Alto Nível em Redes Neurais

Rodrigo de Salvo Braz

Dissertação apresentada ao Instituto de Matemática e Estatística da Universidade de São Paulo para a obtenção do grau de Mestre em Matemática Aplicada.

Orientador: Prof. Dr. Flávio Soares Corrêa da Silva

Área de Concentração: Ciência da Computação

São Paulo, 17 de janeiro de 1998.

Alto Nível em Redes Neurais

Este exemplar corresponde à
redação final da dissertação
devidamente corrigida e defendida
por Rodrigo de Salvo Braz e
aprovada pela comissão julgadora.

São Paulo, 17 de janeiro de 1998.

Banca examinadora:

- Prof. Dr. Flávio Soares Corrêa da Silva (orientador) - IME-USP
- Prof. Dr. Marco Dimas Gubitoso - IME-USP
- Prof. Dr. André Ponce de Leon F. de Carvalho - ICMSC-USP

Resumo

Nesta dissertação apresentamos diversos modelos de redes neurais, ou conexionistas, na busca de uma visão unificada destes modelos e de um entendimento de suas capacidades e limitações. Em seguida, esta visão unificada é usada para considerar-se o tratamento de problemas de alto nível por parte destes modelos que, embora tidos como inapropriados a este tipo de problema, possuem características que seriam úteis em seu tratamento, tais como aprendizado com generalização automática e degradação suave. Os principais problemas são identificados, e um modelo conexionista que trata um deles é apresentado.

Abstract

In this dissertation we present several neural or connectionist network models, looking for a unified vision and understanding of their capabilities and limitations. Then, this unified vision is used to think about the treatment of high level problems by these models which, although generally thought of as inappropriate for this kind of problems, have several features that could be useful for them, such as learning with automatic generalization and graceful degradation. The main problems are identified, and a connectionist model that deals with one of them is presented.

Agradecimentos

Ao Prof. Flávio Soares Corrêa da Silva, por seu apoio irrestrito e sua disposição, capacidade e respeito ao ouvir idéias alheias.

Ao Prof. Paulo Feofiloff, por sua atenção, cortesia e preocupação constante em prestar o melhor auxílio.

Ao CNPq, que financiou parte desta pesquisa.

À Comunidade IME, que mostrou-se sempre solícita na prestação de seu apoio, com cordialidade além das expectativas.

Aos meus pais, pelo apoio amoroso e desinteressado desde sempre.

Aos vários amigos que me acompanharam com carinho e interesse.

Sumário

1. Introdução	5
2. Redes conexionistas	7
2.1 Primeiras idéias	7
2.2 Limitações	12
2.3 Aprendizado em redes de múltiplas camadas: backpropagation	15
2.4 Auto-associadores	18
2.5 Outros modelos	26
2.6 Síntese	30
3. Alto nível em redes conexionistas	37
3.1 O que é alto nível?	38
3.2 A questão da representação simbólica	42
3.3 Níveis de análise	43
3.4 Quem está mais próximo do nível cognitivo?	45
3.5 Problemas e abordagens conexionistas às necessidades do nível cognitivo	52
3.6 Síntese	67
4. Um modelo conexionista com aprendizagem de múltiplas fases	68
4.1 Alto nível e a questão do conhecimento prévio	68
4.2 O modelo conexionista utilizado	69
4.3 O exemplo apresentado	73
4.4 Ferramentas para o treinamento	74
4.5 Os testes realizados e o que eles indicam	77
4.6 Limitações	79
4.7 Desempenho	80
4.8 Síntese	81
5. Conclusão	82
6. Bibliografia	84

1. Introdução

No final da década de 70, um certo desânimo pairava sobre o campo da Inteligência Artificial (IA). Outrora confiantes, os pesquisadores da área percebiam que fatores como contexto, aproximações, conceitos difusos, além de outros, desempenhavam um papel no raciocínio muito mais importante do que o inicialmente suposto e, além disso, pareciam extremamente difíceis de reproduzir.

Ao mesmo tempo, as redes neurais, também conhecidas por *perceptrons* e, mais recentemente, *redes conexionistas*, recebiam novamente atenção crescente. Desenvolvidas desde a década de 40 como possibilidade de um modelo de computação baseado no que se conhecia sobre o sistema nervoso, as redes neurais compunham-se de diversos processadores simples, cujo funcionamento era inspirado nos neurônios e que operavam em paralelo. Estes modelos exibiam comportamentos interessantes, como degradação suave em função de danos ou entradas incorretas, e alguns tipos de generalização automática. Entretanto, após um importante trabalho de Minsky e Papert, *Perceptrons*, de 1969, as redes neurais haviam sido praticamente abandonadas por se acreditar impossível fazê-las tratar funções além das triviais.

