w_{ij} tenta copiar o_i. Vemos, então, que o aprendizado competitivo comporta-se como um "instar": realiza uma categorização adaptativa.

Observe que a condição imposta no Passo 1 não pode ser violada no Passo 4. Desta forma, alguns pesos são reforçados enquanto outros são diminuídos. É fácil ver que a adaptação preserva a quantidade total de peso de uma determinada unidade. No Passo 4 temos que

$$\Delta w_{ij} = g\left(\frac{o_i}{n_k} - w_{ij}\right).$$

Somando-se todas as alterações de uma determinada unidade j obtemos

$$\sum_i \Delta w_{ij} = g\left(\sum_i \frac{o_i}{n_k} - \sum_i w_{ij}\right).$$

Por definição, na primeira vez que o Passo 4 for realizado $\sum_i w_{ij} = 1$, e por definição $\sum_i o_i/n_k = 1$, também. Ou seja, após o Passo 4 a variação total dos pesos da unidade j, será nula.

Comumente, a condição de parada do algoritmo é simplesmente determinada por um contador de iterações, i.e., após n ciclos (a determinação de n é empírica) de aprendizado interrompe-se o algoritmo. Assim, utiliza-se a rede em duas fases. Primeiro treina-se a rede para em seguida utilizá-la em situações reais (onde os pesos não mais são alterados).

V.3 Interpretação Geométrica

RUMELHART e ZIPSER (1985) apresentam um interpretação geométrica interessante do aprendizado competitivo. Como sabemos, os padrões de entrada podem ser considerados como vetores, mais precisamente vetores n-dimensionais, onde n é o número de unidades de entrada. Se os vetores possuem norma unitária e três componentes, podemos representá-los como pontos na superfície de uma esfera unitária (no caso mais geral seria uma hiperesfera n-dimensional). Desta forma, padrões de entrada similares estarão, na esfera, próximos uns dos outros, e padrões distintos separados. Analogamente, podemos considerar que há um vetor peso para cada unidade de saída com o mesmo número de componentes que os vetores padrão (há uma ligação de cada unidade de entrada para uma determinada unidade de saída). Como todas as unidades possuem a mesma quantidade de peso, e considerando o caso de vetores de três componentes, os vetores peso também corresponderão, aproximadamente, a pontos em uma esfera unitária. (Para que esta interpretação fosse exata, e não aproximada, seria necessário usar a restrição $\sum_i w_{ij}^2 = 1$ e não a restrição adotada. Observe que esta nova restrição corresponderia ao fato de que todos os vetores peso possuem norma unitária.)

Observando a Figura (V.3) podemos entender o comportamento do aprendizado competitivo em termos geométricos. Quando um padrão é apresentado, a unidade que recebe maior excitação, e conseqüentemente vence a competição, é aquela cujo vetor peso está mais próximo ao vetor padrão. (A unidade j que recebe maior estímulo é aquela para qual a expressão $\sum_i w_{ij} o_i = \mathbf{w}_j \mathbf{o}$ é máxima. Mas como por defição, $||\mathbf{w}_j|| = ||\mathbf{o}|| = 1$, esta expressão é equivalente a $\cos\theta$; θ é o ângulo entre $\mathbf{w}_j$ e $\mathbf{o}$. Como sabemos, $\cos\theta$ será maior para ângulos menores e, assim, para um determinado padrão de entrada, a unidade vencedora será aquela que possui o vetor peso "mais paralelo" ao vetor entrada.) Segundo a regra de aprendizado, ao vencer, o peso move-se em direção à localização do padrão na esfera. Em outras palavras, o algoritmo de aprendizado faz com que $\mathbf{w}_j$ se aproxime de $\mathbf{o}$.

Supondo que haja M agrupamentos de padrões e M unidades de categoria, após treino suficiente e supondo que os agrupamentos são suficientemente

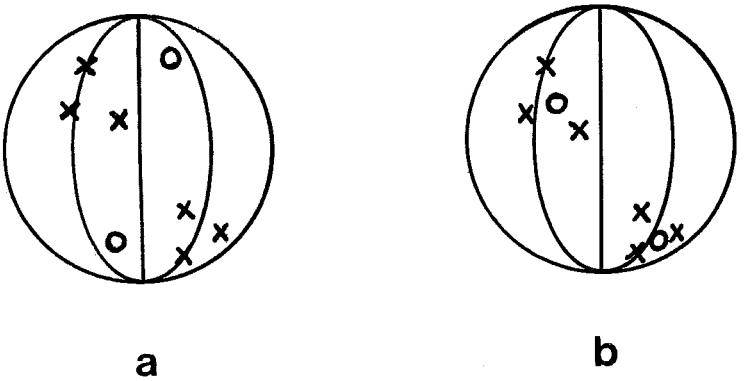

Figura V.3: Interpretação geométrica. a) Situação inicial. b) Posições dos pesos após aprendizado. x: vetor padrão; o: vetor peso

separados, cada vetor peso estará basicamente no centro de um grupo de padrões. Assim, podemos dizer que as unidades passaram a *detectar* os agrupamentos aos quais pertencem os padrões de entrada.

Podemos enxergar o algoritmo de aprendizado competitivo como o refinamento da representação dos padrões de entrada. A representação da entrada é realizada por uma única unidade de categoria e consiste simplesmente do vetor $\mathbf{w}$. Se as mudanças nos pesos forem suficientemente pequenas, algumas versões deste algoritmo minimizam uma medida da adequação da representação. As configurações de pesos obtidas pelo algoritmo são mínimos da função

$$E = \sum_p E_p = \sum_p \sum_i (w_{ij} - o_{pi})^2,$$

onde p varia nos padrões apresentados à rede, $\mathbf{w}_j = (w_{1j}, \ldots, w_{nj})$ é o vetor peso que é alterado na apresentação do padrão p, e E_p é o erro do sistema quando da apresentação do padrão p (HINTON, 1989). Esta expressão está de acordo com a interpretação geométrica de que $\mathbf{w}_j$ tende a ficar no centro dos padrões para os quais a unidade j vence. Infelizmente, a regra adotada por Rumelhart e Zipser não pode ser interpretada como minimizando uma função de erro tão simples. Outras variações de aprendizado competitivo podem ser interpretadas como minimizando funções que podem ser aplicadas a uma série de problemas de otimização; e.g., ver (DURBIN e WILLSHAW, 1987).

V.4 Algumas Características do Aprendizado Competitivo

RUMELHART e ZIPSER (1985) destacam algumas propriedades interessantes do aprendizado competitivo:

1- Como uma regra geral, um sistema com M unidades de categoria classifica o conjunto de padrões em M categorias. Por exemplo, um sistema contendo duas unidades de categoria se comporta como um detector de características binário: uma unidade responde quando uma característica está presente no estímulo, caso contrário, a outra unidade responde. No entanto, isto ocorrerá ou não dependendo da estrutura dos estímulos de entrada.

2- Se há estrutura nos estímulos de entrada, as unidades dividirão estes padrões de acordo com características relevantes. Basicamente, isto significa que o sistema descobrirá conglomerados, se eles existirem.

3- Os conglomerados particulares obtidos pelas unidades de saída dependerão dos valores iniciais dos pesos e da seqüência de estímulos apresentada.

Estas três propriedades, juntamente com a interpretação geométrica da seção anterior, deixam evidentes as ligações entre o aprendizado competitivo e os algoritmos de conglomeração. O aprendizado competitivo pode ser enxergado como um algoritmo de conglomeração onde os vetores peso ($\mathbf{w}$) são os protótipos do sistema e o produto escalar (ver Passo 3) é utilizado como uma forma de estabelecer a similaridade entre vetores. O Passo 3 determina qual o protótipo mais adequado para um novo estímulo de entrada, enquanto que o Passo 4 refina os protótipos escolhidos no Passo 3. Como vimos, este passo faz $\mathbf{w}_j$ se aproximar de $\mathbf{o}$.

A uma primeira aproximação, o aprendizado competitivo estabelece conglomerados que minimizam a distância entre elementos de um mesmo conglomerado, maximiza as distâncias entre elementos de conglomerados diferentes, e tenta equilibrar o número de padrões em cada conglomerado (MCCLELLAND e RUMELHART, 1988). Em geral, o algoritmo obtém soluções de compromisso quanto às tendências mencionadas acima. Em outras palavras, é como se houvesse um critério

de desempenho (ver capítulo 3) baseado nestas tendências. RUMELHART e ZIPSER (1985) apresentam uma aplicação onde é possível estabelecer precisamente o critério de desempenho minimizado pelo sistema (em geral não é simples estabelecer tal critério).

Antes de prosseguirmos é necessário alertarmos para uma possível inconsistência entre o aprendizado competitivo, ou outros algoritmos de categorização, e os algoritmos de conglomeração. É comum encontrar na literatura de redes neuronais autores afirmando que o aprendizado competitivo classifica os padrões de entrada em um conjunto de classes, ou categorias. Entretanto, algoritmos tradicionais de conglomeração não são métodos de classificação, os quais determinam a classe, ou categoria, de um padrão dado. Podemos dizer apenas que estes algoritmos fornecem um meio de *descobrir* possíveis classes em um conjunto de padrões. No entanto, devido ao tipo de uso de redes neuronais tornou-se comum afirmar que algoritmos que implementam métodos de conglomeração classificam padrões.

4- Quando a rede não está mais aprendendo, uma unidade j *codifica* (GROSSBERG, 1976a) todos os padrões S_k para os quais

$$net_j > net_i, \quad j \neq i.$$

A unidade j pode ser considerada um *detector de características* no sentido de que todos os padrões no conjunto $\mathbf{P}_j$ são codificados pela unidade j:

$$\mathbf{P}_j = \{S_k \mid \mathbf{o}_k \mathbf{w}_j > \mathbf{o}_k \mathbf{w}_l;\ l \neq j\}, \tag{V.1}$$

onde k torna explíicita a dependência entre $\mathbf{o}_k$ e S_k. Observe que se $S_k \in \mathbf{P}_j$ vários outros padrões também estarão em $\mathbf{P}_j$ (em particular os padrões S_l tais que $\mathbf{o}_l = \alpha \mathbf{o}_k$, onde $\alpha > 0$). O conjunto $\mathbf{P}_j$ define a *caraterística* codificada pela unidade j (GROSSBERG, 1976a).

Segundo GROSSBERG (1976a), a regra de classificação definida por (V.1) possui uma interpretação geométrica informativa. Sabemos da álgebra linear que

$$net_j = \|\mathbf{o}_k\| \|\mathbf{w}_j\| \cos(\mathbf{o}_k, \mathbf{w}_j).$$

Em outras palavras, net_j é o comprimento da projeção do vetor $\mathbf{o}_k$ sobre o vetor $\mathbf{w}_j$ vezes o comprimento de $\mathbf{w}_j$. Se todos os vetores $\mathbf{o}$ e $\mathbf{w}$ possuem mesmo comprimento,

(V.1) classifica em $\mathbf{P}_j$ todos os padrões S_k cujos ângulos de $\mathbf{o}_k$ com $\mathbf{w}_j$ são menores (i.e., os cossenos são maiores) que qualquer outro ângulo entre $\mathbf{o}_k$ e $\mathbf{w}_i$, $i \neq j$. Em particular, padrões S_k com $\mathbf{o}_k$ paralelos a $\mathbf{w}_j$ são classificados em $\mathbf{P}_j$. Podemos constatar que a interpretação geométrica de Rumelhart e Zipser é bastante similar à análise anterior de Grossberg.

Além disso, segundo GROSSBERG (1976a), a escolha dos vetores $\mathbf{w}_j$ determina como os padrões S_k serão categorizados. Resumindo, podemos dizer que:

(i) O número de unidades de categoria determina o número máximo de classes $\mathbf{P}_j$;

(ii) A escolha dos vetores de classificação $\mathbf{w}_j$ determina quão diferentes estas classes serão — e.g., a escolha de todos os $\mathbf{w}_j$ iguais gerará uma classe que será redundantemente representada por todas as unidades de saída.

5- A *estabilidade* da classificação depende da estrutura dos estímulos de entrada. A idéia de estabilidade aqui é relativa ao fato de que quando uma determinada unidade de categoria passa a codificar um conjunto de estímulos, se a classificação é estável, esta unidade permanecerá codificando apenas estes estímulos. Em um sistema instável, uma unidade responde primeiro a um conjunto de estímulos, podendo mais tarde responder a outros. Por exemplo, em um sistema que classifica letras, uma unidade pode inicialmente responder toda vez que a letra A aparece e mais tarde responder à presença da letra B. Assim, um detector de letras A passa a detectar letras B. Esta instabilidade da classificação é, obviamente, diretamente relacionada com o problema da instabilidade do aprendizado discutida no capítulo 2.

Com o aprendizado competitivo, se os estímulos de entrada não estão suficientemente estruturados, as categorizações do sistema são instáveis, e a apresentação de cada estímulo poderá alterar essas categorizações. Ou seja, o sistema recodifica os estímulos; esta recodificação pode continuar indefinidamente. Por outro lado, no caso de estímulos de entrada suficientemente estruturados, o sistema será bastante estável (i.e., as mesmas unidades responderão aos mesmos estímulos).

O problema da instabilidade é diluído nas situações em que o aprendizado é interrompido e a rede passa a ser usada sem alteração de seus pesos. Entretanto, em muitas formas de aprendizado não-supervisionado a rede está continuamente se adaptando, o que exige uma solução mais efetiva para este problema.

No capítulo 9, ao avaliarmos os algoritmos de categorização teremos a oportunidade de observar melhor a instabilidade do aprendizado competitivo.

6- Quando um sistema está em um estado no qual, em média, os pesos não estão mudando (i.e., estados nos quais o acréscimo médio de peso em uma determinada conexão é igual ao decréscimo médio nesta mesma conexão) dizemos que o sistema alcançou um *estado de equilíbrio*. Neste caso, pode-se determinar uma série de informações interessantes: (i) w_{ij} torna-se proporcional à probabilidade de que a unidade de entrada i esteja ativada dado que a unidade j vença, ou seja,

$$w_{ij} \rightarrow \frac{1}{n} P\left(o_i = 1 \mid j \text{ vence}\right),$$

sendo $n_k = n$ para todo k. (ii) No equilíbrio uma unidade responde mais fortemente a padrões que possuem interseção com outros padrões aos quais ela também responde e fracamente a padrões distantes daqueles padrões. Além disso, o sistema sempre responde da mesma forma a um particular estímulo. (iii) É possível que os pesos sejam alterados de forma que o sistema saia do equilíbrio, podendo mover-se para um novo estado de equilíbrio. Seja v_{kj} a probabilidade da unidade j vencer quando o estímulo S_k é apresentado. Alguns estados de equilíbrio são mais estáveis do que outros no sentido de que os valores v_{kj} não sofrem alterações por longos períodos de tempo. Uma possível medida da estabilidade de um estado de equilíbrio é dada pela quantidade média pela qual a entrada para as unidades vencedoras é superior à entrada de todas as outras unidades (com relação a todos os padrões e todas as unidades de categoria do sistema):

$$T = \sum_k p_k \sum_{j,i} v_{kj}(net_{kj} - net_{ki}),$$

onde p_k é a probabilidade do estímulo S_k ser apresentado e net_{kj} é a entrada na unidade j na presença do estímulo S_k. Observe que quanto maiores as difereças $net_{kj} - net_{ki}$, menos tendência terá a rede em alterar as unidades vencedoras com

pequenas alterações nos pesos. Quanto maior T, mais estável será, estão o sistema. Para maiores detalhes ver (RUMELHART e ZIPSER, 1985).

V.5 Algumas variações do aprendizado competitivo

Em algumas situações é possível que uma determinada unidade nunca vença, e, assim, nunca altere os pesos de suas ligações. Como afirmam RUMELHART e ZIPSER (1985), isto *retira* a competição do aprendizado competitivo. (Esta situação pode ocorrer facilmente quando existem padrões de entrada esparsos; uma unidade pode ter a maior parte de seus pesos em ligações de unidades de entrada que nunca serão ativados, e desta forma, ela nunca consegue vencer.)

Geometricamente, esta situação ocorre, por exemplo, quando todos os padrões estão próximos na esfera, e um dos pesos, por acaso, está próximo ao agrupamento de padrões, enquanto que os outros encontram-se mais afastados; ver Figura (V.4). Claramente, os pesos afastados não vencerão nunca, e conseqüentemente, não se aproximarão dos padrões.

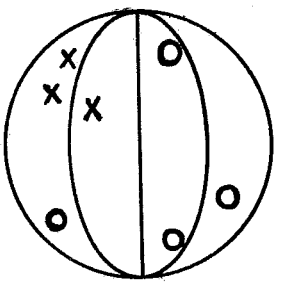

Figura V.4: Pesos afastados não vencerão

Uma modificação imediata para amenizar este problema, é permitir, também, que os vetores peso das unidades perdedoras se aproximem dos padrões. Entretanto, neste caso, a unidade vencedora aproxima-se dos padrões por passos maiores. Este procedimento faz com que as unidades perdedoras aproximem-se

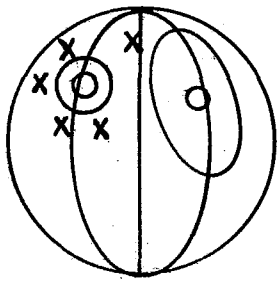

Figura V.5: Interpretação geométrica

vagarosamente da região de padrões e que, então, compitam em pé de igualdade com as outras unidades. A alteração dos pesos é dada por

$$\Delta w_{ij} = \begin{cases} g_l(o_i/n_k - w_{ij}) & \text{se a unidade } j \text{ perde no estímulo } k \\ g_w(o_i/n_k - w_{ij}) & \text{se a unidade } j \text{ vence no estímulo } k \end{cases},$$

onde g_l e g_w são as constantes de aprendizado para as unidades perdedoras e vencedoras, respectivamente, e $g_l \ll g_w$ (e.g., uma ordem de magnitude menor).

Uma outra variação é introduzir limiares variáveis nas unidades de saída. Assim, pode-se ajustar a sensibilidade das unidades permitindo que todas as unidades aprendam. O mecanismo se processa da seguinte forma. Quando uma unidade perde ela decrementa o seu limiar, e quando vence ela o aumenta. RUMELHART e ZIPSER (1985) sugerem que podemos entender este mecanismo em termos de uma interpretação geométrica onde os vetores peso possuem um círculo ao redor de suas extremidades na esfera; ver Figura (V.5). Uma unidade vence se a distância do padrão ao seu círculo é mínima. Toda vez que uma unidade perde ela aumenta o raio de seu círculo (diminui o seu limiar), e toda vez que ela vence ela o diminui (aumenta o seu limiar). Assim, o círculo em torno das unidades perdedoras se tornará grande o suficiente para elas estarem mais próximas de algum padrão do que as outras unidades.

Em termos de estabilidade estas alterações não modificam muito a situação geral (GROSSBERG, 1987). Dependendo da estrutura dos padrões de entrada o aprendizado poderá ser bastante instável.

A idéia de "punir" as unidades que vencem freqüentemente e beneficiar outras que encontram dificuldades para vencer, levou DESIENO (1988) a propor um mecanismo de *consciência* para o aprendizado competitivo. O termo consciência surge devido à introdução de um mecanismo que faz com que uma unidade que vence muito comece a "sentir-se culpada" e deixe de ganhar excessivamente.

Na versão do aprendizado competitivo com consciência a unidade vencedora, a qual adquire o direito de alterar seus pesos, não é mais obtida como no Passo 3 do algoritmo. Um termo de penalização baseado no número de vezes que a unidade venceu é introduzido e a unidade vencedora é a unidade j para qual a expressão

$$(\sum_i w_{ij} o_i) - p_j$$

é máxima, onde p_j é o termo de penalização da unidade j. Podemos considerar p_j como o limiar da unidade j, o que mostra que o mecanismo de consciência é muito similar à proposta de introduzir limiares nas unidades de saída.

V.6 Um Exemplo da Utilização do Aprendizado Competitivo

RUMELHART e ZIPSER (1985) descrevem uma aplicação de aprendizado competitivo envolvendo padrões correspondendo a determinadas letras. As unidades de entrada correspondem a *"pixels"* de uma matriz 7x14 (existem, então, 98 unidades de entrada), sendo possível "acender" duas letras de 7x5 de cada vez; ver Figura (V.6).