Porém, no decorrer da década de 70 algumas das mais importantes limitações das redes neurais, entre elas a dificuldade de aprendizado para redes com múltiplas camadas, foram contornadas. Nos anos 80, as primeiras aplicações com estas redes mais poderosas tornaram evidente que os elementos estatísticos com que a IA simbólica se debatia há anos podiam, de um modo ou de outro, ser tratados por estas redes. Elas podiam, como aplicações típicas, reconhecer padrões, aprender com base em exemplos e tratar imprecisões e incertezas, realizando em certo grau aquilo em que a IA simbólica encontrava dificuldades.

Em meio ao entusiasmo com que se atacava os problemas, principalmente de baixo nível, que podiam ser resolvidos com as redes conexionistas, havia o sentimento entre alguns de que elas ampliavam os horizontes de todo o campo. Se a maneira das redes conexionistas lidarem com elementos estatísticos pudesse ser empregada nos demais problemas, um grande passo seria dado.

Porém, o que se percebe hoje em dia, mais de uma década depois, é que ainda não houve este passo. As aplicações de redes conexionistas permanecem basicamente as mesmas: instâncias de reconhecimento de padrões, seja sobre sinais, imagens, sons ou uma série temporal. A grande maioria da pesquisa na área dedica-se ao aperfeiçoamento técnico, buscando melhorar o desempenho no reconhecimento de padrões. Muitos manifestam a idéia de que a IA conexionista presta-se somente ao processamento de baixo nível, enquanto a IA simbólica cuida daqueles de alto nível. Um sintoma disto são os sistemas híbridos, que solucionam problemas dividindo-os estritamente em subproblemas pertencentes a um ou outro nível.

A desvantagem desta visão, porém, é que a IA de alto nível deixa de se beneficiar justamente das características das redes conexionistas que poderiam resolver alguns de seus problemas. Mostraremos que esta não é uma necessidade. Os problemas de alto nível podem ter seu tratamento auxiliado pelo que se aprendeu com os modelos conexionistas, superando as barreiras com que os pesquisadores defrontaram-se já no final da década de 70.

Nesta dissertação serão discutidas as características de redes neurais e de problemas de alto nível que indicam possível a contribuição em sua resolução por parte dos modelos conexionistas. O capítulo 2 faz uma apresentação crítica do histórico das redes e suas propriedades, procurando definir quais são suas principais características e possibilidades.

O capítulo 3 apresenta considerações sobre o tratamento de problemas de alto nível em redes neurais. Para isso, procura definir melhor o que caracteriza estes problemas e apresentar os argumentos a favor e contra a possibilidade destas redes tratá-los com sucesso. Em particular, a disputa entre aqueles que defendem os símbolos como única base possível para a IA e os defensores das redes neurais é discutida. Alguns dos trabalhos com as redes nesta área são comentados.

O capítulo 4 ilustra os tópicos apresentados através de um modelo conexionista que trata um aspecto relacionado a alto nível sem perder as propriedades desejáveis destas redes.

2. Redes connexionistas

2.1 Primeiras idéias

A relação entre redes de neurônios e sistemas processadores de informação passou a ser traçada já na década de 40, quando McCulloch e Pitts apontaram para o fato de que neurônios podiam ser considerados processadores booleanos de suas conexões, sendo possível criar uma rede neural que calculasse qualquer função lógica (Gardner, 1987; Rumelhart & McClelland, 1986).

Estas idéias baseavam-se nos estudos sobre o comportamento de células nervosas em rede e individualmente. Um dos modelos mais simples de um neurônio, e de fato um dos mais usados até hoje, espelha o fato de um neurônio receber, através de suas sinapses com outras células nervosas, os estímulos destas células, integrá-los e, de acordo com o resultado desta integração (chamado comumente de *potencial de ativação*), disparar ele próprio seu estímulo, em determinada frequência, para outras células nervosas.

A integração dos estímulos que se dá no neurônio é complexa e varia de tipo para tipo de célula. Entretanto, de forma geral observa-se que um estímulo vindo de uma outra célula é mediado pela força da conexão entre elas. Assim, o estímulo de uma determinada célula A pode influir muito no disparo ou não de uma célula B, enquanto outra célula C tem pouca ou nenhuma influência sobre isto. Pode ocorrer também de o disparo de uma célula inibir o de outra. Este tipo de comportamento pode ser sintetizado pelo modelo como ilustrado na figura 2-1.

onde a função de transferência T pode ser, entre outras:

figura 2-1: Um modelo simples de neurônio. Uma soma ponderada pelo peso de cada conexão integra os diversos estímulos $x_1, x_2, \dots, x_n$ vindos de outras células. Uma função de transferência, da qual três exemplos típicos são mostrados, determina como a soma ponderada gera o estímulo de saída. No terceiro exemplo, a célula só dispara a partir de um determinado potencial L , chamado limiar, ser atingido.

A partir deste modelo, McCulloch e Pitts mostraram que qualquer função lógica podia ser implementada, através da composição das funções lógicas básicas **e**, **ou** e **não**, implementadas pelo modelo como ilustra a figura 2-2.

Estes fatos levaram ao interesse em modelos de redes neurais que pudessem reproduzir procedimentos mentais, através da compreensão detalhada da circuitaria cerebral. Mas embora lançassem uma ponte entre a neurologia e a computação, não era claro como utilizá-las para atingir este objetivo. Um dos grandes obstáculos, por exemplo, era reproduzir o aprendizado, entendido como a adaptação crescente ao ambiente através da experiência. Isto permitiria que redes fossem construídas apenas com a interação automática com o ambiente ou uma simulação deste, uma natural e atraente alternativa à montagem exaustiva de redes que realizassem tarefas inteligentes.

figura 2-2: Exemplos de como o modelo de neurônio pode calcular funções lógicas, com "1" representando Verdadeiro e "0" representando Falso. O número dentro das unidades representam seu limiar. Compondo estas funções elementares, pode-se calcular qualquer função lógica.

Apesar das especulações por parte de Donald Hebb em seu livro *Organizations of Behaviour* (1949) de que o conhecimento no sistema nervoso se baseava nas conexões entre os neurônios e que portanto o aprendizado devia se refletir na força destas conexões, apenas em 1958 surgiu um primeiro modelo de rede conexionista artificial que podia aprender. Desenvolvido por Frank Rosenblatt, o chamado *Perceptron* chamou a atenção por suas características computacionais peculiares. Voltado principalmente para a reprodução do processamento visual no sistema nervoso, ele consistia de um conjunto de unidades, ou neurônios artificiais, organizados em três camadas: uma de entrada (ou *retina*), uma intermediária (ou *de predicados*) e uma de saída (ou de *decisão*) (figura 2-3). Quando uma entrada era apresentada a esta máquina, ela era representada na retina através de ativações nas unidades desta camada. Cada unidade da camada de predicados, então, através de suas conexões com um subconjunto da retina, podia ou não disparar, ou *tornar-se ativada*, de acordo com os pesos das conexões. As unidades de decisão, finalmente, usavam estes predicados para calcular a função final, através dos pesos de suas conexões com elas. Estes pesos das conexões entre a camada de predicados e a de decisão eram modificadas de acordo com a decisão correta desejada, fornecida por um supervisor.

figura 2-3: O Perceptron de Rosenblatt

O resultado era uma máquina que podia aprender a classificar um conjunto arbitrário de padrões, a partir de exemplos. Os pesos das conexões alteravam-se sucessivamente, adaptando-se às respostas desejadas e tornando a máquina cada vez mais correta.

O Perceptron já exibía uma característica que tornar-se-ia marcante nas redes conexionistas: a *degradação suave*. A degradação suave é a capacidade de um sistema em fornecer respostas úteis, ainda que não necessariamente perfeitas, mesmo que sua entrada esteja consideravelmente corrompida, incompleta ou com ruído. Isto acontece porque cada unidade usa diversas fontes para tomar a sua “decisão,” ou seja, para integrar e disparar no caso do limiar da função de transferência ser atingido ou ultrapassado. Então, se algumas destas fontes não estiverem corretas, mas apenas com informações aproximadas, o perceptron ainda assim poderá disparar (ou não) corretamente.

Outra característica importante é a generalização automática baseada em similaridades que o Perceptron apresenta. Quando uma unidade de decisão aprende corretamente qual valor disparar, algumas de suas conexões com os predicados estarão mais fortes do que outras. Aqueles predicados cujas conexões estão fracas são, na verdade, pouco importantes para aquela decisão. Basicamente, qualquer valor por eles tomado conduz ao mesmo valor na unidade de decisão, se o restante dos predicados for o mesmo. Portanto, a unidade de decisão tomará o valor correto para toda uma *classe de entradas* em que só varie o valor destes predicados menos importantes, que dão origem a diferentes instâncias desta classe. Terá havido, portanto, uma generalização sobre esta classe.

Rosenblatt provou um importante teorema mostrando que qualquer solução representável por um Perceptron podia ser aprendida por ele (Rosenblatt, 1962). Como veremos adiante, porém, havia sérias limitações sobre *o que* podia ser aprendido por ele.