Em um dos experimentos de RUMELHART e ZIPSER (1985), uma rede com 7x14 unidades de entrada e duas unidades de categoria foi estimulada com os seguintes padrões: AA, AB, BA, e BB. A análise do comportamento do sistema mostra claramente que as unidades funcionam como detectores de letras em posições específicas. Em alguns casos, uma unidade responde às configurações AA e AB enquanto a outra responde aos estímulos BA e BB. Podemos dizer que a primeira unidade é um detector de A's na primeira posição do par e a outra um detector de B's nesta mesma posição. Em outros testes a classificação se invertia; uma unidade respondia a A's na segunda posição (respondia aos estímulos AA e BA) e a outra

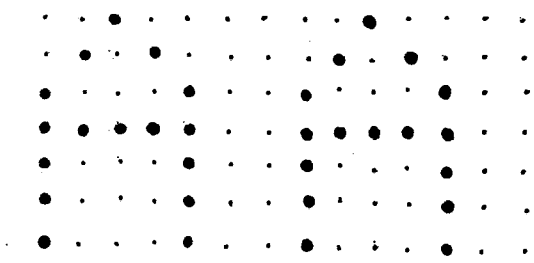

Figura V.6: Dois A's "acesos" na matriz de unidades de entrada

a B's na segunda posição (respondia aos estímulos AB e BB). De qualquer forma, uma unidade torna-se um detector de A's em uma determinada posição (posição 1 ou 2) e a outra unidade detecta B's na posição complementar.

Cada unidade de entrada possui uma ligação com todas as unidades de saída. Desta forma, pode-se exibir os valores dos pesos das ligações da entrada para cada unidade de saída em um arranjo similar à matriz de entrada, permitindo a visualização dos efeitos do aprendizado. Na Figura (V.7) o tamanho de um ponto corresponde à intensidade da ligação de uma determinada unidade de entrada para uma unidade de categoria específica. Observe que apesar destas unidades de categoria só responderem às letras na primeira posição do par, algumas ligações das unidades de entrada que "acendem" a segunda letra possuem peso não nulo. No entanto, estes pesos não são capazes de diferenciar um A de um B. Observe, também, que as unidades da segunda posição que só são ativadas por um determinado tipo de letra possuem pesos semelhantes, e aproximadamente valem metade dos pesos das unidade comuns a A e B; os pesos das unidades que nunca são ativadas possuem valor próximo a zero. Estes valores confirmam a previsão teórica de que no equilíbrio $w_{ij} \rightarrow P(\text{unidade } i \text{ ativa} \mid \text{unidade } j \text{ vence})$.

Este pequeno exemplo mostra uma característica fundamental do aprendizado competitivo: *a estrutura essencial que o mecanisnmo de aprendizado competitivo pode descobrir é representado na interseção dos estímulos.* Rumelhart e Zipser mostram diversas variações desta aplicação, dentre as quais uma onde o

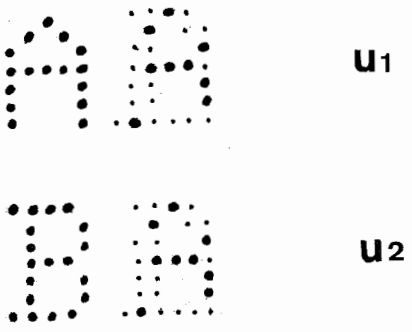

Figura V.7: Pesos das ligações. U1: unidade 1; U2: unidade 2

número de unidades de saída é aumentado para quatro e estas passam a responder a palavras particulares (e.g., AB); cada unidade responde a uma palavra. Isto ilustra a capacidade do algoritmo de aprendizado competitivo em descobrir vários tipos de estrutura dependendo do número de categorias da rede.

Capítulo VI

Auto-organização de Mapas de Características

VI.1 Introdução

Como afirmam RITTER e KOHONEN (1989), um dos problemas da teoria de redes neuronais (artificiais ou biológicas) mais intrigantes é elucidar mecanismos que possibilitem que sistemas simples descubram abstrações, invariâncias, e generalizações. Vimos ao estudar o aprendizado competitivo que diferentes unidades aprendem diferentes aspectos dos estímulos de entrada, o que pode ser considerado como um tipo de abstração bastante simples. A forma de aprendizado desenvolvida por KOHONEN (1982a-c; 1984), conhecida como *auto-organização de mapas de características*, é uma elaboração do aprendizado competitivo capaz de criar representações bastante interessantes da entrada: representações *topográficas*. O princípio básico de um sistema organizado topograficamente é que unidades fisicamente próximas respondam similarmente. Veremos mais adiante que o mecanismo sugerido por Kohonen possui muitos pontos em comum com o aprendizado competitivo básico.

Neste ponto, talvez seja interessante discutir, brevemente, as motivações biológicas por trás do algoritmo de Kohonen. Sabe-se há bastante tempo que o cérebro possui várias regiões com uma organização bastante peculiar, nas quais parece haver *mapas* de informações importantes para o organismo. Assim, na região sensorial somática, há um mapa somatotópico, ou somatosensorial, onde informações sensoriais arranjam-se topograficamente, ou seja, regiões corporais próximas pos-

suem representações neuronais próximas. Desta forma, por exemplo, os neurônios que são estimulados ao pressionarmos um determinado dedo estão fisicamente próximos de neurônios que são estimulados por um outro dedo (da mesma mão), e mais distantes de neurônios relacionados com o braço do indivíduo. Para muitos neurocientistas esta topografia parece bastante importante pois há inúmeras regiões do cérebro com esta organização (e.g., áreas visuais; áreas motoras).

Segundo KOHONEN (1982c), uma das razões para que cérebros — compostos por elementos computacionais muito simples — sejam "dispositivos" tão poderosos é que estes são estruturados segundo alguns princípios poderosos. Ou seja, o cérebro não é apenas um coleção imensa de elementos computacionais; é uma coleção estruturada. Uma das características de organização mais ubíquas no cérebro, mas que foi sempre ignorada em redes neuronais, é a *ordem espacial das unidades de processamento* (KOHONEN, 1982c). Para Kohonen, esta ordem é bastante importante no cérebro, e ele explora este princípio em redes neuronais elaborando mecanismos que permitam que estas formem, automaticamente, representações topológicas (estruturadas) dos estímulos de entrada.

Os mecanismos de auto-organização permitem que unidades alterem as suas respostas de uma maneira tal que a localização da unidade em uma rede torne-se específica a uma certa característica no conjunto de estímulos de entrada. Além disso, esta especificidade ocorre na mesma *ordem topológica que está presente nas relações de similaridade (métricas) dos estímulos de entrada.* Em outras palavras, após o aprendizado as unidades respondem a diferentes estímulos de uma maneira *ordenada, definindo um sistema de coordenadas de características sobre a rede.* Cada estímulo gera uma resposta localizada, cuja posição na rede reflete as suas mais importantes "coordenadas de características".

A arquitetura típica para a auto-organização de mapas de características é exibida na Figura (VI.1). Há duas camadas de unidades. As unidades de entrada possuem ligações para todas as unidades de saída, e os pesos das ligações são adaptáveis. As unidades de saída estão organizadas em um arranjo bi-dimensional e estão altamente interconectadas (os pesos destas conexões laterais são fixos). A Figura (VI.1) exibe uma rede onde as unidades de saída estão organizadas segundo

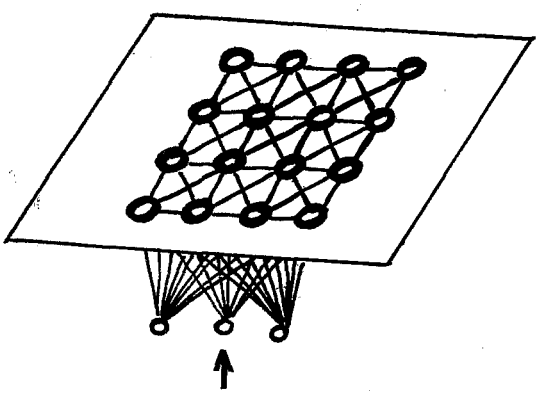

Figura VI.1: Arquitetura para a auto-organização de mapas

um quadrado.

A ativação das unidades de saída é determinada pelo estímulo direto de entrada e pelas interações laterais com outras unidades de saída. Estas interações laterais são dadas pela função da Figura (VI.2a). Uma determinada unidade de saída excita os seus vizinhos mais próximos e inibe outros mais distantes, gerando "bolhas de ativação" na camada de saída (ver Figura (VI.2b–c)), as quais ocorrem em regiões onde a excitação de entrada é máxima (a forma exata da bolha depende das excitações de entrada). Além disso, o raio da bolha depende da função da Figura (VI.2a). Se as conexões laterais positivas forem mais fortes temos a Figura (VI.2b), e no caso da inibição prevalecer ocorre como na Figura (VI.2c). Para maiores detalhes acerca das interações laterais e a formação de "bolhas" ver, e.g., (KOHONEN, 1984).

O mecanismo de auto-organização em si é bastante simples. Os pesos de todas as ligações recebem valores aleatórios fazendo com que as unidades respondam ao acaso aos padrões de entrada. Ao ser apresentado um novo padrão, alguma unidade de saída responderá melhor à entrada e esta unidade e suas vizinhas — definidas por alguma região em torno dela — terão os pesos de suas ligações alterados. Com isto, a unidade de melhor resposta — ou unidade vencedora — aumentará as suas chances de responder a padrões semelhantes, e suas vizinhas responderão mais como ela do que antes. Algumas ligações serão reforçadas (aquelas cujas unidades de

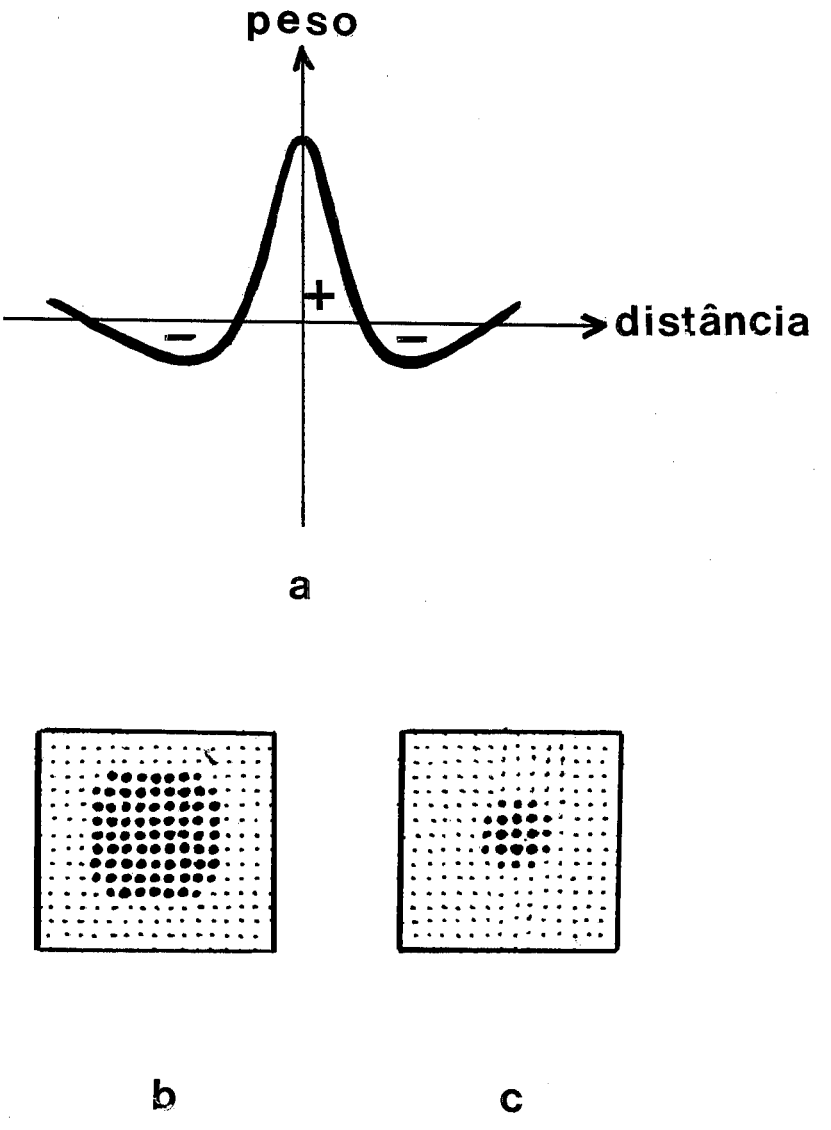

Figura VI.2: a). Interações laterais. b-c). "Bolhas" de ativação

entrada estão ativadas) e outras diminuídas (unidades de entrada que estão desativadas) mantendo a soma total de pesos para a unidade aproximadamente constante. Apesar de extremamente simples, este mecanismo de aprendizado obtém uma organização topográfica: regiões da camada de saída apresentam comportamento de resposta similares, e uma ordem global se origina a partir da organização de uma região em relação às outras.

É interessante ressaltar que esta organização *global* das unidades de saída surge como resultado de modificações *locais* — alteração dos pesos nas vizinhanças da unidade vencedora. Intuitivamente, isto ocorre pois quando uma unidade vencedora altera seus pesos tornando-se mais seletiva a um particular padrão, todas as unidades da sua vizinhança passam a responder melhor a padrões semelhantes. Cada região passa, então, a influenciar regiões vizinhas, o que com o tempo, acaba por gerar uma organização (topográfica) global.

Tecnicamente falando, após o aprendizado, as posições espaciais das unidades especificam um mapeamento dos padrões de entrada sobre um arranjo bi-dimensional, sendo que este arranjo possui a propriedade de um mapa topográfico, ou seja, as unidades de saída representam relações de distância no espaço n-dimensional dos estímulos de entrada aproximadamente como distâncias no arranjo bi-dimensional de unidades.

Estas propriedades do algoritmo de Kohonen decorrem das interações laterais das unidades de saída e de uma regra de aprendizado bastante simples. Aparentemente, os principais requisitos para a auto-organização são (RITTER e KOHONEN, 1989): (i) as unidades são expostas a um número suficiente de estímulos diferentes; (ii) para cada estímulo, os pesos das ligações das unidades excitadas (i.e., da "bolha") são os únicos afetados; (iii) atualizações similares nos pesos são impostas em várias unidades adjacentes; e (iv) o ajuste resultante é tal que aumenta as mesmas respostas a um estímulo subseqüente suficientemente similar.

Não é preciso muito esforço para observar que o mecanismo de auto-organização é praticamente idêntico ao algoritmo de aprendizado competitivo. A diferença básica é que no aprendizado competitivo as alterações estão confinadas à

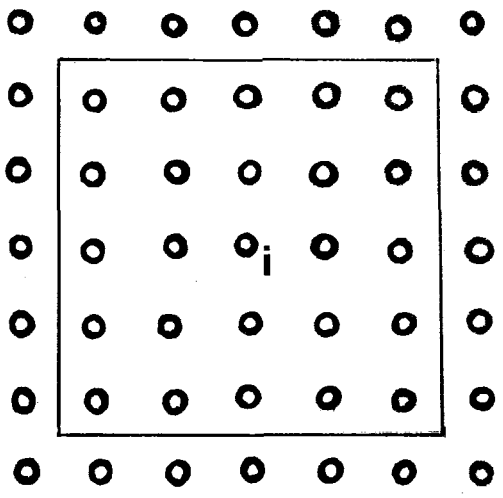

Figura VI.3: Possível vizinhança de uma unidade. A área delineada corresponde à vizinhança da unidade i

unidade vencedora, enquanto que neste novo esquema toda uma região em torno da vencedora sofre alterações. A formação de uma bolha de ativação em uma região restrita da rede é o fenômeno mais importante para a auto-organização desta área. Como a bolha se forma no local de excitação máxima, podemos especificar o algoritmo de aprendizado da seguinte forma: (i) descubra o máximo de ativação; (ii) defina uma vizinhança em torno deste máximo (correspondendo à bolha); e (iii) altere os pesos das ligações nesta região.

A Figura (VI.3) exibe como pode ser definida a vizinhança em um arranjo bi-dimensional quadrado das unidades de saída. Resultados experimentais (KOHONEN, 1984) e teóricos (RITTER e SCHULTEN, 1988) mostraram que os melhores resultados de auto-organização são obtidos quando a vizinhança de uma determinada unidade diminui com o tempo. Observe, também, que a vizinhança das unidades das bordas do arranjo não são completas (isto pode gerar deformações no mapa resultante).

VI.2 O algoritmo de auto-organização

Descreveremos, agora, o algoritmo proposto por Kohonen para a auto-organização de mapas de características.

Em muitos casos é comum que o algoritmo de Kohonen determine o máximo de ativação de acordo com a menor diferença entre os vetores **o** (associado ao estímulo de entrada) e **w**. Ou seja, o algoritmo utiliza a distância Euclidiana para determinar a unidade de saída que possui o vetor peso que melhor representa o vetor estímulo. A seguir apresentamos o algoritmo.

- **Auto-organização de Mapas de Características**

  - **Passo 1. Inicialização**

    Inicializar pesos com valores aleatórios pequenos.

  - **Passo 2. Entrada**

    Novo padrão é fornecido à rede. (O vetor **o** associado deve ser normalizado.)

  - **Passo 3. Competição**

    Determinar a unidade j tal que

    $$\|\mathbf{o} - \mathbf{w}_j\| = \min_c\{\|\mathbf{o} - \mathbf{w}_c\|\},$$

    onde $\mathbf{w}_k = (w_{1k}, \ldots, w_{nk})$.

  - **Passo 4. Adaptação**

    Atualizar pesos segundo a regra

    $$w_{ik}(t+1) = \begin{cases} w_{ik}(t) + \eta(t)(o_i(t) - w_{ik}(t)) & k \in N_j(t) \\ w_{ik}(t) & k \notin N_j(t) \end{cases},$$

    onde j é a unidade vencedora, $N_j(t)$ é o conjunto de unidades consideradas na vizinhança de j, i varia nas unidades de entrada, e $\eta(t)$ $(0 \leq \eta(t) \leq 1)$ é a função taxa de aprendizado. Em muitas utilizações do algoritmo $\eta(t)$ decresce com o tempo; em outras $\eta(t)$ é uma função constante.

– **Passo 5. Loop**

Vá para o Passo 2.

Uma maneira mais conveniente de expressar as vizinhanças das unidades é utilizar uma função h_{kj} para especificar a extensão e a forma da atividade centrada na unidade j do Passo 3. A função h_{kj} determina o valor da ativação da unidade k, se a ativação é centrada em j. Esta função pode ser constante para todas as unidades k próximas de j; tipicamente h_{kj} é uma Gaussiana com valor máximo para $k = j$ e tendendo a zero à medida que k se afaste de j. O Passo 4 pode ser, então, reescrito como em (VI.1):

$$w_{ik}(t+1) = w_{ik}(t) + \eta(t)h_{kj}(o_i(t) - w_{ik}(t)). \qquad \text{(VI.1)}$$

Tanto a equação (VI.1) quanto o Passo 4 possuem a propriedade de confinar as adaptações à vizinhança de j e melhorar, nesta região, respostas subseqüentes ao padrão apresentado.

A equação (VI.1) deixa claro que o algoritmo de Kohonen é uma sofisticação do "instar", na qual as alterações dos pesos são proporcionais à distância à unidade vencedora.

Deve ser enfatizado que o algoritmo de aprendizado descrito, ou a equação (VI.1), é uma aproximação de um conjunto de equações que especificam formalmente o comportamento do sistema; ver, e.g., (KOHONEN, 1984) para maiores detalhes. A formação da bolha, através das interações laterais é substituída pela definição de uma vizinhança N (ou pela função h). Além disso, como nesta versão simplificada não há normalização dos pesos, é comum convencionar-se que os padrões de entrada estarão normalizados a priori. Como os pesos tendem às entradas, este procedimento resolve o problema da saturação sináptica.

Resultados matemáticos rigorosos do algoritmo de Kohonen podem ser obtidos em (KOHONEN, 1982b) e (RITTER e SCHULTEN, 1988). No presente contexto, é suficiente destacar que os mapas resultantes são projeções não-lineares do espaço de entrada na superfície das unidades de saída que apresentam duas propriedades principais (RITTER e KOHONEN, 1989): (i) As relações de distância entre

os dados de entrada são preservadas pelas suas imagens no mapa da forma mais acurada possível. Entretanto, um mapeamento de um espaço n-dimensional para outro m-dimensional, onde $n > m$, freqüentemente distorcerá a maioria das distâncias e preservará apenas as relações de vizinhança mais importantes dos estímulos de entrada; nesta representação reduzida informações irrelevantes são ignoradas. (ii) Se estímulos diferentes aparecem com freqüências diferentes, os mais comuns serão mapeados para regiões maiores (no arranjo de saída), às custas dos menos comuns. Isto resulta em uma representação bastante econômica. Assim, a *escala* ou *fator de magnificação* do mapa não é constante, e sim uma função da freqüência dos estímulos de entrada que forem mapeados para aquela região.