Outro modelo, o ADALINE, de Widrow e Hoff, foi criado um pouco mais tarde e apresentava uma regra de aprendizado sofisticada, a Regra Delta, que se tratava de um método de regressão linear que diminuía a cada exemplo a distância entre a saída obtida e a desejada (figura 2-4), através de adaptações graduais dos pesos do Perceptron.

α : taxa de aprendizado

D: saída desejada

S: saída real

$$\Delta w_i = \alpha.(D - S).x_i$$

figura 2-4: A unidade Adaline

2.2 Limitações

Na década de 60, diversas limitações dos Perceptrons tornaram-se claras. Em um importante trabalho, *Perceptrons*, Minsky e Papert (1969) exibiram de maneira rigorosa as diversas limitações computacionais dos modelos até ali existentes. Uma delas consistia no fato de que algumas funções muito simples, como a função booleana **ou-exclusivo**, não podiam ser aprendidas por um perceptron. Isto acontece porque um perceptron é, fundamentalmente, um discriminador linear, ou seja, pode apenas distinguir, no espaço de suas entradas, entre as duas classes pertencentes a um ou outro dos semi-espacos definidos por um hiperplano (figura 2-5), por sua vez definido pelos pesos do perceptron.

figura 2-5: Um perceptron só pode distinguir entre classes linearmente separáveis, como as da esquerda. Em muitas funções isto não ocorre, como no caso da ou-exclusivo, à direita.

Esta é uma severa limitação, pois em geral apenas funções muito simples são linearmente separáveis. Não é o caso da já citada função booleana **ou-exclusivo** (XOR), que aparece à direita da figura 2-5.

Pode-se pensar em uma solução para este problema com a superposição de camadas de perceptrons, compondo funções mais simples para chegar a uma mais complexa (basta lembrar que as redes de McCulloch e Pitts faziam exatamente isto). A figura 2-6 mostra um exemplo de implementação de uma rede com mais de uma camada que calcula a função **ou-exclusivo**. Note que mesmo com mais de uma camada a rede mantém a desejável característica da degradação suave, pois as entradas têm uma faixa de variação que deixa o resultado inalterado. Devido ao fato das unidades em camadas intermediárias não conterem valores nem de entrada nem de saída, são chamadas usualmente de *unidades ocultas*. Seus valores não influenciam nem vêm do ambiente diretamente.

figura 2-6: Uma rede de perceptrons que calcula a função ou-exclusivo sobre as entradas x_1 e x_2 . Os números dentro dos perceptrons são seus limiares, e os números próximos às conexões são seus pesos. A unidade central detecta o caso em que as duas entradas valem 1 e inibe a ativação da unidade de saída.

Pode-se também falar da questão da necessidade de unidades ocultas em termos de *estruturas* de primeira, segunda ou maior ordem. Um conjunto de vetores apresenta *estrutura de primeira ordem* se a possibilidade de um de seus componentes assumir um valor não depende dos valores dos outros componentes. Isto significa que este componente e a maneira como ele se comporta podem ser estudados sem levar os demais componentes em consideração. O que importa é a probabilidade de cada componente assumir um ou outro valor. Redes conexionistas em geral representam estruturas de primeira ordem através do limiar de cada unidade, predispondo-a a assumir um valor ou outro. Um conjunto de vetores apresenta *estruturas de segunda ordem* se o valor de um componente depende do valor de outros componentes em maior ou menor grau. Isto reflete a probabilidade condicional de uma certa componente assumir um valor, dado que a outra tenha assumido um certo valor. Esta estrutura normalmente é representada, em redes conexionistas, pelo peso da conexão entre duas unidades, e é facilmente determinada pela contabilização das ocorrências conjuntas entre elas. O aprendizado baseado em exemplos de estruturas de primeira e segunda ordens é algumas vezes chamado *aprendizado fácil*, exatamente por depender de uma simples contabilização sobre estes exemplos.

Um conjunto de vetores apresenta *estruturas de terceira ordem* se o valor de certos componentes depende não do valor de outros componentes isolados, mas de valores conjuntos de *pares* de componentes. É o caso da função XOR: conhecer o valor de apenas um dos componentes de entrada não ajuda na determinação do valor do componente de saída, pois este não depende do valor dos componentes de entrada, mas sim do fato de eles terem ou não o mesmo valor. Os fatores de terceira ou maior ordem não podem ser simplesmente calculados através da contabilização de ocorrências conjuntas de duas

componentes, pois neste procedimento não se leva em consideração o valor das demais componentes. Considerando ainda o exemplo desta função, cujo conjunto de vetores é (0, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 0), onde as duas primeiras componentes de cada vetor são as entradas e a terceira é a saída: há claramente uma estrutura neste conjunto, pois metade dos vetores possíveis não ocorrem. Entretanto toda a estrutura é de terceira ordem; se contabilizarmos a razão entre co-ocorrências e ocorrências opostas entre uma componente de entrada e a componente de saída, verificamos que ela é exatamente 1: há duas co-ocorrências e duas ocorrências opostas, de forma que saber o valor de uma delas não favorece a determinação do valor da outra.

Como os modelos conexionistas representam as relações entre os componentes através de limiares e conexões entre as unidades, propícios a refletir apenas estruturas de primeira e segunda ordem, é necessário tratar grupos de exemplos que contém estruturas de terceira ordem decompondo-as de alguma maneira em estruturas de menor ordem. Isto é tipicamente realizado através da adição de novas componentes.

No caso da função XOR, por exemplo, se adicionarmos uma componente que assume valor 1 quando as duas componentes de entrada forem 1, teremos (0, 0, 0, 0), (0, 1, 1, 0), (1, 0, 1, 0), (1, 1, 0, 1). Neste caso, surgem estruturas de segunda ordem, entre as componentes de entrada e a quarta componente e entre a componente de saída e a quarta componente. Com a simples contabilização das co-ocorrências entre componentes podemos resolver o problema.

Entretanto, não é simples determinar quais componentes devem ser introduzidas para possibilitar o aprendizado pela contabilização, o chamado aprendizado fácil. Determinar quais devem ser estas componentes é o chamado *aprendizado difícil*. Nas redes conexionistas, a introdução de novas componentes é possível com a utilização de unidades ocultas. O aprendizado difícil nas redes conexionistas é determinar quais devem ser os pesos destas unidades para que elas passem a representar as componentes “certas” (Hinton & Sejnowski, 1986).

Na época em que Minsky e Papert escreveram *Perceptrons*, sabia-se que a utilização de novas unidades em camadas adicionais poderia resolver o problema da discriminação linear, mas isto não os tornava mais confiantes em relação a modelos multi-camada na superação das demais limitações. Além disso, não se conhecia um procedimento de aprendizado para as extensões em mais de uma camada. No modelo simples, o aprendizado estava baseado no fato de que para as unidades de saída os pesos eram determinados pelo erro calculado através da diferença entre as saídas real e desejada. Mas como fazer aprender uma unidade, em uma camada interna, que não estivesse calculando uma saída do sistema? Qual seria a resposta desejada para uma tal unidade?

Estas constatações levaram a uma grande estagnação do campo durante a década de 70, devido principalmente ao escasseamento de financiamento para pesquisas relacionadas. Entretanto, algumas pesquisas prosseguiram, com resultados notados principalmente na década seguinte.

2.3 Aprendizado em redes de múltiplas camadas: *backpropagation*

No início da década de 80 houve uma grande retomada das redes neurais, com a apresentação de diversos resultados interessantes. Alguns destes resultados estava contido no trabalho em dois volumes do chamado PDP Group (PDP para Paralell Distributed Processing) (Rumelhart & McClelland, 1986). Estes dois volumes traçavam também algumas linhas gerais das idéias conexionistas, tornando-se uma referência básica para o campo. Foi a partir desta época, também, que nomes menos comprometidos com a neurologia, como PDP e conexionismo, passaram a ser usados, enfatizando esta classe de modelos como uma abordagem aos processos cognitivos em geral ao invés de uma classe de modelos do sistema nervoso.

Foi no início da década de 80 que se tornou popular a solução para o aprendizado em perceptrons com múltiplas camadas (Rumelhart & McClelland, 1986; Werbos, 1995). Este método de aprendizado é chamado *backpropagation*, pois ele se baseia na propagação dos erros a partir da camada de saída (onde eles são facilmente conhecidos devido à saída desejada apresentada) para as camadas anteriores. Pode-se entender intuitivamente o processo considerando que o erro de uma unidade oculta (ou seja, que não está na camada de saída da rede) é calculado somando-se o quanto ela contribuiu para os erros da camada seguinte. Os erros das unidades da camada seguinte são propagados para esta unidade oculta na mesma medida em que estão ligadas, ou seja, proporcionalmente aos pesos das conexões entre elas. Se a camada seguinte é a camada de saída, seu erro é conhecido; caso contrário, será uma camada oculta cujo erro poderá ser calculado da mesma maneira.

Tecnicamente, este processo é realizado simplesmente alterando o peso de cada conexão na direção que torna o erro menor para o conjunto de exemplos fornecido. Isto é feito calculando-se a derivada do erro para cada peso e alterando-o na direção oposta à derivada. Com isto, há um movimento no espaço dos pesos para a rede em questão sempre em direção a um erro menor, até que se atinja um mínimo local.