O algoritmo de auto-organização também se comporta como um algoritmo de conglomeração. Se os estímulos formam conglomerados no espaço de entrada, o algoritmo fará com que os padrões de um conglomerado sejam mapeados para uma região localizada comum no mapa, e arranjará estas regiões ao longo do mapa de maneira a capturar o máximo da topologia geral dos conglomerados de entrada quanto for possível. Além disso, podemos considerar que cada região do mapa possui um vetor protótipo, o qual é refinado ao longo do aprendizado. Segundo LIPPMANN (1987), o algoritmo de Kohonen é similar a um algoritmo de conglomeração bastante conhecido: o algoritmo "K-means".

VI.3 Visualização do processo de auto-organização

Considere a situação onde vetores padrão são tridimensionais; conseqüentemente os vetores peso são também tri-dimensionais. Além disso, suponha que a norma destes vetores seja unitária. Como sabemos, neste caso, tanto os padrões quanto os pesos podem ser representados como pontos na superfície de uma esfera unitária. A Figura (VI.4) exibe um possível conjunto inicial de vetores peso. Para efeitos de visualização do processo de auto-organização, uma linha conectando dois pesos (pontos na esfera) quaisquer indica que estas unidades são adjacentes no arranjo de unidades de saída. Desta forma, os pontos 1 e 2 correspondem aos pesos de duas

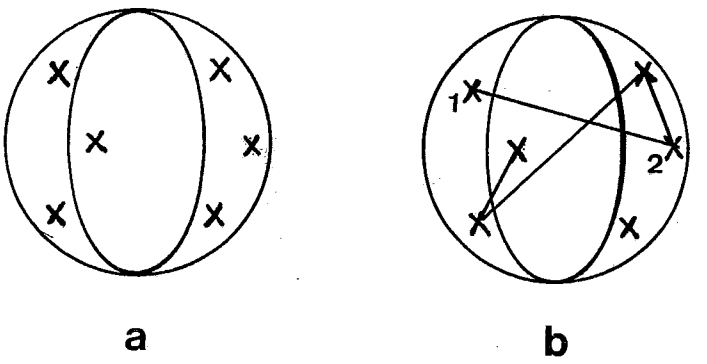

Figura VI.4: Configuração inicial de pesos. a) Conjunto inicial de pesos. b) Unidades adjacentes 1 e 2 possuem pesos bastante distintos

unidades de saída adjacentes. Observe que os pesos são bastante distintos pois estão bastante separados.

A Figura (VI.5) ilustra a evolução dos pesos em uma dada aplicação. Inicialmente, os pesos de duas unidades vizinhas são bastante distintos e conseqüentemente suas respostas são distintas. Com o passar do tempo, unidades próximas adquirem pesos mais e mais semelhantes, até que em um determinado instante ocorre a formação de um mapa topográfico (i.e., unidades adjacentes exibem respostas muito parecidas). Vê-se claramente na Figura (VI.5) que uma rede inicialmente desestruturada organiza-se sucessivamente até exibir uma organização global.

Alguns trabalhos têm tentado caracterizar mais formalmente o processo de auto-organização da rede (ver, e.g., KOHONEN, 1982b). Recentemente, RITTER e SCHULTEN (1988) descreveram — em certos casos — este processo como a minimização de uma função que mede o grau de ordenação topográfica do mapa. Esta função pode assumir diversos mínimos locais, correspondendo a mapas com defeitos topológicos particulares. De forma a melhorar a organização do mapa (diminuir o número de mínimos locais) deve-se começar o processo com vizinhanças grandes, e, vagarosamente, diminuí-las.

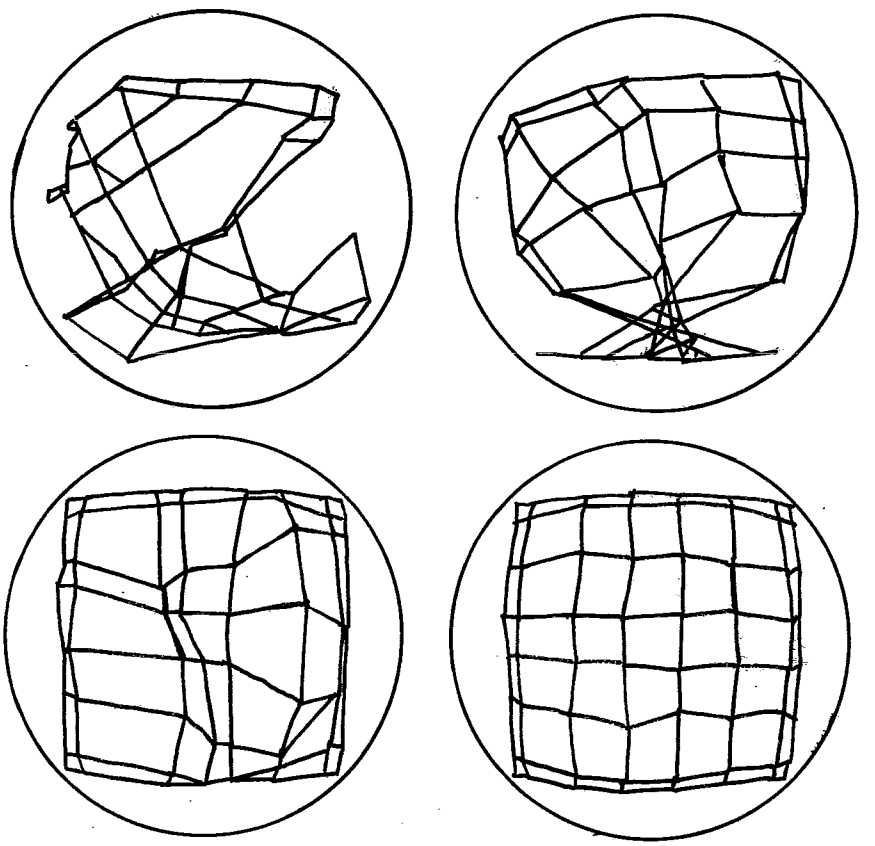

Figura VI.5: Processo de auto-organização: Evolução dos pesos

HINTON (1989) sugere informalmente que a auto-organização pode ser vista como a minimização de uma função de custo que possui dois termos. O primeiro termo mede o quão inacuradamente o vetor peso da unidade vencedora representa o vetor de entrada. O segundo termo mede a dissimilaridade entre os padrões de entrada que são representados por unidades adjacentes. Desta forma, ao mesmo tempo que os vetores peso melhoram as suas representações de padrões particulares, vetores próximos tendem a ser similares (gerando respostas similares). Esta interpretação está, claramente, de acordo com a nossa visão intuitiva do processo de aprendizado.

A existência de uma função que é minimizada pode dar a entender que a auto-organização de mapas é um mecanismo de aprendizado estável. Entretanto, no caso mais geral, onde a complexidade do ambiente externo faz com que os padrões de entrada submetidos à rede mudem com o tempo — i.e., a função estocástica representando o ambiente é variável no tempo —, isto não é verdade.

VI.4 Algumas aplicações do algoritmo de Kohonen

Desde a sua elaboração, o algoritmo de Kohonen vem sendo aplicado em uma vasta gama de aplicações que incluem: (a) Formação de mapas sensorias (tal como no cérebro) (RITTER e SCHULTEN, 1988); (b) Otimização combinatória (e.g., aplicações para o problema do caixeiro viajante) (FORT, 1988); Robótica (e.g., coordenação visuo-motora) (RITTER et al., 1989); (d) Análise estatística de dados (KOHONEN, 1982c); (e) Representação de conhecimento (RITTER e KOHONEN, 1989), e (f) Reconhecimento de fala (KOHONEN, 1988).

VI.4.1 Reconhecimento de fala

O sistema para reconhecimento de fala a ser discutido agora possui 15 unidades de entrada e um conjunto de unidades de saída arranjadas segundo um hexágono. Um dos objetivos principais do sistema desenvolvido por KOHONEN (1988) é criar um

mapa topográfico bi-dimensional de elementos de fala (fonemas) na rede.

As unidades de entrada recebem uma versão pré-processada do sinal acústico original (proveniente de um microfone), correspondendo a uma representação espectral deste sinal. Após o processo de auto-organização observam-se alguns resultados interessantes: As unidades de saída da rede passam a responder seletivamente ao conteúdo espectral de fonemas distintos, e exibem uma organização topográfica ao longo do arranjo de unidades de saída. É importante salientar que ao longo do aprendizado não foram apresentados fonemas à rede, e sim conteúdos espectrais. A razão para a rede organizar-se desta forma é que a entrada está agrupada em torno de fonemas, e o processo de aprendizado descobre os conglomerados. Observa-se, então, a formação de um mapa fonotópico onde as distâncias entre pontos no mapa (unidades) são aproximadamente proporcionais às diferenças vetorias entre os conteúdos espectrais originais (KOHONEN, 1988).

Obviamente, o reconhecimento de fala é algo muito mais complexo do que o descrito brevemente acima. Na realidade, o sistema de Kohonen é bastante sofisticado e pode ser dividido em três módulos principais: (i) Um módulo pré-processador (basicamente transforma sinal acústico em representação espectral); (ii) Módulo contendo uma rede neuronal auto-organizável; e (iii) Um módulo pós-processador que faz ajustes das informações do módulo (ii) e imprime o texto. O sistema foi testado com milhares de palavras (em Finlandês) e com a voz de várias pessoas, e apresenta uma taxa de acertos que varia entre 92 e 97 porcento.

VI.4.2 Auto-organização de mapas semânticos

O algoritmo de aprendizado de Kohonen foi utilizado em uma série de aplicações de reconhecimento de padrões. Recentemente, RITTER e KOHONEN (1989) aplicaram este algoritmo em uma tarefa de mais "alto nível": a auto-organização de mapas semânticos. O objetivo deste estudo foi observar as representações obtidas pela rede quando estimulada por informações lingüísticas.

Não entraremos nos detalhes deste estudo mas a Figura (VI.6) dá uma mostra dos resultados após o aprendizado: As unidades respondem melhor

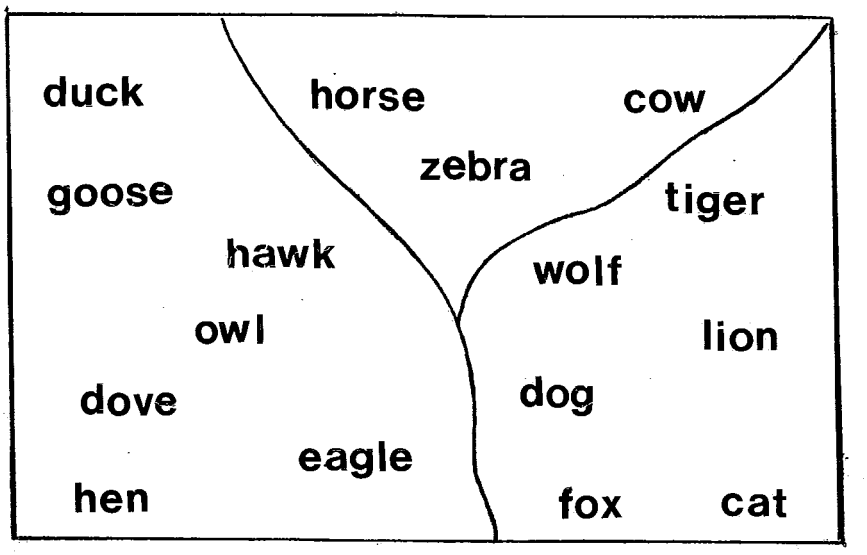

Figura VI.6: Mapa topográfico contendo os padrões (animais) submetidos à rede

a determinados estímulos — determinados animais, neste caso —, e os animais agrupam-se segundo as suas semelhanças.

Capítulo VII

Instabilidade do Aprendizado Competitivo

Como já vimos, as questões de estabilidade do aprendizado não-supervisionado são extremamente importantes pois muitos dos sistemas que utilizam tais mecanismos funcionam continuamente — i.e., são adaptáveis todo o tempo — em ambientes imprevisíveis e complexos. Discutiremos, neste capítulo, a análise de Grossberg sobre o aprendizado competitivo, um trabalho que o levou a desenvolver, posteriormente, a Teoria da Ressonância Adaptativa (ART). Este capítulo é importante também para podermos entender as motivações por trás dos sistemas ART.

Um dos objetivos de Grossberg é estudar mecanismos capazes de gerar *detectores de características* em modelos de redes neuronais. Seus trabalhos são extremamente ricos, com inúmeras contribuições e, além disso, bastante complexos devido ao rigor matemático de seus modelos. Grossberg analisou detalhadamente diversos mecanismos de aprendizado em redes neuronais, e em particular o aprendizado competitivo.

Os mecanismos de aprendizado competitivo estudados por GROSSBERG (1976a) diferem de outros em inúmeros aspectos importantes. Em particular, a sua solução para o problema da saturação sináptica não envolve a conservação da quantidade total de peso de uma determinada unidade. Ao invés disso, ele normaliza a quantidade total de ativação da camada de entrada. Ao receber um estímulo de entrada qualquer (i.e., não normalizado) as interações das unidades da camada de

entrada (regidas por equações diferenciais) normalizam a sua ativação. O sistema estudado por Grossberg obedece às equações (VII.1–2). A equação (VII.1) rege a ativação das unidades de saída, enquanto que (VII.2) especifica como os pesos variam:

$$o_j = \begin{cases} 1 & \text{se } net_j > \max\{\epsilon, net_k | k \neq j\} \\ 0 & \text{se } net_j < \max\{\epsilon, net_k | k \neq j\} \end{cases} \quad \text{(VII.1)}$$

$$\frac{dw_{ij}}{dt} = (-w_{ij} + \theta_i) o_j \quad \text{(VII.2)}$$

onde $\theta_i = o_i / n_k$, ϵ é uma constante pequena e $n_k = \sum_j o_j$, como definido no algoritmo de aprendizado competitivo do capítulo 5. As equações (VII.1-2) são basicamente as equações de RUMELHART e ZIPSER (1985) escritas de outra forma. A expressão $\max\{\epsilon, net_k\}$ em (VII.1) garante simplesmente que a entrada da unidade j deve ter uma excitação mínima para alterar os pesos.

Grossberg analisou o sistema (VII.1–2) rigorosamente e demonstrou o seguinte teorema (GROSSBERG, 1976a):

Teorema. (Um Padrão)

Dado um padrão S_k, suponha que existe uma única unidade j tal que

$$net_j(0) > \max\{\epsilon, net_k(0) | k \neq j\}.$$

Deixe S_k ser praticado durante a seqüência de intervalos $[U_k, V_k]$, $k = 1, 2, \ldots$. Então, o ângulo entre $\mathbf{w}_j(t)$ e $\mathbf{o}$ (associado a S_k) decresce monotonicamente, e $net_j(t)$ tende monotonicamente a $\|\mathbf{o}\|^2$. Além disso, exceto no caso onde $net_j(0) = \|\mathbf{o}\|^2$,

$$\lim_{t \to \infty} \|\mathbf{w}_j(t)\|^2 = \lim_{t \to \infty} net_j(t) = \|\mathbf{o}\|^2,$$

se e somente se

$$\sum_{k=1}^{\infty} (V_k - U_k) = \infty.$$

Ou seja, $\|\mathbf{w}_j(t)\|^2 \to \|\mathbf{o}\|^2$ só ocorre se a seqüência de aprendizado for infinita. Dito em outras palavras, o teorema acima afirma que o aprendizado faz $\mathbf{w}_j$ paralelo a $\mathbf{o}$ e normaliza o comprimento de $\mathbf{w}_j$.

Este importante teorema garante que o sistema definido pelas equações (VII.1–2) consegue aprender perfeitamente um padrão que seja praticado por

tempo suficiente. Entretanto, ele nada afirma quando a rede é estimulada por um conjunto de padrões. Esta situação é, efetivamente, a situação de interesse, já que qualquer aplicação não trivial envolverá a apresentação de muitos estímulos.

O que ocorre se diferentes padrões S_k estimulam a rede ao longo do aprendizado? Como podemos saber que as mudanças nos pesos devidas a um padrão não se oporão a outras mudanças devidas a padrões diferentes? Aparentemente, a *escolha* que ocorre nas unidades de saída (apenas uma unidade fica ativa segundo (VII.1)) garante a coerência das mudanças. Suponha que

$$\mathbf{o}_k\mathbf{w}_k(0) > \max\{\epsilon, \mathbf{o}_k\mathbf{w}_j(0) | j \neq k\}, \tag{VII.3}$$

onde o índice k explicita a vinculação entre $\mathbf{o}_k$ e S_k. Ou seja, no instante $t = 0$, S_k é codificado pela unidade k. Seja S_1 o primeiro padrão a estimular a rede. Por (VII.3), a unidade 1 recebe a maior excitação e só ela ficará ativa. Com isto, só a unidade 1 alterará seus pesos e pelo teorema acima isto faz $\mathbf{w}_1$ mais paralelo a $\mathbf{o}_1$. Quando o padrão S_2 estimula a rede, a unidade 2 recebe a maior excitação já que $\mathbf{w}_1$ está mais paralelo a $\mathbf{o}_1$ do que antes e por (VII.3) todas as outras unidades recebem excitação menor. Novamente, apenas $\mathbf{w}_2$ é alterado (se aproximando de $\mathbf{o}_2$). Este comportamento prossegue ao longo do aprendizado. Ao permitir que apenas a unidade vencedora altere os seus pesos, este esquema, aparentemente, permite o aprendizado de vários padrões.

Entretanto, a argumentação acima não está correta. Considere a Figura (VII.1). É fácil ver que é possível, por exemplo, ao tornarmos $\mathbf{w}_1(t)$ mais paralelo a $\mathbf{o}_1$ fazer com que ele fique mais paralelo a $\mathbf{o}_2$ do que $\mathbf{w}_2(t)$. Assim, quando S_2 for apresentado ele será codificado pela unidade 1. Em outras palavras, o aprendizado pode *recodificar* os padrões de entrada.

GROSSBERG (1976a) ilustra uma situação onde todas as unidades de saída reclassificam os seus padrões de entrada. Além disso, essa reclassificação pode continuar indefinidamente. Este comportamento é sem dúvida altamente insatisfatório. Entretanto, se há *poucos* padrões em relação ao número de unidades de saída, então o aprendizado será *estável.* De fato, Grossberg demonstrou, também, que se os vetores peso inicialmente codificam os padrões em classes esparsas, então

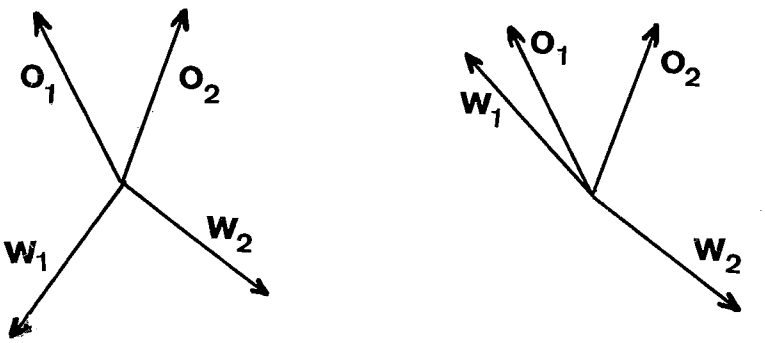

Figura VII.1: Aprendizado pode recodificar vetores.

esta codificação se mantém com o tempo; ver (GROSSBERG, 1976a) para maiores detalhes. Com isto, dado um número de padrões de entrada fixo, torna-se mais fácil obter uma codificação estável à medida que o número de unidades de categoria aumenta. No entanto, uma vez construída a rede não mais podemos aumentar o número de unidades de saída.

Algumas variações do aprendizado competitivo foram também analisadas por Grossberg, as quais mostraram ser, também, instáveis. A análise destas variações mais sofisticadas do esquema básico de aprendizado mostrou que o seu comportamento instável não se deve à sua simplicidade; ver (GROSSBERG, 1987).