Seguem-se as equações que resumem o processo, em uma versão um pouco mais geral do que a apresentada intuitivamente, que fixa um determinado número de camadas. Ao invés de ter-se unidades que recebem entradas apenas das unidades das camadas anteriores, teremos uma rede em que as unidades estão em seqüência e cada uma recebe como entradas as ativações de todas as unidades anteriores. Isto inclui as unidades de saída, que recebem como entrada as ativações de todas as unidades *inclusive* as das outras unidades de saída anteriores a si. Naturalmente, isto não impede que o esquema por camadas, mais comum, seja usado, pois ele é a versão particular em que as conexões que não vão de uma camada para outra estão fixadas em zero (Werbos, 1995). Esta versão apresenta as vantagens de não ser necessário fixar a priori o número de camadas da rede, pois isto não é simples, ou permite que se fixe a topologia da rede de maneira particular e adequada ao problema em questão.

As equações usadas são as seguintes:

$$x_i = E_i, \quad 1 \leq i \leq m$$

$$v_i = \sum_{j=1}^{i-1} w_{ij} \cdot x_j \quad m < i \leq N + n$$

$$x_i = \theta(v_i) \quad m < i \leq N + n$$

$$S_i = x_{i+N} \quad 1 \leq i \leq n$$

onde dispõe-se de $N + n$ unidades: m de entrada, N de processamento (incluindo aí as de entrada para facilidade de notação) e n de saída. O valor de ativação de cada unidade i é x_i , e este valor será igual ao valor da entrada E_i para as unidades de entrada. O resultado da integração das entradas da unidade i , de acordo com seus pesos w_{ij} , será v_i , e o valor x_i de ativação da unidade i que não for de entrada será $\theta(v_i)$, onde θ é uma função de transferência diferenciável, normalmente a sigmóide $\theta(v) = 1/(1 + e^{-v})$, mostrada na figura 2-7. O vetor de saída S é definido como os valores de ativação das últimas n unidades.

figura 2-7: A função sigmóide $\theta(v) = 1/(1 + e^{-v})$ é comumente usada como função de transferência em redes com aprendizado backpropagation. Duas de suas vantagens são os fatos de que ela é diferenciável e que sua derivada pode ser expressa em termos da própria função: $\theta'(v) = \theta(v) \cdot (1 - \theta(v))$.

O fato de cada unidade ter como entrada valores de ativação apenas de unidades anteriores a ela é uma restrição importante, pois permite o cálculo exato das derivadas da função do erro quadrático para um conjunto de treinamento em apenas um passo através da rede, como será mostrado adiante.

Baseados em vetores $D(t)$ contendo a saída desejada para o elemento t de um conjunto de treinamento com T elementos e chamando a saída real para este mesmo elemento de $S(t)$, podemos calcular as derivadas do erro quadrático total E para todo o conjunto em relação

a cada peso. Se usarmos a notação d_A como a derivada do erro em relação à variável A , podemos escrever:

$$E = \sum_t^T \sum_{i=1}^n \frac{(D(t)_i - S(t)_i)^2}{2}$$

derivando esta equação, temos

$$d_S(t)_i = S(t)_i - D(t)_i$$

$$d_x(t)_i = d_S(t)_{i-N} + \sum_{j=i+1}^{N+n} w_{ji} \cdot d_x(t)_j \quad i = N+n, \dots, m+1$$

onde $d_S(t)_{i-N}$ é definido como 0 caso $i \leq N$, ou seja, para as unidades que não são de saída. Note que se a unidade for de saída, há o erro proveniente da resposta desejada para aquela unidade e o erro proveniente das outras unidades de saída de índice maior do que o dela, para as quais seu valor de ativação contribui como entrada. Note também como o índice varia dos valores maiores para os menores; o erro é propagado na rede no sentido contrário ao cálculo de saídas, daí o nome *backpropagation*.

Continuando, pela regra da cadeia:

$$d_v(t)_i = \theta'(v_i) \cdot d_x(t)_i \quad i = N+n, \dots, m+1$$

para, finalmente,

$$d_w_{ij} = \sum_{t=1}^T d_v(t)_i \cdot x(t)_j \quad i = m+1, \dots, N+n; \quad j = 1, \dots, i-1$$

De posse de d_w_{ij} , podemos calcular um novo valor para w_{ij} da seguinte forma:

$$\text{Novo } w_{ij} = w_{ij} - \text{taxa de aprendizado} \cdot d_w_{ij}$$

A taxa de aprendizado é geralmente um valor pequeno, para que o processo se aproxime da idealização contínua mas, se ela for pequena demais, naturalmente o processo será muito lento. Por isso, seu valor será o maior possível sem que a rede deixe de adaptar seus pesos corretamente. Esta é uma escolha geralmente empírica, que depende de cada caso.

Com este processo, são formadas representações internas úteis nas unidades ocultas da rede, representações necessárias para transformar a entrada de forma que a camada de saída possa separar as classes apresentadas linearmente. A formação destas representações depende em certo grau dos valores iniciais dos pesos da conexões, escolhidos aleatoriamente dentro de uma faixa de valores pequenos. Muitas vezes estas

representações internas são mais eficientes do que inicialmente se supõe (Rumelhart & McClelland, 1986).

É interessante notar como, embora o sinal sobre o qual se calcula as derivadas seja normalmente o erro em relação a saídas desejadas, é necessário apenas que se tenha uma medida que se queira minimizar e cuja derivada possa ser calculada em relação aos pesos da rede. Uma aplicação deste princípio já foi usada para forçar a generalização da solução obtida, por exemplo, minimizando uma soma dos erros com uma medida de simplicidade da rede (Werbos, 1995).

Aqui cabem algumas observações sobre os pontos falhos da backpropagation. Nem sempre as representações internas necessárias são formadas, devido ao caráter aleatório dos pesos iniciais. Nestes casos, muitas vezes é necessário variar parâmetros como tamanho e taxa de aprendizado de forma *ad hoc*. Considerando o problema em um outro nível, a minimização do erro quadrático no espaço de pesos das conexões possui o inconveniente de todo método de regressão linear: pode ser atingido um mínimo local que não é o global. Além disso, não há indicações a priori sobre qual deva ser o tamanho da rede, nem quantas camadas ela deveria ter (quando usamos uma rede em camadas). Sua aplicação tem portanto um forte fator empírico. Ainda assim, o aprendizado por backpropagation é bastante eficiente e pode tratar conjuntos complexos de dados. Ele é, de longe, o mais usado em aplicações práticas, quase sempre em classificação de padrões.

2.4 Auto-associadores

Outros modelos conexionistas oferecem recursos diferentes daqueles oferecidos por redes com aprendizado backpropagation. Alguns deles enquadram-se na categoria das redes conexionistas auto-associativas. Estas redes caracterizam-se por não possuir conjuntos de unidades para entrada e saída específicos. Ao contrário, todas as suas unidades visíveis podem ser interpretadas como um único padrão, que será completado quando um subconjunto qualquer das unidades visíveis receber ativação, como mostra a figura 2-8.

Este completamento se dá de acordo com as conexões da rede. Uma conexão positiva entre duas unidades indica que elas tendem a estar ativadas simultaneamente, enquanto uma conexão negativa fará com que a ativação de uma iniba a da outra. Assim, o conjunto inicial de ativações se propaga através das unidades gerando um padrão de acordo com a informação inicial. Incorreções e ruídos tendem a desaparecer neste processo, porque em geral todas as unidades têm conexões com todas as outras, e incorreções da minoria são sobrepostas pela correção da maioria.

figura 2-8: Auto-associadores são redes que, ao invés de receberem uma entrada e calcularem uma saída, recebem partes de um único padrão e o restauram de forma a completar, dentre os padrões que foram aprendidos, aquele mais próximo do padrão inicial recebido. Neste exemplo, uma rede que tivesse aprendido a representação de algumas letras poderia restaurar o padrão inicial para a letra "A," a mais próxima do padrão inicial.

O sinal e intensidade das conexões são determinados durante o aprendizado que, diferentemente da backpropagation, não se baseia no erro, mas no reforço e enfraquecimento de conexões baseando-se em exemplos. Eles são apresentados e uma conexão entre duas unidades é reforçada quando ocorre uma ativação conjunta destas duas unidades e enfraquecida quando estas ativações diferem. Assim, duas unidades que ocorrem normalmente juntas nos exemplos do conjunto de treinamento terão uma conexão positiva e tenderão a estar ativadas ou não ativadas conjuntamente quando mais tarde a rede for usada. Quando duas unidades ocorrem em ativações opostas na maioria dos exemplos, sua conexão tornar-se-á negativa e elas tenderão a estar em ativações opostas dali por diante. Unidades razoavelmente independentes uma da outra terão uma conexão próxima de zero. Esta é uma versão generalizada das observações que Hebb fez a respeito do aprendizado no sistema nervoso, pois cobre também as ativações negativas ou nulas, e não apenas as ativações positivas como ele originalmente estipulou. Por este motivo é chamado algumas vezes de *aprendizado Hebbiano* (Hebbian learning).

Outra peculiaridade das redes auto-associativas é que normalmente elas são *recorrentes*, ou seja, uma vez fornecido um padrão inicial, os estados de ativação das unidades em um instante são utilizados para o cálculo de sua ativação no instante seguinte, até o eventual equilíbrio. Isto possibilita o interessante efeito de uma parte do padrão, embora não imediatamente preenchida a partir das informações iniciais disponíveis, ser preenchida em passos seguintes, quando outras partes forem preenchidas e fornecerem informações adicionais para seu preenchimento.