Segundo Grossberg, um princípio bastante importante para a elaboração de mecanismos de categorização estáveis é que após um vetor peso particular aprender um determinado padrão, ou classe de padrões, ele deve ser impossibilitado de codificar padrões muito diferentes, independentemente da seqüência de aprendizado.

A instabilidade do aprendizado competitivo, ou outros sistemas tipo "instar", é proveniente de duas fontes (CARPENTER, 1989). Primeiro, mesmo que a categoria escolhida seja a melhor para representar uma dada entrada, esta representação pode, mesmo assim, ser pobre, escolhida apenas porque todas as outras

são ainda piores. Segundo, regras de alteração de pesos como a da equação (VII.2) fazem com que o vetor peso tenda a um novo vetor que codifica o padrão de ativação corrente, diminuindo assim o registro do passado. Desta forma, vetores peso podem mais cedo ou mais tarde se distanciar dos seus padrões originais, mesmo que o aprendizado seja muito lento e que cada padrão de entrada individual case bem com a sua representação.

Tanto o aprendizado competitivo quanto a auto-organização de mapas de características podem ser vistos como formas de estabelecer códigos — i.e., representações — do ambiente externo. Segundo Grossberg, o desenvolvimento de categorizações estáveis exige *mecanismos de "feedback"* capazes de estabilizar os códigos aprendidos.

Ao estudar o condicionamento clássico, Grossberg (ver GROSSBERG, 1980) foi levado a considerar uma série de questões, tais como: a) mecanismos de atenção; b) o papel de expectativas; e c) a comparação de expectativas com informações sensoriais. Isto levou-o a estabelecer a importância de mecanismos de feedback para comparar expectativas com informações a serem categorizadas, e a incorporar tais mecanismos em seus modelos de aprendizado em redes neuronais. Como afirma GROSSBERG (1988):"... we perceive, in part, what we *expect* to perceive based upon past *learning.*"

Considere uma rede neuronal simples com duas camadas de unidades; uma de entrada e uma de categorias denotadas por F_1 e F_2, respectivamente. O funcionamento desta rede é similar ao de uma que obedece ao aprendizado competitivo (ver capítulo 5). Suponha que o padrão de entrada S_1 é codificado pela categoria c_1 de F_2. Quando um novo padrão S_2 é apresentado à rede, ele deve ativar uma outra categoria no caso dele ser bastante diferente de S_1. Suponha, no entanto, que a rede comete um erro e S_2 também é codificado por c_1. Independentemente do mecanismo para corrigir este erro, ele não pode ser mediado por F_2 já que segundo F_2, S_1 e S_2 são indistinguíveis; F_2 sozinho não é capaz de corrigir a categorização errônea.

Onde na rede pode, em princípio, ser detectado este erro? No instante

em que S_2 ativou c_1, não há nenhuma informação armazenada de que anteriormente era S_1 que ativava c_1, e não S_2. De alguma forma esta informação deve ser representada na *dinâmica* da rede. Para testar se a categorização realizada é correta, criam-se ligações de feedback ($F_2 \rightarrow F_1$), as quais enviam para F_1 o padrão que F_2 "acha" que está presente em F_1 — i.e., o padrão que F_2 acha que foi responsável pela ativação de c_1. O comportamento desta nova rede envolve, agora, dois passos: primeiro o padrão de entrada ativa a sua categoria; segundo, esta categoria manda para F_1 um padrão de feedback. No caso em questão, o padrão de feedback é o padão S_1 previamente aprendido (pois c_1 era ativada por S_1). Assim, quando S_2 erradamente ativa c_1, esta pode gerar o padrão S_1 em F_1. Quando isto acontece, os padrões S_1 (vindo de F_1) e S_2 (vindo da entrada) estarão simultaneamente ativando F_1, e poderão ser comparados para testar se a categorização foi adequada.

Resumindo, o padrão aprendido de feedback representa o padrão que c_1 *espera* estar em F_1 devido ao seu prévio aprendizado. Com as ligações de feedback, o padrão de entrada de F_1 e a expectativa aprendida de F_2 podem, então, ser comparados em F_1.

Uma maneira de compreender melhor a discussão acima é através do instar e do outstar. As ligações $F_1 \rightarrow F_2$ implementam um instar e permitem a categorização de padrões. Por sua vez, as ligações de feedback $F_2 \rightarrow F_1$ implementam um outstar que deve aprender o padrão de ativação presente em F_1. Assim, quando S_1 é categorizado por c_1, o vetor peso de feedback associado a esta categoria aprende o padrão $\mathbf{o_1}$ (correspondente a S_1). Mais tarde, quando S_2 ativa novamente c_1, o padrão $\mathbf{o_1}$ será mandado para F_1 podendo, então, ser comparado com $\mathbf{o_2}$ (correspondente à entrada da rede).

Intuitivamente, ao não permitir que as categorias do sistema aceitem padrões muito distintos (os vetores peso não variem muito) pode-se obter uma categorização estável. Entretanto, muitos aspectos importantes foram deixados de lado nesta discussão inicial. Talvez o mais óbvio deles seja como a detecção de erro de categoria permite a recodificação adequada de S_2 por uma outra categoria de F_2. Esta, e muitas outras questões importantes serão discutidas no próximo capítulo ao apresentarmos os sistemas ART.

Capítulo VIII

Teoria da Ressonância Adaptativa

VIII.1 O Dilema da Estabilidade-Plasticidade

Neste capítulo estudaremos a Teoria da Ressonância Adaptativa (ART) introduzida por GROSSBERG (1976b) e posteriormente elaborada por CARPENTER e GROSSBERG (1987a).

Neste sentido, é importante destacar um dos principais problemas que os sistemas ART tentam, e de fato conseguem, resolver: *O Dilema da Estabilidade-Plasticidade* (CARPENTER e GROSSBERG, 1987a). Um sistema auto-organizável deve possuir duas características importantes:

-*Plasticidade*: Um sistema deve ser capaz de se adaptar em resposta a um conjunto de estímulos do ambiente representando adequadamente as regularidades dos estímulos.

-*Estabilidade*: Representações já estabelecidas — i.e., aprendidas — não devem ser perdidas ao longo da evolução do sistema.

Surge, então, o importante dilema da estabilidade-plasticidade: Um sistema deve apresentar *plasticidade* de modo a aprender novos eventos significativos, mas ao mesmo tempo ele deve permanecer *estável* ao responder a eventos irrelevantes ou repetidos. O dilema sugere as seguintes questões: Como um sistema pode preservar o seu conhecimento previamente adquirido e ao mesmo tempo ser capaz de aprender eventos novos? O que previne a destruição de memórias antigas

quando novas memórias são aprendidas? Em outras palavras, como um sistema pode conciliar duas propriedades, a priori, conflitantes: a estabilidade e a plasticidade?

Estas questões salientam uma das características principais para tornarem o sistema cognitivo humano tão poderoso: A sua habilidade de aprender representações internas de uma enorme quantidade e variedade de estímulos ambientais em tempo real e sem um professor. Desta forma, ao modelarmos sistemas cognitivos não podemos ignorar esta questão fundamental.

Como já vimos, diversas variações do aprendizado competitivo não são capazes de resolver o dilema descrito; eles são naturalmente instáveis. Outros modelos (e.g., Kohonen), são também instáveis. As soluções adotadas por muitos modelos para contornar (mas não solucionar) o problema da instabilidade são basicamente duas:

(a) Interromper externamente a plasticidade do sistema antes da destruição de representações. Entretanto, surge aí um problema sério: Quando interromper? A interrupção precoce poderá acarretar a não considerção de eventos importantes. Já a interrupção tardia poderá dar margem a recodificações indesejáveis.

(b) Restringir a natureza do ambiente externo. Ou seja, a questão importante é determinar que tipos de ambientes restritos podem ser tratados pelos respectivos modelos sem que suas representações sejam perdidas.

Nas seções a seguir discutiremos como os sistemas ART resolvem o dilema, bem como as principais características e propriedades destes modelos. Neste capítulo designaremos por ART os sistemas que são correntemente conhecidos por ART 1, os quais funcionam apenas com padrões binários. Sistemas ART 2 são capazes de processar estímulos analógicos (CARPENTER e GROSSBERG, 1987b). Mais recentemente, foram desenvolvidos os sistemas ART 3, os quais suportam hierarquias de representações (possuem várias camadas) (CARPENTER e GROSSBERG, 1990).

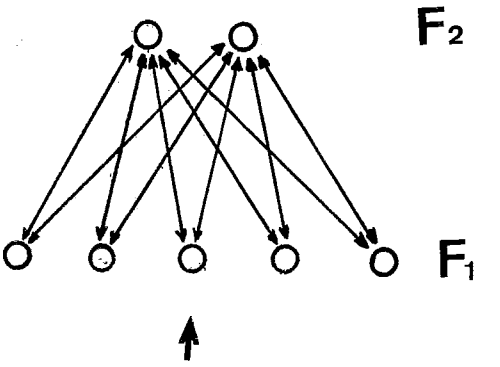

Figura VIII.1: Alguns componentes da arquitetura ART

VIII.2 A Arquitetura ART

A Figura (VIII.1) mostra alguns componentes da arquitetura ART. O sistema possui duas camadas de unidades: uma camada de entrada usualmente denotada por F_1 e uma camada de saída, F_2. As duas camadas estão totalmente interligadas; as ligações são bi-direcionais.

Como em um sistema de aprendizado competitivo, a camada F_2 possui unidades de categoria, e realiza uma categorização da entrada; cada unidade representa uma categoria distinta.

A arquitetura ART completa possui muitos outros componentes e é bastante elaborada. A Figura (VIII.2) exibe a configuração completa de um módulo ART. Na prática, considera-se que o sistema possui três camadas, sendo que a saída da camada de entrada é distribuída para a camada F_1, para os controles de ganho e para o mecanismo de "arousal", **A** (as linhas da figura correspondem, na verdade, a um conjunto de ligações).

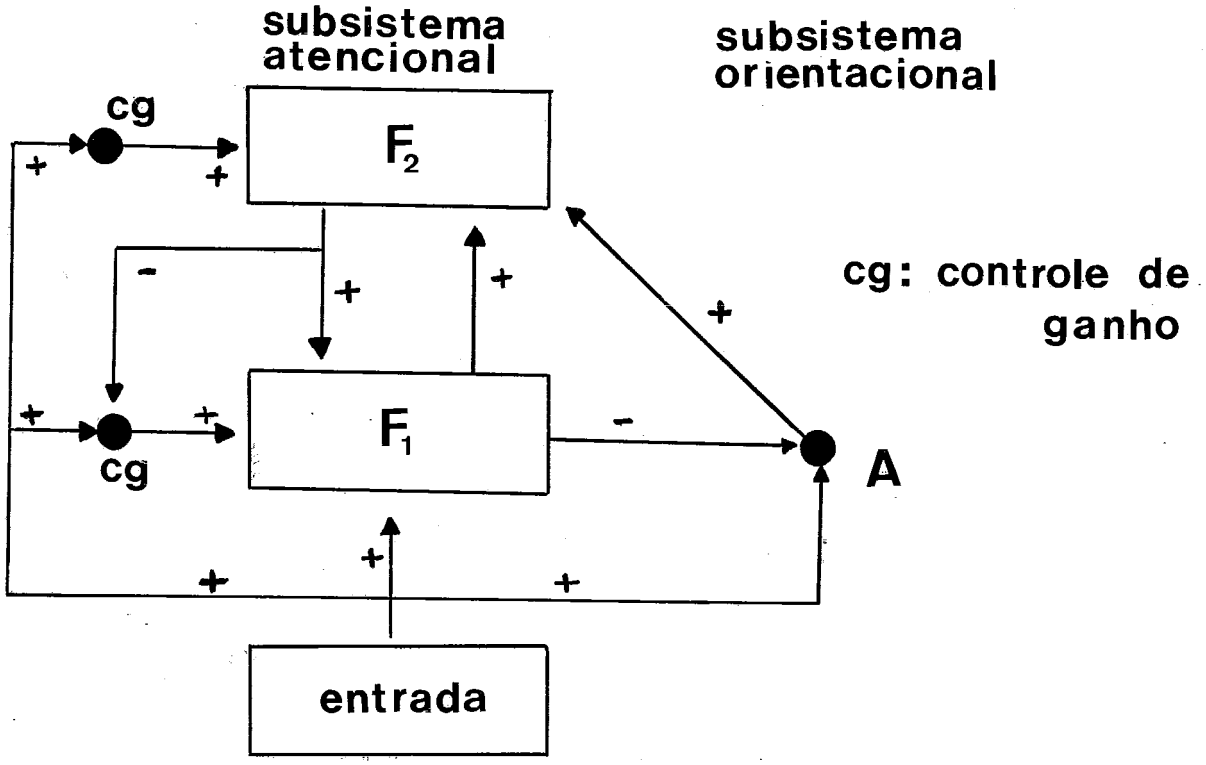

Figura VIII.2: Um módulo ART

VIII.3 O Funcionamento Básico de Um Sistema ART

Para efeitos de explicação, o funcionamento de um sistema ART pode ser dividido nas seguintes fases:

1-*Filtragem adaptativa "bottom-up" e ativação de categoria.* Este é o termo empregado por Carpenter e Grossberg para designar a categorização realizada em F_2 após a apresentação de um estímulo de entrada I (designaremos por I o padrão de entrada). Os padrões de entrada devem ser binários. O funcionamento é análogo ao de um sistema competitivo. As unidades de F_2 competem entre si e a que recebe o maior estímulo de entrada possuirá valor de ativação 1 e as demais ativação nula. O aprendizado difere do paradigma competitivo e será discutido a seguir.

2-*Comparação com o padrão "top-down".* A unidade de categoria ativada de F_2 manda um sinal de "feedback" (chamado de *"template top-down"* ou *expectativa aprendida*) para F_1. Este padrão de feedback corresponde ao protótipo (ou representação) da categoria ativada, o qual a rede irá comparar com o padrão I para verificar se a categorização foi de fato adequada. (O padrão de feedback é exatamente o vetor peso da categoria vencedora pois somente ela está ativada em F_2 — o padrão de feedback será então $1(w_{j1}, \ldots, w_{jn})$, onde n é o número de unidades de F_1.) As unidades de F_1 são agora perturbadas pela entrada I e pelo padrão de feedback. Isto gera em F_1 um padrão X^*, o qual normalmente difere do padrão X induzido somente por I (no presente contexto podemos considerar $X = I$). A rede realiza, então, uma comparação dos padrões top-down e I. A discrepância entre X e X^* determina o funcionamento futuro da rede.

3-*Interações entre os subsistemas atencional e orientacional.* O subsistema orientacional possui um mecanismo de "arousal", **A**, que é ativado no caso do padrão de feedback ser diferente de I. Por isto, **A** possui ligações de F_1 e da entrada; pode assim, comparar X e X^*. A eventual diferença entre os padrões X e X^* atenua a ativação em F_1, diminuindo o sinal inibitório $F_1 \rightarrow \mathbf{A}$; ver Figura (VIII.2). No caso da discrepância ser significativa, **A** dispara excitando não especificamente

(i.e., todas) as unidades de F_2. Mais precisamente, **A** dispara quando recebe uma excitação da entrada maior do que a inibição de F_1. Um funcionamento especial das unidades de F_2 permite que apenas a unidade ativa de F_2 seja inibida; esta inibição deve ser duradoura. A inibição em F_2 elimina o sinal top-down, permitindo que I restaure o padrão X em F_1. A entrada não pode, agora, criar a mesma ativação (categorização) em F_2 devido à inibição duradoura da unidade de categoria previamente excitada. Uma nova categorização se estabelece então, e F_2 gera mais uma vez seu sinal top-down. O processo todo de comparação se repete e no caso de discrepância **A** novamente dispara. O processo de comparação é chamado teste de hipótese já que a rede está verificando se a sua tentativa (hipótese) de categorização está correta.

Desta forma, pode ocorrer uma série de comparações em F_1 e disparos de **A**; este processo é denominado *busca* de categoria. Cada disparo inibe a categoria ativada de F_2; como as inibições são duradouras elas são impossibilitadas de recategorizar a entrada — i.e., elas não mais participam da competição. O ciclo de comparações termina quando a discrepância entre X e X^* não é significativa — i.e., menor do que um valor especificado pelo nível de *vigilância atencional.* Neste caso, **A** não dispara e a rede entra em um estado de *ressonância*; os sinais de $F_1 \rightarrow F_2$ e de $F_2 \rightarrow F_1$ fluem na rede se reforçando, e os pesos das ligações podem ser alterados refinando ainda mais as categorizações da rede. Em particular, o protótipo, ou expectativa aprendida, de F_2 — codificado pelas ligações $F_2 \rightarrow F_1$ — ajusta-se de forma a abarcar o padrão de entrada categorizado.

Uma outra situação pode interromper a série de comparações. Uma unidade de categoria de F_2 não utilizada, ou seja, que não sofreu prévio aprendizado, é recrutada. Neste caso, uma nova categoria é formada com a alteração das ligações $F_1 \rightarrow F_2$ e $F_2 \rightarrow F_1$ (esta última alteração é denominada aprendizado de "template"). Na prática as duas situações de término de busca são tratadas da mesma forma (ver seção VIII.7).

Quando não há unidades a serem recrutadas e o nível de vigilância atencional não permite a categorização de um estímulo de entrada, este é simplesmente recusado pela rede; ele não será tratado. Em outras palavras, padrões de

entrada que não se adequam às categorias pré-estabelecidas, ou que são apresentados pela primeira vez após a utilização plena dos recursos (unidades) da rede, são rejeitados pelos *mecanismos de estabilização* da rede.

VIII.4 A Regra 2/3

Um sistema ART mais geral não possui apenas as camadas F_1 e F_2; pode possuir n camadas. É possível, então, que a camada F_2 seja excitada mesmo que F_1 não possua ativação (esta excitação viria, e.g., de F_3). Esta ativação em F_2 pode excitar F_1. Entretanto, segundo CARPENTER e GROSSBERG (1987a) é preciso diferenciar entre uma excitação top-down ("priming") e uma entrada externa em F_1. Cria-se, então, um mecanismo auxiliar adicional, chamado *controle de ganho atencional*, para distinguir entre estes dois casos. A distinção adequada é obtida fazendo-se uso do controle de ganho diretamente ligado a F_1; ver Figura (VIII.2). No caso em que F_1 recebe estímulos de entrada e top-down, os sinais se cancelam no controle de ganho e este não excita F_1. Haverá, então, duas fontes excitando F_1: o sinal externo e o padrão top-down. Quando há apenas o sinal top-down, é fácil ver que apenas uma fonte excitará F_1.

O comportamento adequado do sistema, ou seja, aquele que permite diferenciar sinais top-down de estímulos externos, pode ser entendido através da arquitetura do sistema e da Regra 2/3, a qual diz que *uma determinada unidade de F_1 só dispara no caso de pelo menos duas dentre três fontes de sinais ativarem esta unidade.* Isto implica, imediatamente, que F_1 não gerará sinais com apenas estímulos top-down, mas emitirá sinais em resposta a estímulos externos.

Como a Regra 2/3 rege o funcionamento de F_1, será ela que comandará o processo de comparação nesta camada. Na comparação dos padrões top-down e de entrada em F_1, o controle de ganho é bloqueado (já que as duas camadas mandam sinais que se neutralizam) e uma unidade só fica ativa se for excitada tanto pela entrada quanto por F_2. É exatamente este mecanismo que faz que discrepâncias diminuam a ativação em F_1 (e eventualmente **A** dispare). Intuitivamente, este comportamento diz que uma unidade de F_1 só permanece ativa se

a expectativa aprendida "concorda" com o padrão de entrada — i.e., a hipótese é confirmada.

VIII.5 O Parâmetro de Vigilância

O nível de vigilância da rede, determinado pelo *parâmetro de vigilância* ρ, define as categorizações realizadas pela rede. Em resposta a um mesmo conjunto de estímulos, uma rede pode adotar categorizações de natureza diversa, dependendo do valor de ρ (esta situação é distinta das variações de categorização obtidas no aprendizado competitivo devido às diferentes ordens de apresentação dos padrões de entrada).

Considere a arquitetura ART exibida na Figura (VIII.2). **A** recebe um sinal excitatório da entrada $P|\mathbf{I}|$; onde $|\mathbf{I}|$ é o número de unidades ativas do padrão de entrada I e P é a excitação transmitida por cada unidade. Ao mesmo tempo, **A** recebe um sinal inibitório de F_1, $Q|\mathbf{X}|$; onde $|\mathbf{X}|$ é o número de unidades ativas em F_1 e Q é o valor da inibição de cada unidade. Salvo dito em contrário, X será considerado o padrão presente em F_1 em um instante qualquer. Sabemos que quando **A** recebe uma excitação maior do que a sua inibição, **A** libera um sinal para F_2. Isto ocorre justamente quando

$$P|\mathbf{I}| > Q|\mathbf{X}|.$$

O parâmetro de vigilância é dado por

$$\rho \equiv \frac{P}{Q}$$

e **A** dispara quando

$$\frac{|\mathbf{X}|}{|\mathbf{I}|} < \rho;$$

caso contrário **A** não libera sinal.

Quando F_2 está inativo $|\mathbf{X}| = |\mathbf{I}|$. Neste caso é preciso garantir que **A** não liberará sinal para permitir a categorização correta de I. Isto é obtido se $P \leq Q$, o que estabelece que

$$\rho \leq 1.$$

Podemos entender intuitivamente o papel do parâmetro de vigilância se observarmos que $|\mathbf{X}|/|\mathbf{I}|$ dá exatamente a fração do padrão X que coincide com o padrão I (já que os padrões são binários). O valor de ρ determina, então, o grau de casamento (similaridade) entre estes padrões para prevenir o sinal de **A**. Lembre-se que sempre $|\mathbf{X}| \leq |\mathbf{I}|$ já que o padrão X é regido pela Regra 2/3.

Como veremos no capítulo 9 ao testar o aprendizado em sistemas ART, valores de ρ distintos geram categorizações distintas. Valores de ρ próximos a 1 acarretam categorizações bastante finas — apenas padrões muito similares são tidos como da mesma categoria —, enquanto que valores menores fornecem categorizações mais grosseiras.

VIII.6 Formalização do Aprendizado em ART

Após a descrição informal do funcionamento do sistema é preciso especificar como o comportamento é obtido formalmente. Com estamos mais interessados nas questões de aprendizado abordaremos apenas os aspectos relacionados a este tópico. Entretanto, todo o comportamento de um sistema ART obedece a um conjunto de equações que o especifica completamente.

As conexões bottom-up e top-down, por razões a serem posteriormente discutidas, são alteradas de forma distinta. Vejamos a seguir cada um destes casos.

VIII.6.1 Adaptação Top-Down

Formalmente as conexões top-down ($F_2 \rightarrow F_1$) são alteradas segundo a equação

$$\frac{dw_{ji}}{dt} = o_j(-w_{ji} + o_i), \tag{VIII.1}$$

onde i e j são unidades de F_1 e F_2 respectivamente. Como F_2 é organizado de forma a permitir apenas um vencedor a equação (VIII.1) pode ser descrita da seguinte forma:

$$\frac{dw_{ji}}{dt} = \begin{cases} -w_{ji} + o_i & \text{se } j \text{ vence} \\ 0 & \text{se } j \text{ perde} \end{cases}, \tag{VIII.2}$$

já que $o_j \in \{0,1\}$.

CARPENTER e GROSSBERG (1987a) preferem destacar três casos da equação (VIII.1):

$$\frac{dw_{ji}}{dt} = \begin{cases} -w_{ji}+1 & \text{se } i \text{ e } j \text{ estão ativas} \\ -w_{ji} & \text{se } i \text{ está inativa e } j \text{ ativa .} \\ 0 & \text{se } j \text{ está inativa} \end{cases} \qquad \text{(VIII.3)}$$

O primeiro caso é denominado *regra de aprendizado de template* ($w_{ji} \to 1$). O segundo caso é denominado *regra de decaimento associativo* ($w_{ji} \to 0$), enquanto que no terceiro não há aprendizado. A equação (VIII.3), ou suas equivalentes (VIII.1–2), diz que o template, ou protótipo, de j tenta aprender (copiar) o padrão de ativação em F_1 quando j está ativa (vence). Como sabemos, este comportamento é exatamente o de um "outstar".

Os importantes resultados obtidos por CARPENTER e GROSSBERG (1987a) pressupõem a hipótese do *aprendizado rápido*: as taxas de aprendizado permitem que os pesos obtenham aproximadamente os valores determinados pelos padrões de ativação em cada ciclo de aprendizado. Ou seja, os pesos aproximam-se rapidamente, i.e., em apenas um ciclo, dos valores assintóticos de (VIII.3). Adotando esta hipótese, ao fim de um ciclo de aprendizado a equação (VIII.3) diz que para uma unidade vencedora j de F_2 ocorre

$$w_{ji} \cong \begin{cases} 1 & \text{se } i \in \mathbf{X} \\ 0 & \text{se } i \notin \mathbf{X} \end{cases}, \qquad \text{(VIII.4)}$$

onde $\mathbf{X}$ é o conjunto de unidades ativas de F_1. É preciso lembrar que só ocorre aprendizado quando a rede está em ressonância e que $\mathbf{X}$ é determinado pela Regra 2/3.

A equação (VIII.4) têm conseqüências interessantes para a formação dos protótipos (top-down) $\mathbf{w}_j = (w_{j1}, \ldots, w_{jn})$. Primeiro ela implica que os protótipos serão binários. Segundo, que estes corresponderão à interseção (ou AND lógico) dos padrões (binários) de entrada I categorizados pelo protótipo. Para entender isto, suponha que $\mathbf{w}_j(0) = \mathbf{1}$. Quando o primeiro padrão I ativar a categoria j, após um certo tempo, as unidades ativas em F_1 serão dadas por $\mathbf{I} \cap \mathbf{W}_j$, onde $\mathbf{I}$ é o conjunto de unidades ativas na entrada e $\mathbf{W}_j$ é o conjunto de unidades de F_1 que recebem conexões não nulas de F_2 — isto é basicamente a Regra 2/3 dita de outra

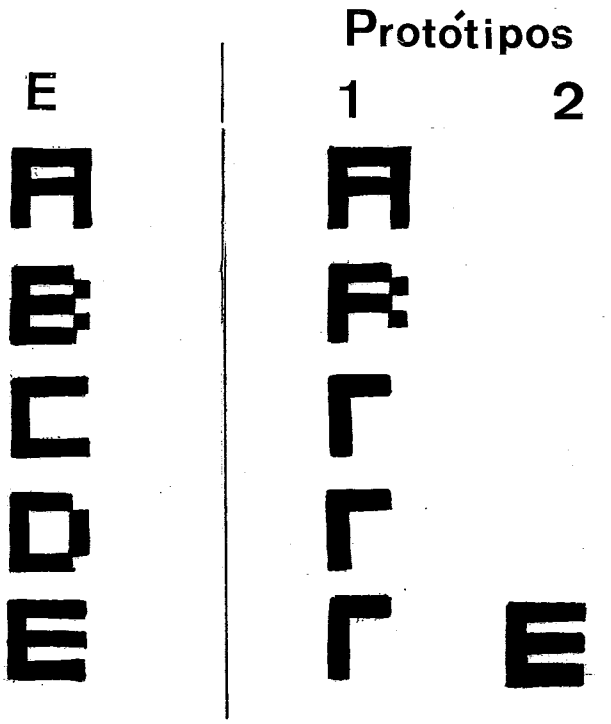

Figura VIII.3: Categorização de alguns padrões. E: padrões de entrada

forma. Observe que $\mathbf{W}_j(0) = \{1, \ldots, n\}$, onde n é o número de unidades de F_1. Com o aprendizado

$$w_{ji} = \begin{cases} 1 & \text{se } i \in \mathbf{I} \cap \mathbf{W}_j \\ 0 & \text{se } i \notin \mathbf{I} \cap \mathbf{W}_j \end{cases}.$$

Repetindo este mesmo raciocínio para vários padrões de entrada I, é fácil ver que $\mathbf{w}_j$ será a interseção dos padrões categorizados (supondo que $\mathbf{w}_j(0) = \mathbf{1}$).

A Figura (VIII.3) tenta ilustrar a evolução dos protótipos ao longo do aprendizado. Por exemplo, quando o padrão B é categorizado pela categoria que previamente codificava A, o protótipo é alterado de para categorizar também B, e isto faz

$$w_{1i} = \begin{cases} 1 & \text{se } i \in \mathbf{A} \cap \mathbf{B} \\ 0 & \text{se } i \notin \mathbf{A} \cap \mathbf{B} \end{cases}.$$

Uma outra maneira de expressar (VIII.4) é através da alteração de pesos, como proposto por LIPPMANN (1987):

$$w_{ji}(t+1) = w_{ji}(t) o_i,$$

onde j é a unidade vencedora. Esta equação de alteração de pesos garante que o protótipo de j, $\mathbf{w}_j$, é alterado de acordo com o padrão de entrada e faz com que

alguns componentes de $\mathbf{w}_j$ passem a valer 0; mais especificamente, $w_{ji}(t+1) = 0$ se $o_i(t) = 0$. Ela será usada quando discutirmos uma possível implementação do aprendizado em sistemas ART.

VIII.6.2 Adaptação Bottom-Up

A equação de alteração de pesos bottom-up é mais complexa do que a regra top-down:

$$\frac{dw_{ij}}{dt} = Ko_j(-E_{ij}w_{ij} + o_i) \tag{VIII.5}$$

onde

$$E_{ij} = o_i + L^{-1}\sum_{k\neq i} o_k, \tag{VIII.6}$$

sendo K e L constantes; $L > 1$; $i \in F_1$ e $j \in F_2$.

As equações (VIII.5–6) implementam a *regra da lei de Weber*, a qual será introduzida intuitivamente agora. Suponha que dois padrões, I_1 e I_2, são codificados por uma rede, e que I_1 é um subconjuto de I_2 — ou seja, I_2 é igual a I_1 em todas as unidades ativas de I_1. No caso de I_1 e I_2 serem suficientemente diferentes, eles devem ser categorizados por unidades distintas de F_2. No entanto, como I_2 é igual a I_1 na interseção destes padrões, e como todas as unidades de F_1 onde I_2 não é igual a I_1 ficam inativas quando I_1 é apresentado, como a rede decide entre as duas categorias quando I_1 é apresentado? Suponha que as unidades c_1 e c_2 de F_2 codificam I_1 e I_2 respectivamente. A Figura (VIII.4) abaixo exibe dois padrões possíveis. Para que a rede se comporte adequadamente, é preciso que c_1 receba um sinal maior do que c_2 quando da apresentação de I_1. Desta forma, os pesos das ligações para c_2 das unidades de I_1 precisam ser menores do que os de I_1 para c_1. Como os pesos de c_2 foram codificados pelo padrão maior I_2, isto sugere que padrões maiores são codificados por pesos menores. Assim, os valores dos pesos para as unidades c_1 e c_2 refletem a escala geral dos padrões codificados por estas unidades. A realização desta relação inversa entre magnitude dos pesos e escala dos padrões de entrada é denominada regra da lei de Weber.

Ao mesmo tempo que o padrão subconjunto I_1 deve ativar sua categoria c_1, o superconjunto I_2 deve ativar a categoria c_2. Como a apresentação de

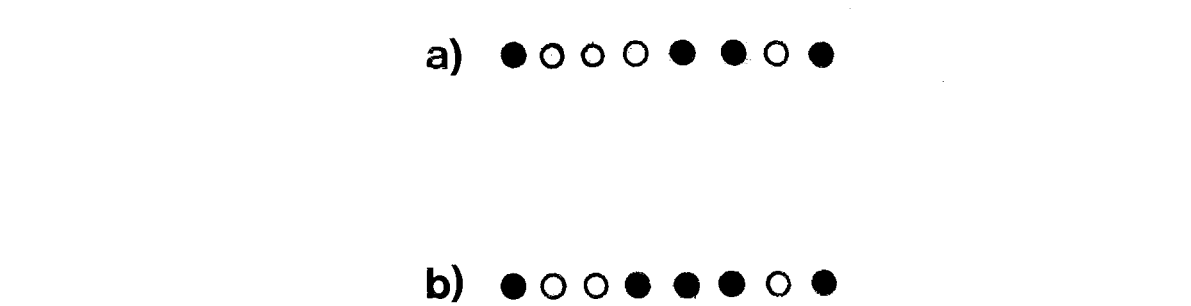

Figura VIII.4: Dois possíveis padrões. a) padrão I_1. b) padrão I_2. Círculos cheios: unidades ativas. Círculos vazios: unidades desativadas

I_2 ativa todo subpadrão I_1, uma outra propriedade é necessária para garantir que a unidade c_1 não seja ativada pelo padrão I_2. Esta propriedade é obtida pela regra do decaimento associativo vista anteriormente, a qual implica que alguns pesos decaem para zero ao longo do aprendizado.

A Figura (VIII.5) tenta esclarecer a situação geral. Durante o aprendizado de I_1 os pesos das ligações oriundas de unidades desativadas de F_1 decaem para zero. Ao mesmo tempo, ligações de unidades ativas são reforçadas; ver Figura (VIII.5a). Durante o aprendizado de I_2, como todas as unidades de F_1 estão ativas, todas as ligações para c_2 são reforçadas. No entanto, pela regra da lei de Weber, os pesos para a unidade c_2 (Figura (VIII.5b)) não crescem tanto quanto os já aprendidos para c_1 (Figura (VIII.5a)), pois I_2 possui mais unidades ativas do que I_1. Por outro lado, após o aprendizado, a quantidade *total* de pesos existente para c_2 é maior do que para c_1. Estas relações entre magnitude de ligações individuais e o número de unidades ativas excitando cada unidade de F_2 permite que I_1 seja codificado por c_1 e I_2 por c_2; ver Figura (VIII.5c–d).

Resumindo, a alteração de pesos bottom-up é determinada pela ação conjunta da regra da lei de Weber e da regra do decaimento associativo para garantir que padrões subconjunto possam ser devidamente categorizados.

É possível mostrar que as equações (VIII.5–6) implementam as duas regras mencionadas acima; ver (CARPENTER e GROSSBERG, 1987a). Reescre-

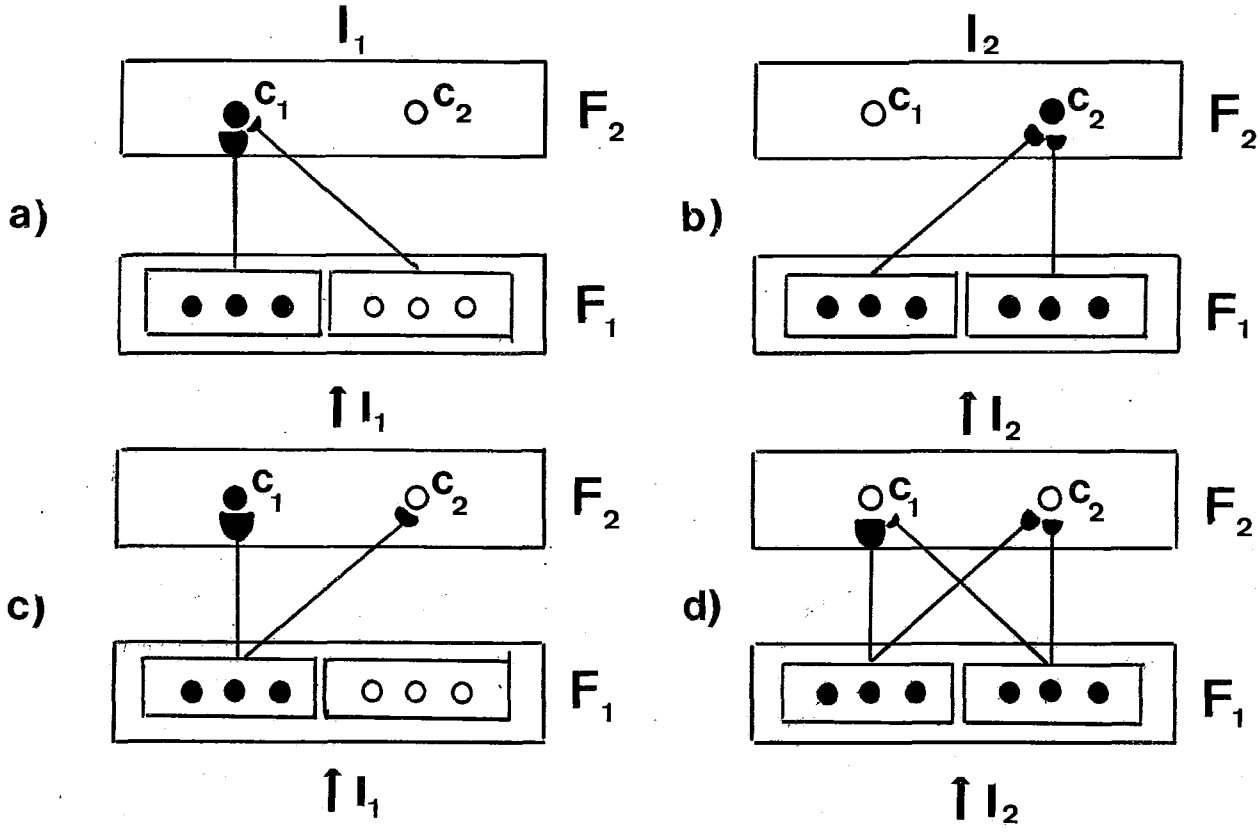

Figura VIII.5: Aprendizado segunda a regra da lei de Weber e do decaimento associativo. a) Aprendizado de I_1. b) Aprendizado de I_2. Após aprendizado: c) Ativação de c_1 com padrão I_1. d) Ativação de c_2 com padrão I_2. Círculos cheios correspondem a unidades ativas

vendo as equações de adaptação bottom-up é possível enxergar estas regras mais facilmente:

$$\frac{dw_{ij}}{dt} = \begin{cases} K\left((1-w_{ij})L - w_{ij}(|\mathbf{X}|-1)\right) & \text{se } i \text{ e } j \text{ estão ativas} \\ -K|\mathbf{X}|w_{ij} & \text{se } i \text{ está inativa e } j \text{ ativa} \\ 0 & \text{se } j \text{ está inativa} \end{cases}, \qquad \text{(VIII.7)}$$

onde K e L são as constantes encontradas em (VIII.5–6).

O primeiro caso da equação (VIII.7) corresponde à regra da lei de Weber. No equilíbrio, ou seja, $dw_{ij}/dt = 0$, este caso implica que

$$w_{ij} = \frac{\alpha}{\beta + |\mathbf{X}|}, \qquad \text{(VIII.8)}$$

com $\alpha = L$, $\beta = L - 1$ e $|\mathbf{X}|$ é o número de unidades ativas em F_1. Por (VIII.8), valores grandes de $|\mathbf{X}|$ implicam em pesos menores nas ligações codificando $|\mathbf{X}|$, o que é justamente o necessário para a regra da lei de Weber. O segundo caso de (VIII.7) é exatamente a regra do decaimento associativo — decaimento exponencial para zero. E no último caso não há aprendizado.

Os pesos bottom-up também são atualizados segundo a hipótese do aprendizado rápido. Assim, a equação (VIII.7) implica (CARPENTER e GROSSBERG, 1987a) que ao final de um ciclo de aprendizado

$$w_{ij} \cong \begin{cases} \frac{L}{L-1+|\mathbf{X}|} & \text{se } i \in \mathbf{X} \\ 0 & \text{se } i \notin \mathbf{X} \end{cases}, \qquad \text{(VIII.9)}$$

onde $L > 1$, e $\mathbf{X}$ é o conjunto de unidades ativas de F_1.

LIPPMANN (1987) sugere que podemos usar a seguinte regra para implementar a adaptação bottom-up:

$$w_{ik}(t+1) = \frac{w_{ki}(t)o_i}{0.5 + \sum_i w_{ki}(t)o_i} \qquad \text{(VIII.10)}$$

onde k é a unidade vencedora de F_2 e $i \in F_1$. Em (VIII.9), $\mathbf{X}$ é determinado pela Regra 2/3, e é exatamente isto que a equação (VIII.10) tenta implementar. Pela Regra 2/3 uma unidade de F_1 só ficará ativa se ela receber estímulo externo ($o_i = 1$) e da camada F_2 ($o_k = 1$ e $w_{ki} = 1$). O numerador em (VIII.10), $w_{ki}(t)o_i$, garante este comportamento já que $o_k = 1$ pois k é a unidade vencedora. Observe, também, que $\sum_i w_{ki}(t)o_i$ corresponde ao termo $|\mathbf{X}|$ de (VIII.9) e que $w_{ki}(t)o_i$ e 0.5 de (VIII.10) aproximam L e $L-1$, respectivamente.

VIII.6.3 Valores Iniciais dos Pesos

Considerações técnicas (CARPENTER e GROSSBERG, 1987a) mostram que os valores dos pesos da rede — tanto bottom-up quanto top-down — precisam ser adequadamente inicializados para não corromper o seu funcionamento.

Para garantir que padrões perfeitamente aprendidos tenham *acesso direto* às unidades de F_2 que os codificam — i.e., não haja busca —, é preciso que os pesos bottom-up obedeçam a *desigualdade do acesso direto*:

$$0 < w_{ij}(0) < \frac{L}{L-1+M} \tag{VIII.11}$$

onde $L > 1$ e M é o número de unidades de F_1. A violação desta desigualdade pode fazer com que todas as unidades de F_2 codifiquem o mesmo padrão de entrada. Isto é evitado fazendo-se com que os pesos iniciais $w_{ij}(0)$ não sejam muito grandes.

Já os pesos top-down iniciais não podem ser muito pequenos. Esta condição decorre diretamente da Regra 2/3, a qual implica que se os pesos iniciais $w_{ji}(0)$ fossem muito pequenos, então as unidades de F_2 não poderiam aprender padrões, uma vez que a atividade de F_1 seria eliminada sem a excitação extra proveniente de F_2. Pode-se mostrar, então, que a seguinte *desigualdade do aprendizado de template* deve ser obedecida:

$$1 \geq w_{ji}(0) > \frac{B_1 - D_1}{D_1}, \tag{VIII.12}$$

sendo que

$$\max\{1, D_1\} < B_1 < 1 + D_1.$$

VIII.7 Uma Possível Implementação do Aprendizado em Sistemas ART

LIPPMANN (1987) sugere uma possível implementação dos mecanismos de aprendizado do sistema ART. A seguir descreveremos o algoritmo em si (com algumas pequenas alterações).

- Aprendizado em ART

– **Passo 1. Inicialização**

Inicializar pesos top-down e bottom-up (respeitando as desigualdades (VIII.11–12))

$$w_{ji}(0) = 1$$

$$w_{ij}(0) = \frac{1}{1+M},$$

onde $i \in F_1$, $j \in F_2$ e M é o número de unidades de F_1. Inicializar também o parâmetro de vigilância ρ.

– **Passo 2. Entrada**

Um novo padrão binário de entrada é apresentado à rede.

– **Passo 3. Competição**

Selecionar unidade k de F_2 tal que

$$net_k = \max_j \{net_j\}.$$

– **Passo 4. Teste de Vigilância**

$$|\mathbf{I}| = \sum_i o_i$$

$$|\mathbf{X}| = \sum_i w_{ki} o_i$$

onde k é a unidade vencedora do Passo 3 e $i \in F_1$. Se

$$\frac{|\mathbf{X}|}{|\mathbf{I}|} \geq \rho,$$

então passou no teste de vigilância e vai para o Passo 6. Caso contrário vai para o Passo 5.

– **Passo 5. Inibição Duradoura**

o_k, onde k é a unidade do Passo 3, passa a valer zero, e k não mais participa do passo de competição (este comportamento corresponde à inibição duradoura da categoria de F_2 devido ao disparo de **A**). Vá para o Passo 3 (i.e., busca categoria).

– **Passo 6. Adaptação**

Alterar os pesos top-down segundo:

$$w_{ki}(t+1) = w_{ki}(t) o_i.$$

Alterar os pesos bottom-up segundo:

$$w_{ik}(t+1) = \frac{w_{ki}(t)o_i}{0.5 + \sum_i w_{ki}(t)o_i},$$

sendo k a unidade vencedora, $i \in F_1$ e $j \in F_2$.

– **Passo 7. Loop**

Reconsidere todas as unidades inibidas no Passo 5. Vá para o Passo 2.

Observando esta versão do aprendizado em sistemas ART podemos notar a sua semelhança com o algoritmo de conglomeração descrito ao final do capítulo 3.

VIII.8 Principais Características dos Sistemas ART

Os sistemas ART são bastante sofisticados e possuem: (a) Mecanismos bottom-up encontrados em modelos de aprendizado competitivo, os quais tentam categorizar adequadamente os estímulos de entrada; (b) Processos top-down que aprendem expectativas ou protótipos das categorias; (c) Processos para a comparação de padrões bottom-up (entrada) e top-down (expectativa aprendida); e (d) Processos para liberar sinais para F_2 em caso de discrepância de padrões — i.e., erro de categorização.

Os sistemas ART resolvem o dilema da estabilidade-plasticidade. Intuitivamente, isto é obtido fazendo com que o sistema alterne entre dois modos: Estável e Plástico. As interações entre dois subsistemas garantem este comportamento:

-*Subsistema Atencional.* É composto por F_1, F_2 e suas interligações, incluindo também os controles de ganho. Processa eventos familiares estabelecendo representações mais e mais precisas destes eventos.

-*Subsistema Orientacional.* É composto por **A** e suas ligações. Libera sinal para o subsistema atencional na ocorrência de um evento não familiar ou não esperado.

São as interações entre estes dois subsistemas que determinam quais eventos são familiares, e quais serão as categorizações realizadas pela rede.

Apesar da estabilidade do aprendizado em sistemas ART ser garantida formalmente (ver o Teorema do Aprendizado Estável de Categorias em (CARPENTER e GROSSBERG, 1987a)), uma outra maneira intuitiva de constatar este fato é observar que as alterações nos pesos só ocorrem em uma direção. Como vimos, um determinado protótipo (top-down) $\mathbf{w}_j$ só perde componentes (alguns w_{ji} passam a valer 0) um número finito de vezes. Além disso, é possível mostrar (ver CARPENTER e GROSSBERG, 1987a) que isto implica que as alterações nos pesos bottom-up são monotônicas (i.e., os pesos sempre crescem ou valem sempre zero). Desta forma, o aprendizado é estável. A estabilidade está, então, diretamente ligada à hipótese do aprendizado rápido. De fato, RYAN e WINTER (1987) demonstraram instabilidade quando esta hipótese é violada. Mais recentemente, alguns trabalhos têm estudado o aprendizado lento em sistemas ART (CARPENTER e GROSSBERG, 1990).

VIII.9 Propriedades Importantes dos Sistemas ART

Discutiremos, a seguir, algumas propriedades dos sistemas ART.

-*Auto-Escala.* Suponha que os dois padrões da Figura (VIII.6) são fornecidos a uma rede. Apesar destes padrões diferirem em apenas uma posição, esta diferença é significativa já que os padrões são pequenos. Desta forma, seria interessante que a rede classificasse estes padrões em categorias distintas. Observe que a Regra 2/3, acoplada ao parâmetro de vigilância ρ, permite este tipo de funcionamento. Se o padrão B ativar a mesma categoria, previamente estabelecida, do padrão A, então, supondo que o protótipo top-down é exatemente o padrão A, pela Regra 2/3, haverá disparo de **A** apenas se

$$\frac{|\mathbf{X}|}{|\mathbf{I}|} = \frac{2}{3} < \rho.$$

Como 2/3 é um valor relativamente baixo para ρ, a rede tipicamente não aceitará esta categorização — i.e., B será colocado em outra categoria. No caso de padrões grandes, maiores discrepâncias podem ser toleradas já que

$$\frac{|\mathbf{X}|}{|\mathbf{I}|} = \frac{|\mathbf{I}| - n}{|\mathbf{I}|} \rightarrow 1$$

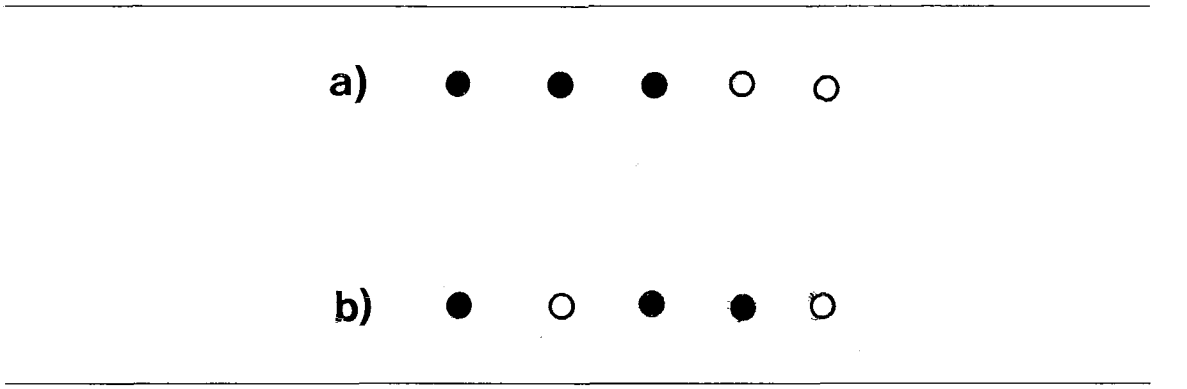

Figura VIII.6: Dois padroões a serem categorizados. a) Padrão A. b) Padrão B

onde n é o número de posições de discrepância.

Esta propriedade é extremamente importante pois fornece um meio de distinguir *sinal* de *ruído*. O contexto do padrão — correspondente ao valor $|\mathbf{I}|$ — é usado para determinar que características são informativas (sinal) e quais constituem ruído. No caso da categorização dos padrões da Figura (VIII.6), a diferença entre A e B foi considerada como sinal. Entretanto, em outro contexto — em padrões maiores — a mesma discrepância seria tratada como ruído e os padrões correspondentes seriam classificados pela mesma categoria.

-*Acesso direto a códigos (ou categorias) aprendidos.* O processo de busca (de categoria) regido pelas interações entre os sistemas atencional e orientacional, é bastante útil na procura de uma categoria adequada para um estímulo de entrada novo. Entretanto, esta busca torna-se altamente indesejável se a apresentação repetida de um particular estímulo aciona-a seguidamente. Felizmente, após uma fase de consolidação de categoria, padrões familiares (i.e., aprendidos) *acessam diretamente*, ou seja, ativam diretamente, sua categoria. Além disso, um padrão não aprendido pode acessar diretamente a sua categoria, sendo preciso para tanto que ele seja suficientemente parecido (no sentido do parâmetro de vigilância) com o protótipo de uma categoria estabelecida.

Aqueles padrões não aprendidos incapazes de acessar diretamente a sua categoria, ativarão uma busca para encontrar uma categoria adequada. Após o estabelecimento desta nova categoria estes padrões não mais gerarão uma busca.

-*Ambiente como professor: Modulação de vigilância atencional.* Ape-

sar de um sistema ART estabelecer suas categorias de uma forma não-supervisionada, a existência do parâmetro de vigilância permite que o ambiente onde a rede existe possua um papel modulatório sobre o processo de aprendizado. Este fato permite que uma rede com um conjunto fixo de categorias funcione satisfatoriamente em ambientes diversos — diferentes ambientes podem exigir discriminações mais grosseiras ou mais finas do mesmo conjunto de objetos.

Pode-se pensar que o parâmetro de vigilância está diretamente ligado a um sinal de reforço externo. No caso de categorizações errôneas, um reforço negativo é fornecido à rede, o qual faz o parâmetro de vigilância aumentar. Assim, a rede torna-se mais *vigilante*: mais sensível a discrepâncias nas suas categorizações, realizando categorizações mais finas.

Na configuração básica do sistema ART este comportamento não acontece. Para obtê-lo é preciso alterar o parâmetro de vigilância de acordo com reforços do ambiente.

VIII.10 Algumas Aplicações de Sistemas ART

A verdadeira importância do módulo ART exibido na Figura (VIII.2) é que ele constitui um módulo de categorização bastante efetivo. Suas propriedades de categorização estável, acesso direto a categorias aprendidas, modulação da categorização através do parâmetro de vigilância, dentre outras, o tornam um sistema poderoso. Observe que estas propriedades são formalmente, ou matematicamente, demonstráveis (ver CARPENTER e GROSSBERG, 1987a), garantindo o funcionamento do sistema.

Os sistemas ART são usados em muitos modelos psicológicos/biológicos, tais como a) Condicionamento Pavloviano (GROSSBERG e SCHMAJUK, 1987); b) Percepção visual; c) Reconhecimento de palavras (GROSSBERG e STONE, 1986), dentre outros. Outra aplicação bastante importante é o reconhecimento de padrões. Em todos estes casos o modelo geral inclui um ou mais módulos ART acoplados aos outros elementos do sistema. Por exemplo, no reconhecimento de

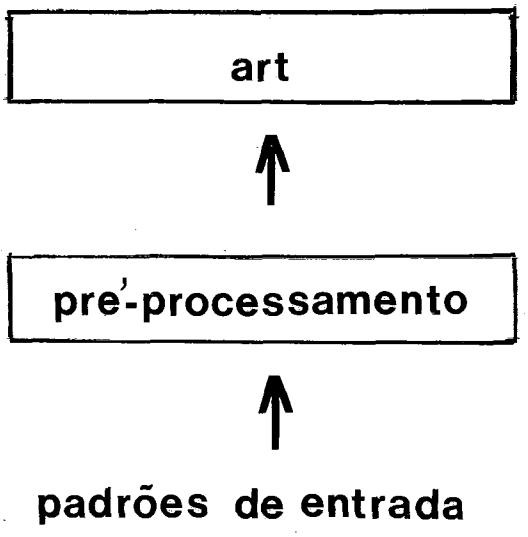

Figura VIII.7: Módulo ART acoplado a um módulo pré-processador

padrões, o módulo ART realiza as categorizações após um módulo pré-processador ter tornado os estímulos de entrada invariantes a translação, escala e rotação; ver Figura (VIII.7).

Uma limitação óbvia dos sistemas ART 1 é que eles só lidam com padrões binários, o que diminui sensivelmente o número de aplicações possíveis. A limitação binária é literalmente construída na estrutura e operação do sistema (veja, e.g., a Regra 2/3).

É possível combinar vários módulos ART 1 (e.g., dois módulos podem implementar uma memória associativa; ver CARPENTER, 1989), ou módulos ART 1 e ART 2 (ver CARPENTER e GROSSBERG, 1988) obtendo configurações bastante interessantes. Recentemente, vem-se estudando a construção modular de redes neuronais para sistemas grandes (e mais realistas), e, como mencionado, os sistemas ART apresentam diversas facilidades neste sentido.

Capítulo IX

Avaliação dos Algoritmos de Aprendizado

IX.1 Introdução

Neste capítulo tentaremos avaliar os algoritmos de aprendizado não-supervisionados estudados nesta tese. Talvez a avaliação mais importante a ser feita seja teórica. Entretanto, informações bastante interessantes podem ser obtidas de avaliações mais experimentais. As análises a serem realizadas a seguir são estritamente experimentais; análises teóricas importantes foram citadas ao longo da tese.

Muitos experimentos encontrados na literatura com os algoritmos estudados são bastante complexos, por vezes envolvendo outras formas de (pré-) processamento (e.g., experimentos em reconhecimento de imagens e de fala). Outros são por demais diretos; ver (MCCLELLAND e RUMELHART, 1988). Escolhemos, então, um experimento intermediário cuja formulação é bastante simples mas cujo resultado não é imediatemente previsível: O reconhecimento de letras. A rede é estimulada com padrões binários, que correspondem às letras, e deve aprender a categorizá-los corretamente. Na situação ideal, seria interessante que a rede descobrisse 26 categorias e que a partir deste instante cada letra apresentada ativasse a sua categoria adequada. Foram realizados alguns experimentos com dois tipos (formatos) de letras diferentes.

Os resultados a serem descritos a seguir foram obtidos utilizando-se

um simulador de redes neuronais desenvolvido especificamente para implementar os algoritmos de aprendizado em questão.

IX.2 Experimento 1

Neste primeiro experimento escolheu-se um tipo de letra bastante peculiar, onde as letras são bastante parecidas e algumas são subconjuntos de outras (e.g., F é um subconjunto de A). A Figura (IX.1) exibe as vinte e seis letras; as letras possuem 5x5 pontos; há, portanto, 25 unidades de entrada. A seguir descrevemos os principais resultados obtidos nos testes com os três algoritmos.

Aprendizado Competitivo. Esta forma de aprendizado revelou-se bastante ineficaz para resolver a categorização em questão. Com relação ao Experimento 1 podemos dizer, em resumo, que: a) A capacidade de categorização é baixa (nos testes realizados estabeleceu entre 5 e 12 categorias); b) É bastante instável; e c) Leva na ordem de milhares de iterações para obter seu resultado (este critério é, reconhecidamente, subjetivo, já que o algoritmo não resolve o problema).

Alguns testes representativos revelam estes resultados. Na Tabela (IX.1) vemos uma categorização típica. A rede descobriu 9 categorias, as quais mostram bastante bem as similaridades entre os padrões de entrada. Por exemplo, os padrões A, F, P, e R tendem, em muitas simulações, a serem agrupados na mesma categoria. Isto também ocorre com os padrões das demais categorias da Tabela (IX.1). Entretanto, outras categorizações distintas são também possíveis, como a da Tabela (IX.2), ou a da Tabela (IX.3).

Em geral constantes de aprendizado menores tendem a obter menos categorias do que constantes maiores (compare as Tabelas (IX.2) e (IX.3), por exemplo).

Quanto à instabilidade, tanto valores altos quanto valores mais baixos da constante de aprendizado exibem tal comportamento. No entanto, constantes mais altas tendem a exibir instabilidades mais "violentas" (como seria de se esperar). Compare a oscilação exibida na Tabela (IX.4) com a da Tabela (IX.5).

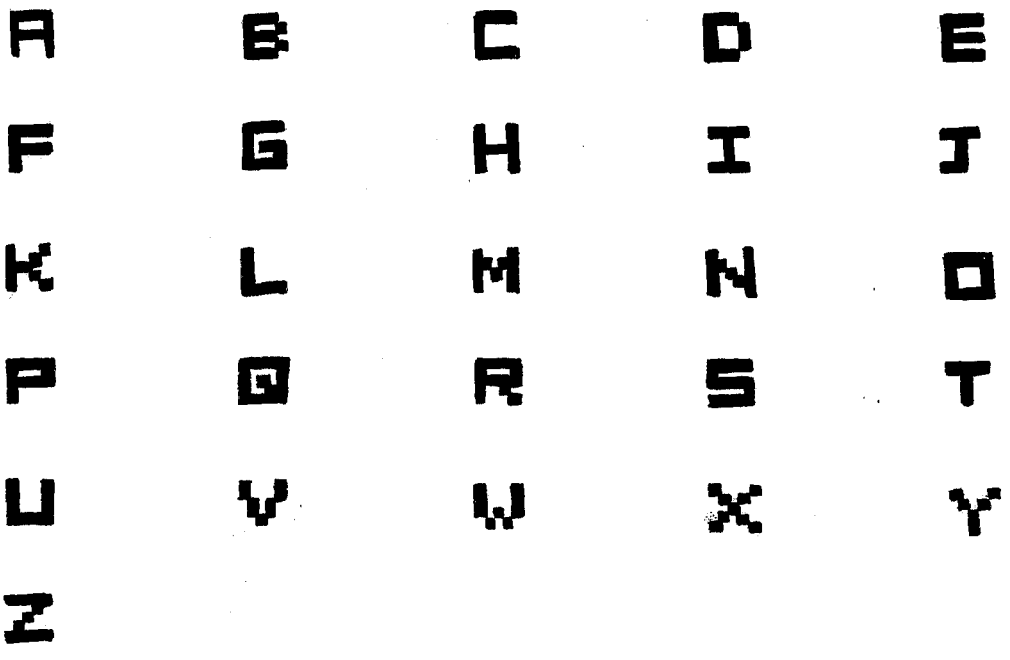

Figura IX.1: Conjunto de letras utilizado no Experimento 1

nº unid. saída	26
const. aprendizado	0.8
ciclo	10000
categorias detectadas	9
categoria	**elementos**
1	E, G, S
2	K, X
3	Q
4	U, V, W
5	B
6	A, F, P, R
7	I, J, T, Y, Z
8	C, D, L, O
9	H, M, N

Tabela IX.1:

nº unid. saída	26
const. aprendizado	0.8
ciclo	9000
categorias detectadas	10
categoria	**elementos**
1	E
2	C, D, O
3	G
4	L, M, N U, V, W
5	A, F, H, K, P, R
6	B
7	X, Z
8	S
9	I, J, T, Z
10	Q

Tabela IX.2:

nº unid. saída	26
const. aprendizado	0.1
ciclo	15000
categorias detectadas	5
categoria	**elementos**
1	I, J, T, Y, Z
2	A, F, P, R
3	C, D, L, O, Q, U, W
4	B, E, G, S
5	H, K, M, N, V, X

Tabela IX.3:

ciclo	7000	8000
categoria	elementos	elementos
1	W	A, F, P, R
2	K, X, Y	-
3	G	D, L, O
4	I, J, T, Z	E, G
5	F, P, R	J, T, X, Y
6	-	U, V, W
7	B	B
8	V	H, K, M, N
9	E	-
10	S	S
11	A, H, M, N	C, I, Z
12	Q	Q
13	C, D, L, O, U	-

Tabela IX.4: Constante de aprendizado igual a 0.8

ciclo	5000	9000
categoria	elementos	elementos
1	A, F, P, R	A, F, H, P, R
2	I, J, T, Y, Z	I, J, T, Y, Z
3	B	B
4	C, D, G, O	C, D, G, L, O
5	Q	Q
6	E, S	E,S
7	L, U, V, W	U, V, W
8	H, K, M, N, X	H, M, N, X

Tabela IX.5: Constante de aprendizado igual a 0.2

Alguns testes com a camada de saída com 80 unidades não mostraram comportamento diverso do descrito acima. Um maior número de unidades de categoria, supostamente, permitiria o estabelecimento de mais categorias já que os vetores peso poderiam estar mais próximos dos padrões.

O aprendizado competitivo é capaz de descobrir as similaridades entre os padrões de entrada. Entretanto, as suas distinções são, aparentemente, por demais grosseiras; muitas letras são colocadas na mesma categoria. Além disso, a instabilidade das categorizações é bastante acentuada.

O Experimento 1 revela que o comportamento do algoritmo (básico) nesta aplicação não é tão ideal quanto o mencionado por RUMELHART e ZIPSER (1985); ver seção V.4. Em particular, uma rede com 26 unidades de categoria não consegue classificar os padrões em 26 categorias.

A questão da instabilidade do algoritmo merece ser discutida do ponto de vista prático. Não é possível estabelecer se uma categorização inicialmente instável irá, no futuro, estabilizar-se. No entanto, podemos detectar estabilidade caso isto ocorra. Em muitas simulações do aprendizado competitivo utilizamos um teste de estabilidade, o qual não detectou tal estado em nenhum teste (foram empregadas inúmeras constantes de aprendizado). O seguinte algoritmo indica, em alto nível, como isto foi testado.

- Aprendizado Competitivo com Teste de Estabilidade

  - **Passo 1. Inicialização**

    Faça as inicializações do aprendizado competitivo básico.

    Faça $C_i(0) = \phi,\ i = 1, \ldots, n$ (n é o número de categorias)

  - **Passo 2. Entrada**

    O padrão S_k é apresentado à rede

  - **Passo 3. Competição**

    Determina a unidades vencedora j

  - **Passo 4. Adaptação**

    Altera o vetor $\mathbf{w}_j$

  - **Passo 5. Limpa Conjuntos**

    Faça $C_i(t+1) = C_i(t) - S_k,\ i = 1 \ldots, n$

  - **Passo 6. Atualiza Conjuntos**

    Se o vetor peso $\mathbf{w}_j$ foi alterado pelo padrão S_k então

    $$C_j(t+1) = C_j(t) \cup \{S_k\}$$

    Senão

    $$C_j(t+1) = C_j(t) - \{S_k\}$$

    onde j é a unidade vencedora.

  - **Passo 7. Testa Quase-Estabilidade**

    Se $C_i = \phi,\ i = 1, \ldots, n$ então

    repete Passos 2 a 6 até que todos os padrões tenham sido apresentados

    Senão

    vai para o Passo 2

  - **Passo 8. Testa Estabilidade**

    Se $C_i = \phi,\ i = 1, \ldots, n$ então

    Estabilizou

    Senão

    vai para o Passo 2

Cria-se um conjunto C_i para cada unidade de categoria i para guardar os padrões S_k que alteram seus pesos. Quando nenhum padrão altera mais os pesos, o aprendizado estabilizou. A idéia básica do algoritmo é colocar um padrão S_k em C_j se ele altera w_j, ou retirá-lo se ele não causa alteração, até que todos os C_i sejam vazios. Entretanto, alguns passos adicionais são necessários. O Passo 5 retira eventuais padrões que não ativam mais a categoria j (são codificados por outra unidade), e que, assim, não sairiam mais dos conjuntos através do Passo 6. Se no Passo 7 todos os C_i estão vazios, é preciso verificar (ainda) se a apresentação de algum padrão novo não irá causar recodificações. Se todos os padrões forem apresentados e os C_i continuarem vazios, detectou-se estabilidade.

Auto-organização de mapas de características. O algoritmo de Kohonen obteve bons resultados neste teste.

Na utilização deste algoritmo, tipicamente organizam-se as unidades de saída segundo um arranjo bi-dimensional. Em um dos testes, foi escolhido um arranjo quadrado 13x13. Abaixo exibimos os resultados obtidos. Em instantes de tempo selecionados interrompemos o algoritmo e geramos dois tipos de diagramas: um mostra as posições (no arranjo 13x13) das unidades que respondem mais fortemente aos padrões de entrada (unidades vencedoras), e outro mostra que padrão gera maior ativação em uma unidade de saída específica (mapa). Inicialmente não há organização nenhuma no mapa de saída (ciclo 0). Já no ciclo 100 observa-se uma organização inicial (padrões próximos no mapa são semelhantes). Com o tempo o mapa vai se organizando e mais e mais padrões entram nele até que no ciclo 3000 observa-se a formação de um mapa correto (e completo). No ciclo 5000 o mapa encontra-se mais refinado e sofre apenas uma alteração até o ciclo 6000.

```
--- Ciclo --- 0

.............   RGAQDQQBQQQRQ
........S...Q   QEQSQSQASEMQQ
```

```
......W......   SASQQUQSQDGQQ
.........HL..   QQGBQQSBSHEGN
............    QQSBOEOBBQSGQ
.......OM....   GEQBNQSQMPKEG
............Z   GQERQEQQBQAAQ
............    BVESQMGANNRSS
.U.........J    AEAREEGQRQPRJ
........Y....   QSSAQQQBVQRSR
.N........E..   RQQGSAQBQAEQA
T............   RMQEEBAAQSBGE
....R........   EAGARASREQZQB


--- Ciclo --- 100


TJ.IZ.S.P..RH   TTTIIEEPPPPPA
.....G.......   TTIIOQQPPPRRR
......B......   IICOOOQAARRKK
.....OQ......   CCODDDOQARKKK
C...........K   CODDDDDQQNKKK
....D........   LODDDDDQNNKKK
.........N...   LODDDDDQNNNKK
.........M...   LLDDDDOQNNNNK
L...........X   LLUODOOQMMMMM
............    LUUUUWWUYYYYY
............    UUUUWWWYYYYYY
....W........   UUUWWWWYYYYYY
U....V.....Y.   UUUWWWWYYYYYY


--- Ciclo --- 500
```

```
.T....Y.....P   TTTTYYYYYPPPP
............R   TTTTTYYYYPAAA
............    TTTTTYYYYAAAA
J...........A   JJJJJYYMMHAAA
......XK.....   IIIIIZMMMMHAA
............    IIIIZZMMMMHHA
I..Z........H   IIIIZQMMMMMHH
........M....   IIIZZQMMMMMMH
....C.....N..   IIQQQQMMMMMMH
............    GQQQQQQMMMMMW
S...........    GQQQQQQUMMWWW
G..Q...LV....   QQQQQQQOUWWWW
D.....OU...W.   QQQQQQOOUWWWW
```

--- Ciclo --- 1000

```
T..Y..X.K...P   TTTIXXXXKRRRP
.J..........    TTIIZXXXKRRRR
.I.........R.   IIISSEERRRRRR
............    ZZZSEEEERAARR
Z...S........   ZZGGEEEEAAAAA
...G.E....A..   ZZGGEEEEAAAAA
............    CCCGEEEEAAAAH
.......B.H...   CCCCGEEEAAHMM
..C..........   CCCCCGGGHHMMM
Q.........N.M   OCCOOCLUUMMMM
O.....L......   OOUUUULUUNMMM
............    OUWWWUUUVVVMM
D..W..U..V...   OWWWWUUUVVVVV
```

--- Ciclo --- 3000

```
.Y....X.K..F.   YYYZZZXXKKFFF
....Z........   YYYZZZZXKRFFP
............P   TTZZZZEERRRPP
T.....E..R...   TTIZZEEERRRAA
...I........A   JJIIIEEERRRAA
.J...........   JJIIISSSBBAAA
......S..B...   JJIGGGSSBBBHH
....G.......H   CCGGGGGBBBBHH
C............   CCOOGQQQNNNMH
...O..Q..N...   CCOOOQQQNNNMM
............M   COOOOQQQNNNMM
.............   LUUUUDDDWWWVV
L..U..D..W..V   LLUUUDDDWWWVV
```

--- Ciclo --- 5000

```
Y....X..K..F.   YYZZXXXKKKFFF
...Z.........   YYZZZXXKKKFFP
............P   TTZZZEEERRRPP
T.....E..R...   TTIIIEEERRRAP
...I.......A.   TTIIIEEERRAAA
.............   JJIIISSSBBBAH
J.....S..B..H   JJGGGSSSBBBHH
...G.........   JJGGGSSSBBBHH
.............   CCGGGQQQNNNMM
C.....Q..N..M   CCOOOQQQNNNMM
....O........   CCOOOOQQNNNMM
.............   LLUUODDDWWWVV
L..U..D..W..V   LLUUUDDDWWWVV
```

```
--- Ciclo --- 6000

Y....X..K..F.    YYZZXXXKKKFFF
...Z.........    YYZZZXXKKKFFP
............P    TTZZZEEERRRPP
T.....E..R...    TTIIIEEERRRAP
...I.......A.    TTIIIEEERRAAA
............     JJIIISSSBBAAH
J.....S..B..H    JJGGGSSSBBBHH
...G.........    JJGGGSSSBBBHH
............     CCGGGQQQNNNMM
C.....Q..N..M    CCOOOQQQNNNMM
....O........    CCOOOOQQNNNMM
............     LLUUODDDWWWVV
L..U..D..W..V    LLUUUDDDWWWVV
```

Para esta simulação foram empregados alguns parâmetros e cálculos apresentados em (RITTER e KOHONEN, 1989). Usou-se $\eta(t) = 0.8$ e

$$h_{ij} = exp\left(-\frac{d(i,j)^2}{\sigma(t)^2}\right),$$

onde $d(i,j)$ é a distância (Euclidiana) entre as unidades i e j e

$$\sigma(t) = \sigma_i(o_f/o_i)^{t/t_{max}},$$

onde t conta os ciclos, $\sigma_i = 4$, $\sigma_f = 0.5$. O parâmetro σ controla o raio da bolha de ativação, o qual decresce ao longo da simulação. Estes parâmetros fazem com que a vizinhança de uma unidade diminua com o tempo, localizando mais as alterações nos pesos.

As comparações dos mapas nos ciclos 5000 e 6000 mostram claramente que o algoritmo é bastante estável (nesta aplicação).

O que acontece se tentarmos empregar este algoritmo com 26 unidades de saída como no aprendizado competitivo? Dados ciclos suficientes (8000 nas simulações), o algoritmo consegue distinguir, normalmente, 24 categorias. Abaixo mostramos as unidades vencedoras em um determinado teste no ciclo 11000:

```
TJZCLUW.V.SEGQODBFPRHNMKXY.
```

Mas, infelizmente, o algoritmo mostrou-se instável neste caso. Observe a seguir como as vencedoras do teste anterior no ciclo 12000

```
TJIZCLUW.VESGQODBPFRAHNKXY
```

encontram-se alteradas no ciclo 14000:

```
TJIZCLUW.V.SEGQDBPFRHNMKXY.
```

Com constantes de aprendizado menores (e.g., $\eta(t) = 0.2$) as instabilidades parecem ser menores. Se colocarmos 30 unidades de saída o algoritmo resolve o problema, como mostram as vencedoras abaixo

```
TJIZ.V.WULCOQDBGSEFPRAHNMK.X.Y,
```

mas ainda apresenta pequenas instabilidades (e.g., apenas uma categoria muda). Com 80 unidades de saída o algoritmo obtém 26 categorias organizadas ao longo de um arranjo unidimensional e é, aparentemente, estável. Abaixo mostramos as unidades vencedoras e o mapa no ciclo 7000:

```
Z...I...J..T...Y...X...K..M..N...V...W..U...L..C..Q..O..D..B..S.G.E..F.P..R.A..H
ZZIIIIJJJJTTTYYYYYXXXKKKKMMMNNNVVVVWWWWUUULLLLCCCQQQOOODDDBBBSSSGGEEFFFPPRRAAAHH
```

Se alterarmos o algoritmo básico de Kohonen, só permitindo alteração de pesos na unidade vencedora (i.e., a vizinhança de uma unidade só inclui ela própria), este se torna muito semelhante ao aprendizado competitivo; há apenas duas diferenças: a normalização dos padrões de entrada e a forma de obter a unidade vencedora (através de distância mínima e não de produto escalar máximo). Este fato foi constatado experimentalmente, sendo o comportamento exibido bastante parecido com o do aprendizado competitivo (aparentemente, o algoritmo de Kohonen sem vizinhança é capaz de obter algumas categorias a mais).

O algoritmo de aprendizado de Kohonen apresenta um desempenho bastante superior ao do aprendizado competitivo. Em situações usuais ele é capaz de resolver o problema satisfatoriamente. Com relação ao tempo de aprendizado podemos dizer que o algoritmo é lento; são necessárias milhares de iterações para obtermos os resultados desejados.

Os testes com o algoritmo de Kohonen alterado deixam claro que o sucesso do algoritmo original depende bastante da existência de uma bolha de ativação na qual ocorrem alterações de pesos. Como vimos no capítulo 6, isto faz com que surja um mapa topográfico nas unidades de saída. Aparentemente, as influências que uma unidade exerce nas suas vizinhas permite que mais unidades se sensibilizem a padrões de entrada particulares.

A formação de uma organização global parece ser responsável pela estabilidade das categorizações. Como cada região exerce uma "força" nas regiões vizinhas as categorizações tendem a não se alterar (muito). A rigor poderíamos considerar as categorizações do algoritmo de Kohonen neste experimento instáveis já que há pequenas alterações no mapa; o número de unidades que codificam uma certa letra pode variar um pouco após o estabelecimento do mapa. No entanto, as mudanças são muito pequenas e não alteram a ordem do mapa.

ART. Como sabemos, os sistemas ART são estáveis. Além disso, a existência do parâmetro de vigilância faz com que possamos regular as categorizações da rede; valores perto de 1 fazem com que a rede só aceite diferenças muito pequenas em suas categorias. Por fim, os pesos variam segundo a hipótese do aprendizado rápido. Todos esses fatos nos levam a crer que os sistemas ART serão capazes de resolver o problema em questão em um tempo bastante pequeno.

Os testes realizados comprovam esta expectativa. As Tabelas (IX.6–10) exibem os resultados obtidos com diferentes parâmetros de vigilância. O sistema foi capaz de resolver o problema adequadamente com um valor de ρ alto (0.95). Estes resultados servem para demonstrar, também, os efeitos do parâmetro de vigilância nas categorizações do sistema.

A ordem da apresentação é bastante importante. Por exemplo, se a

nº unid. saída	26
ρ	0.3
ciclo	150
categorias detectadas	4
categoria	**elementos**
1	F, I, J, T, Z
2	K, X, Y
3	A, H, M, N, P, R, V, W
4	B, C, D, E, G, L, O, Q, S, U

Tabela IX.6:

nº unid. saída	26
ρ	0.5
ciclo	150
categorias detectadas	8
categoria	**elementos**
1	T
2	F, I, J, P, Z
3	X, Y
4	V, W
5	A, D, H, M, O, U
6	C, L
7	K, N, Q, R
8	B, E, G, S

Tabela IX.7:

nº unid. saída	26
ρ	0.7
ciclo	150
categorias detectadas	14
categoria	**elementos**
1	C
2	Y
3	T
4	W
5	V
6	L
7	K
8	F, P
9	X
10	I, J
11	A, E, G, R, S
12	B, D, O, Q
13	H, M, N, U
14	Z

Tabela IX.8:

apresentação dos padrões for seqüencial, obtém-se a categorização exibida na Tabela (IX.11) quando $\rho = 0.5$; compare com a Tabela (IX.7). Obviamente que para valores altos de ρ a ordem de apresentação não vai alterar os resultados da rede.

IX.3 Experimento 2

A dificuldade de categorização encontrada pelo aprendizado competitivo no Experimento 1 pode ser devida a pelo menos dois fatores: a) ao fato das letras serem muito parecidas e haver várias letras subconjuntos de outras; e b) ao fato das letras serem relativamente pequenas (poucos pontos), dificultando as categorizações do sistema. Criou-se, então, um segundo tipo de letra com 9x7 pontos onde as letras são mais distintas entre si do que no tipo anterior; ver Figura (IX.2). Em particular, nenhuma letra é subconjunto de outra.

Neste segundo experimento, o aprendizado competitivo foi testado. Apesar do tipo de letra ser diferente, os resultados foram basicamente os mesmos

n° unid. saída	26
ρ	0.8
ciclo	150
categorias detectadas	15
categoria	**elementos**
1	H, M, N
2	C, O
3	J, T
4	I, Z
5	Y
6	V
7	L
8	W
9	B, D
10	X
11	F, P
12	U
13	K
14	A, E, R
15	G, Q, S

Tabela IX.9:

nº unid. saída	26
ρ	0.9
ciclo	150
categorias detectadas	22
categoria	**elementos**
1	H
2	L
3	T
4	W
5	Y
6	Z
7	V
8	E, S
9	M, N
10	D
11	C
12	X
13	F, P
14	J
15	A, R
16	G
17	I
18	U
19	B
20	K
21	O
22	Q

Tabela IX.10:

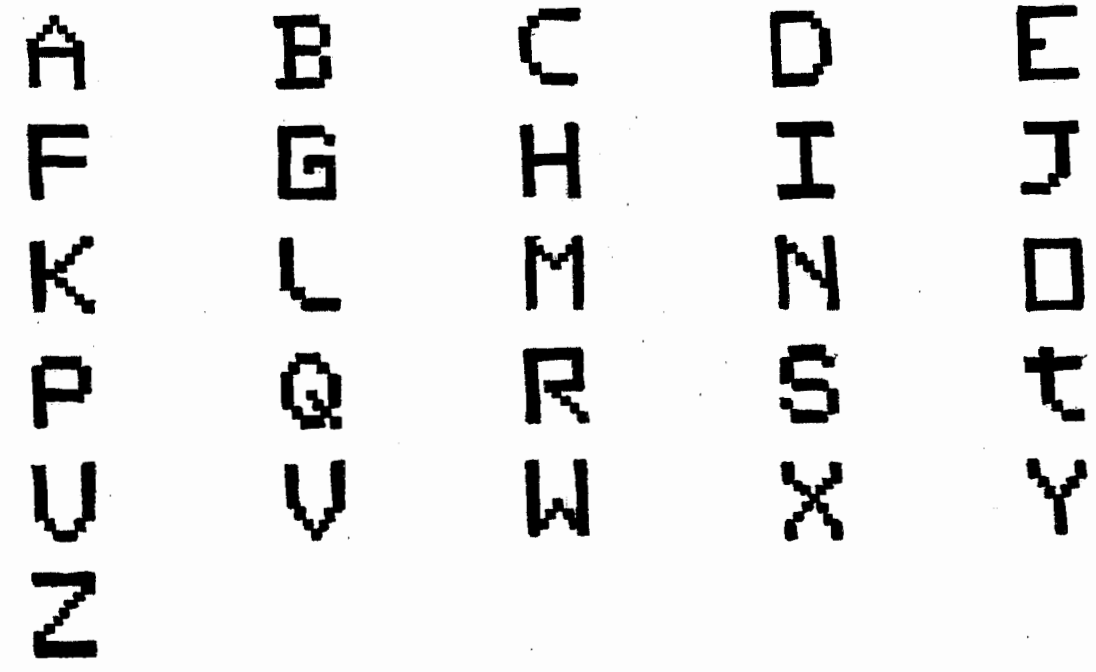

Figura IX.2: Conjunto de letras utilizado no Experimento 2

nº unid. saída	26
ρ	0.5
ciclo	150
categorias detectadas	8
categoria	**elementos**
1	A, B, C, D, F, O, P, R
2	K, L
3	Y
4	V, W
5	X, Z
6	H, M, N, U
7	I, J, T
8	E, G, Q, S

Tabela IX.11: Categorização com apresentação seqüencial dos padrões

nº unid. saída	26
ρ	0.2
ciclo	5000
categorias detectadas	5
categoria	**elementos**
1	T
2	V, X, Y
3	A, F, H, K, M, N, P, R, W
4	B, I, J, Z
5	C, D, E, G, L, O, Q, S, U

Tabela IX.12:

do Experimento 1. Em alguns casos algumas categorias detectadas pela rede foram parecidas com as categorias do Experimento 1. Entretanto, na maioria das vezes as similaridades observadas neste experimento são distintas das anteriores. Uma categorização típica é mostrada na Tabela (IX.12).

IX.4 Experimento 3

Um outro motivo que poderia ser responsável pelo baixo poder de categorização do aprendizado competitivo é uma possível competição "desleal" das unidades de saída.

Como visto no capítulo 5, algumas unidades podem não vencer nunca, diminuindo, assim, o número de categorias estabelecidas pela rede. Uma das possíveis soluções para este problema é introduzir um termo de penalização para cada unidade; unidades que vencem muito terão este termo elevado, enquanto que unidades que vencem menos possuirão uma penalização menor.

Neste terceiro experimento, o mecanismo de consciência foi incorporado no algoritmo básico de aprendizado competitivo. Esta pequena alteração provou ser bastante eficaz. A rede passou a estabelecer entre 20 e 25 categorias para ambos os tipos de letras.

Dois métodos de consciência foram utilizados. Um método é bastante simples: cada unidade possui um termo de penalização p_i, o qual é incrementado cada vez que a unidade i ganha a competição. Neste caso o algoritmo conseguiu obter entre 20 e 25 categorias. O outro método é um pouco mais sofisticado, mas a idéia é análoga. Ele apenas utiliza um termo de penalização que varia mais lentamente e que pode tanto aumentar quanto diminuir com o tempo; ver (DESIENO, 1988). Este segundo método obteve entre 23 e 25 categorias.

Na realidade DESIENO (1988) mostra que o mecanismo de consciência melhora sensivelmente os resultados do sistema em muitos tipos de aplicação, revelando a importância deste mecanismo.

Esta melhoria de desempenho ajuda a entender, também, o sucesso do algoritmo de Kohonen. Ao alterar os pesos em toda uma vizinhança, o algoritmo permite que unidades perdedoras também se sensibilizem a padrões particulares, aumentando o número de categorias da rede.

IX.5 Experimento 4

Um aspecto importante do reconhecimento de padrões é a tolerância a variações do método de reconhecimento. No caso em questão, gostaríamos que uma rede que reconhece, por exemplo, um A consiga identificar um A deformado.

Neste quarto experimento avaliamos as capacidades do algoritmo de Kohonen em reconhecer estímulos deformados que não foram apresentados à rede durante o aprendizado. Não testamos o aprendizado competitivo porque ele não resolve adequadamente o problema, nem os sistemas ART porque o parâmetro de vigilância alto não permitiria o reconhecimento de padrões deformados (é preciso enfatizar que os sistemas ART são módulos de categorização; para serem usados no reconhecimento de padrões eles devem estar associados a pré-processadores).

Se considerarmos que o reconhecimento correto ocorre quando a mesma unidade de categoria é ativada (vence) com apresentação tanto do padrão original quanto do deformado, o desempenho do algoritmo é bastante fraco. Abaixo mostramos um dos testes realizados quando a deformação das letras foi de 5 porcento (neste teste foi usado o tipo de letra 1). Apenas oito poições coincidem, ou seja, só houve reconhecimento correto em oito casos. Entretanto, se observarmos os respectivos mapas veremos que eles são semelhantes. Com deformações de 10 porcento ou mais os resultados são bastante pobres.

```
--- Ciclo --- 6000

K..M..H..P.F   KKMMMHHHPPFF
............   KXXMNNHHAPFF
.X..N...A...   XXXNNNNAAARR
...........R   XXXNNQQQAARR
......Q.....   YYVVVQQQDDQR
Y..V.....D..   YYVVVQQODDDB
.......O...B   YYVVVUOOODBB
.....U......   TTWWUUUOOGBB
T..W.....G..   TTWWWULLGGGS
......L....S   TTWWWLLLGGSS
............   JJIIZZLCCCSE
J..I.Z..C..E   JJIIIZZCCCEE
```

```
--- Deformado ---

...M..H..P..    RRMMMHHHPPPP
....N......F    MXXMMHHHPPPP
.X..........    XXXMMMHRRRRR
.......B...R    XXXMMOOORRRR
Y..........K    YYVVVOOOOOOR
......Q.....    YYVVVOOOOOOS
....V.DO....    YYVVUUOOOOSS
.T..........    TTWWUUUOOSSS
.....U....G.    TTWWWUUUSSSS
....W......S    TTWWWUUUSSSS
..I..CLJ....    ZZZZZZUOOOSS
.....Z....E.    ZZZZZZZOOOSS
```

Podemos dizer que, em geral, a taxa de reconhecimento é baixa, mas os mapas são parecidos. Entretanto, segundo Kohonen (KANGAS et al., 1989) o mecanismo básico de auto-organização tem como função apenas a visualização de relações métricas-topológicas dos sinais de entrada. E afirma que os mapas não devem ser usados em aplicacões de reconhecimento de padrões. Para tanto pode-se utilizar variações do algoritmo básico mais apropriadas; ver, e.g., (KOHONEN et al., 1988).

Capítulo X

Conclusões

A área de aprendizado em redes neuronais tem crescido rapidamente nos últimos anos. Uma quantidade enorme de trabalhos é publicada nesta área. Dentre os principais avanços recentes podemos destacar:

- Elaboração de algoritmos de aprendizado para redes recorrentes, e.g., (WILLIAMS e ZIPSER, 1989);

- Algoritmos de categorização estáveis, tais como os dos sistemas ART 1 e ART 2 (CARPENTER e GROSSBERG, 1988), e outros algoritmos de categorização poderosos, como o de KOHONEN (1984);

- Mecanismos de aprendizado não-supervisionado estáveis para sistemas com função de Liapunov (KOSKO, 1988; 1989), e vários outros mecanismos de aprendizado para estes sistemas (ABBOTT, 1990);

- Formas de aprendizado por reforço poderosas, como o modelo A_{R-P} de BARTO (1985), e análises deste paradigma de aprendizado (WILLIAMS, 1986);

- Algoritmos supervisionados para redes de múltiplas camadas, tais como back-propagation (RUMELHART et al., 1986b).

Não podemos dar a entender, entretanto, que esta área não enfrenta alguns problemas sérios. Dentre eles, um se destaca: a velocidade do aprendizado. A grande maioria dos algoritmos são (bastante) lentos e requerem muitas iterações para a obtenção de resultados satisfatórios. Este problema pode ser minimizado através da utilização de hardware específico, mas de qualquer forma o problema é bastante sério. Ver (HINTON, 1989) para uma discussão mais detalhada de alguns problemas com o aprendizado em redes neuronais.

Dentro de um campo tão vasto e dinâmico, escolhemos apenas uma subclasse dos algoritmos não-supervisionados para nos concentrarmos: os algoritmos de categorização. Mesmo dentro desta subclasse não fomos exaustivos. Ao invés disso, preferimos nos aprofundar em uns poucos algoritmos, estudando com cautela seus conceitos principais. Dentre estes podemos destacar o conceito de estabilidade, cujas questões relacionadas vêm ganhando extrema importância recentemente.

Vimos, nesta tese, que um algoritmo simples como o do aprendizado competitivo é capaz de exibir um comportamento interessante, apesar de apresentar alguns problemas sérios, tais como a instabilidade e a competição "desleal". Poucas modificacões no esquema básico deste algoritmo dão margem a resultados poderosos (e.g., mapas topográficos), como vimos ao estudar a auto-organização de mapas de características de Kohonen. Investigamos detalhadamente, também, os sistemas ART 1, módulos capazes de estabelecer categorizações estáveis, que apresentam muitas características interessantes.

Os sistemas ART 1 (e ART 2) têm recebido mais atenção recentemente com o reconhecimento (tardio) de seu poder, e o número de trabalhos vem se multiplicando, abordando áreas diversas, tais como reconhecimento de padrões, implementações em hardware, e estudo das relações de sistemas biológicos e sistemas ART (quanto a esta última área, ver, e.g., GOCHIN, 1990).

Sem dúvida, pode-se esperar para o futuro próximo muitos novos mecanismos de aprendizado capazes de expandir o poder dos algoritmos correntes. Uma linha de pesquisa de especial relevância para os algoritmos de categorização desta tese é o desenvolvimento de modelos de aprendizado não-supervisionado em

redes de múltiplas camadas; e.g., (CARPENTER e GROSSBERG, 1990), (LINSKER, 1988) e (FROHN et al., 1987). Uma possível tendência futura para a área de aprendizado é a elaboração de algoritmos mistos, aproveitando aspectos dos paradigmas supervisionado, não-supervisionado, e aprendizado por reforço. Por fim, não podemos deixar de mencionar a eventual combinação de mecanismos de aprendizado conexionistas com outras formas não-conexionistas (CARBONELL, 1989); uma união que pode render mecanismos ainda mais poderosos.

Referências Bibliográficas

ABBOTT, L. F. (1990), "Learning in neural memories", *Network*, 1, 105-122.

ANDERSON, J. A. (1983), "Cognitive and psychological computation with neural models", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 13, No. 5, 799-815.

ARBIB, M. A. (1987), *Brains, machines, and mathematics*, NY, Springer-Verlag, Second Edition.

BARTO, A. G. (1985), "Learning by statistical cooperation of self-interested neuron-like computing units", *Human Neurobiology*, 4, 229-256.

BIENENSTOCK, E. L., COOPER, L. N., e MUNRO, P. W. (1982), "Theory for the development of neuron selectivity", *Journal of Neuroscience*, 2, 32-48.

BROWN, T. H., CHAPMAN, P. F., KAIRISS, E. W., e KEENAN, C. L. (1988), "Long-term synaptic potentiation", *Science*, Vol. 242, 724-728.

CARBONELL, J. G. (1989), "Introduction: Paradigms for machine learning", *Artificial Intelligence*, 40, 1-9.

CARPENTER, G. A. (1989), "Neural network models for pattern recognition and associative memory", *Neural Networks*, 2, pp. 243-257.

CARPENTER, G. A., e GROSSBERG, S. (1987a), "A massively parallel architecture for a self-organizing neural pattern recognition machine", *Computer Vision, Graphics, and Image Processing*, 37, 54-115.

CARPENTER, G. A., e GROSSBERG, S. (1987b), "ART 2: Self-organization of stable recognition codes for analog input patterns", *Applied Optics*, 26, 4919-4930.

CARPENTER, G. A., e GROSSBERG, S. (1988), "The ART of adaptive pattern recognition by a self-organizing neural network", *IEEE Computer*, Vol 21, No. 3, 77-88.

CARPENTER, G. A., e GROSSBERG, S. (in press), "ART 3: Hierarquical search using chemical transmitters in self-organizing pattern recognition architectures", *Neural Networks.*

DESIENO, D. (1988), "Adding a conscience to competitive learning", *Proceedings of the IEEE International Conference on Neural Networks (Vol. I)*, 117-124, San Diego.

DURBIN, R., e WILLSHAW, D. (1987), "An analogue approach to the traveling salesman problem using an elastic net method", *Science*, 326, 689-691.

FAHLMAN, S. E. (1988), "An empirical study of learning speed in back-propagation networks", Technical Report CMU-CS-88-162, Computer Science Department, Carnegie Mellon University.

FANTY, M. A. (1988), "Learning in structured connectionist networks", Technical Report 252, Computer Science Department, University of Rochester.

FERREIRA, A. B. (1975), *Novo dicionário da língua portuguesa*, Rio de Janeiro, Editora Nova Fronteira.

FISCHLER, M. A., e FIRSCHEIN, O. (1987), *Intelligence: The eye, the brain, and the computer*, Addison-Wesley Publishing Company, Reading, MA.

FODOR, J. A., e PYLYSHYN, Z. W. (1988), "Connectionism and cognitive architecture: A critical analysis", *Cognition*, 28, 3-71.

FORT, J. (1988), "Solving a combinatorial problem via self-organizing process: An application of the Kohonen algorithm to the traveling salesman problem", *Biological Cybernetics*, 59, 33-40.

FROHN, H., GEIGER, H., e SINGER, W. (1987), "A self-organizing neural network sharing features of the mammalian visual system", *Biological Cybernetics*, 55, 333-343.

GOCHIN, P. M. (1990), "Pattern recognition in primate temporal cortex: But is it ART?", *Proceedings of the International Joint Conference on Neural Networks (Vol.*

I), 77-80, Washington, DC.

GROSSBERG, S. (1976a), "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural detectors", *Biological Cybernetics*, 23, 121-134.

GROSSBERG, S. (1976b), "Adaptive pattern classification and universal recoding: II. Feedback, oscillation, feedback, and illusions", *Biological Cybernetics*, 23, 187-207.

GROSSBERG, S. (1980), "How does a brain build a cognitive code?", *Psychological Review*, 87, 1-51.

GROSSBERG, S. (1987), "Competitive learning: From interactive activation to adaptive resonance", *Cognitive Science*, 11, 23-63.

GROSSBERG, S. (1988), "Nonlinear neural networks: Principles, mechanisms, and architectures", *Neural Networks*, 1, 17-61.

GROSSBERG, S., e SCHMAJUK, N. A. (1987), "Neural dynamics of attentionally modulated Pavlovian conditioning", *Psychobiology*, 15, 195-240.

GROSSBERG, S., e STONE, G. (1986), "Neural dynamics of word recognition and recall: Attentional priming, learning, and resonance", *Psychological Review*, 93, 46-74.

HEBB, D. O. (1949), *The organization of behavior*, New York, Wiley.

HECTH-NIELSEN, R. (1987), "Counterpropagation networks", *Applied Optics*, 26, 4979-4985.

HECTH-NIELSEN, R. (1989), "Theory of the backpropagation neural network", *Proceedings of the International Joint Conference on Neural Networks (Vol. I)*, 593-605, San Diego.

HINTON, G. E. (1989), "Connectionist learning procedures", *Artificial Intelligence*, 40, 185-234.

JORDAN, M. I. (1986), "An introduction to linear algebra in Parallel Distributed Processing", em D. E. RUMELHART e J. L. MCCLELLAND (Eds.), *Parallel Distributed Processing (Vol. 1)*, Cambridge, MA, The MIT Press.

KANGAS, J., KOHONEN, T., LAAKSONEN, J., SIMULA, O., e VENTA, O. (1989), "Variants of self-organizing maps", *Proceedings of the International Joint Conference on Neural Networks (Vol. II)*, 517-522, San Diego.

KOHONEN, T. (1982a), "Self-organized formation of topologically correct feature maps", *Biological Cybernetics*, 43, 59-69.

KOHONEN, T. (1982b), "Analysis of a simple self-organizing process", *Biological Cybernetics*, 44, 135-140.

KOHONEN, T. (1982c), "Clustering, taxonomy and topological maps of patterns", *Proceedings of the Sixth International Conference on Pattern Recognition*, 114-128.

KOHONEN, T. (1984), *Self-Organization and Associative Memory*, Berlin, Springer-Verlag.

KOHONEN, T. (1987), "Adaptive, associative, and self-organizing funtions in neural computing", *Applied Optics*, 26, 4910-4918.

KOHONEN, T. (1988), "The "neural" fonetic typewriter", *IEEE Computer*, Vol. 21, No. 3, 11-22.

KOHONEN, T., BARNA, G., e CHRISLEY, R. (1988), "Statistical pattern recognition with neural networks: Benchmarking studies", *Proceedings of the International Conference on Neural Networks (Vol. I)*, 61-68, San Diego.

KOSKO, B. (1987), "Adaptive bidirectional associative memories", *Applied Optics*, 26, 4947-4960.

KOSKO, B. (1989), "Unsupervised learning in noise", *Proceedings of the International Joint Conference on Neural Networks (Vol. I)*, 7-17, San Diego.

LEVINE, D. (1983), "Neural population modeling in psychology: A review", *Mathematical Biosciences*, 66, 1-86.

LINSKER, R. (1988), "Self-organization in a perceptual network", *IEEE Computer*, Vol. 21, No. 3, 105-117.

LIPPMANN, R. (1987), "An introduction to computing with neural nets", *IEEE ASSP Magazine*, 4, 4-22.

LIPPMANN, R. (1988), "Review of neural networks for speech recognition", *Neural Computation*, 1, 1-38.

MCCLELLAND, J. L., e RUMELHART, D. E. (1985), "Distributed memory and the representation of general and specific information", *Journal of Experimental Psychology: General*, 114, 159-188.

MCCLELLAND, J. L., e RUMELHART, D. E. (1988), *Explorations in Parallel Distributed Processing*, Cambridge, MA, The MIT Press.

MATHEUS, C. J., e HOHENSEE, W. E. (1987), "Learning in artificial neural systems", *Comput. Intell.*, 3, 283-294.

MINSKY. M., e PAPERT, S. (1969), *Perceptrons*, Cambridge, MA, MIT Press.

MINSKY. M., e PAPERT, S. (1986), *Perceptrons*, Cambridge, MA, MIT Press, Expanded Edition.

NILLSON, N. (1965), *Learning Machines*, NY, McGraw-Hill.

PAPERT, S. (1988) "One AI or many?", *Daedalus*, 117(1), 1-14.

PESSOA, L. A., e REIS, A. (1990), "Avaliação da representação de schemata em modelos conexionistas", *Revista Brasileira de Computação*, Vol. 5., No. 4.

RITTER, H. J., e KOHONEN, T. (1989), "Self-organizing semantic maps", *Biological Cybernetics*, 61, 241-254.

RITTER, H. J., MARTINEZ, T. M., e SCHULTEN, K. J. (1989), "Topology-conserving maps for learning visuo-motor-coordination", *Neural Networks*, 2, 159-168.

RITTER, H, e SCHULTEN, K. (1988), "Kohonen's self-organizing feature maps: Exploring their computational capabilities", *Proceedings of the IEEE International Conference on Neural Networks (Vol I)*, 109-116, San Diego.

RUMELHART, D. E., HINTON, G. E., e MCCLELLAND, J. L. (1986a), "A general framework for Parallel Distributed Processing", em D. E. RUMELHART e J. L. MCCLELLAND (Eds.), *Parallel Distributed Processing (Vol. I)*, Cambridge, MA, The MIT Press.

RUMELHART, D. E., HINTON, G. E., e WILLIAMS, R. J. (1986b), "Learning internal representations by error propagation", em D. E. RUMELHART e J. L. MCCLELLAND (Eds.), *Parallel Distributed Processing (Vol. I)*, Cambridge, MA, The MIT Press.

RUMELHART, D. E., e ZIPSER, D. (1985), "Feature discovery by competitive learning", *Cognitive Science*, 9, 75-112.

RYAN, T., e WINTER, C. (1987), "Variations on adaptive resonance", *Proceedings of the IEEE International Conference on Neural Networks (Vol. I)*, 673-680, San Diego.

SEJNOWSKI, T., e ROSENBERG, C. (1987), "Parallel networks that learn to pronounce English text", *Complex Systems*, 1, 145-168.

SHEPHERD, G. M. (1988), *Neurobiology*, Oxford University Press, New York, Second Edition.

SIMPSON, P. K. (1990), *Artificial neural systems: Foundations, paradigms, applications, and implementations*, Elmsford, NY, Pergamon Press.

SUTTON, R. S., e BARTO, A. G. (1981), "Toward a modern theory of adaptive networks: Expectation and prediction", *Psychological Review*, 88, 135-170.

TESAURO, G., e SEJNOWSKI, T. (1989), "A parallel network that learns to play backgammon", *Artificial Intelligence*, 39, 357-390.

TOU, J. T., e GONZALEZ, R. C. (1974), *Pattern recognition principles*, Addison-Wesley Publishing Company, Reading, MA.

WILLIAMS, R. J. (1986), "Reinforcement learning in connectionist networks: A mathematical analysis", Technical Report 8605, Institute for Cognitive Scinece, University of California at San Diego.

WILLIAMS, R. J., e ZIPSER D. (1989), "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation*, 1, 270-280.