Um exemplo típico das aplicações das redes auto-associativas é o da restauração de imagens. Um conjunto de imagens é aprendido e, a partir de qualquer trecho de qualquer uma delas, recupera-se a que mais provavelmente contém o trecho apresentado. Também é usada como uma memória endereçada por conteúdo (em oposição à memória convencional em computadores, que necessita de um endereço de um item para encontrá-lo), quando uma parte do conteúdo procurado em um banco de memória é fornecido como chave da busca e o restante é completado. Por exemplo, pode-se procurar entre os dados de animais em um banco de memórias a partir de algumas características como "voa," "tem pêlos," "é pequeno." Estas propriedades (ou micro-características, como

explicado adiante na seção), estão associadas a outras que tornam-se ativadas ou inibidas, completando o padrão que caracteriza o animal procurado.

2.4.1 Brain-State-in-a-Box

Um dos primeiros modelos de redes auto-associativas foi desenvolvido ainda nos anos 70, por John A. Anderson (Anderson, 1995). Trata-se do BSB, sigla de Brain-State-in-a-Box. Seguindo os princípios de redes auto-associativas descritos acima, as unidades da rede de Anderson podem assumir valores de ativação contínuos, dentro do intervalo $[-1, 1]$. Isto é obtido com a função de transferência mostrada na figura 2-9(a). Desta maneira, a rede pode ser vista como um vetor de ativações dentro de um hipercubo. Através das atualizações síncronas (ou seja, todas as unidades têm suas ativações calculadas e só depois disso atualizadas) contínuas, o vetor navega dentro deste hipercubo até atingir um de seus vértices, como exibido na figura 2-9(b), que torna clara a escolha do nome deste modelo.

figura 2-9: (a) Função de transferência da rede BSB, com valores contínuos entre -1 e 1 e truncados neste limite, e (b) visualização da mudança dos estados de ativação da rede até o equilíbrio em um dos vértices do hipercubo formado pelos limites -1 e 1 das ativações.

O vetor que representa o vértice atingido será, normalmente, um dos padrões usados para o treinamento da rede e formação da matriz. Entretanto, em função do grau de dependência entre os padrões apresentados, pode haver o que se chama *padrões espúrios*, padrões não apresentados durante o treinamento mas que, uma vez atingidos, mostram-se estáveis.

O cálculo de cada estado da BSB é baseado no estado anterior e usa a matriz de conexões A , onde A_{ij} é o peso da conexão da unidade i para a unidade j , conforme a equação abaixo:

$$X_i = \theta(\alpha A X_{i-1} + X_{i-1}),$$

onde θ é a função de transferência, X_i é o i -ésimo estado da rede e α é uma medida de progressão da rede, um passo, que deve ser escolhido pequeno o suficiente para que a rede convirja para um estado mais próximo do estado inicial mas grande o suficiente para que o processo não seja muito lento.

A matriz A pode ser calculada a partir de um conjunto de treinamento de modo direto, a partir da soma dos produtos externos dos exemplos, dois a dois:

$$A = \sum_{x,y \in T} x' \cdot y,$$

onde T é o conjunto de treinamento e x' é sua transposta.

A matriz pode, após a sua formação, ser ajustada a partir de exemplos usando-se a regra Delta, aumentando sua capacidade de armazenamento. Uma rede deste tipo com n unidades pode armazenar perfeitamente até n padrões ortonormais sem a regra Delta, e n padrões ortogonais com a regra. Além deste limite a rede continua funcionando, mas com certo grau de degradação.

2.4.2 Rede de Hopfield

Um auto-associador que se tornou muito popular foi apresentado por Hopfield em 1984. Bastante semelhante à BSB, a rede de Hopfield guarda porém algumas diferenças em relação à primeira: suas unidades só assumem valores de ativação iguais a 0 ou 1 (mais tarde seria desenvolvida a rede de Hopfield com valores contínuos); não há conexões de uma unidade para ela mesma; e a atualização era assíncrona, ou seja, apenas uma unidade era atualizada a cada passo. A cada momento, uma unidade é escolhida aleatoriamente e seu valor de ativação é atualizado de acordo com seus pesos e valores das demais unidades. Estas características tornavam o modelo apto a ser implementado facilmente em hardware, o que contribuiu ainda mais para sua popularidade.

O aprendizado na rede de Hopfield é similar ao da BSB: a conexão entre duas unidades sofre um decréscimo para cada exemplo em que as ativações das unidades sejam diferentes e um acréscimo para cada exemplo em que elas estejam iguais, sendo portanto sempre simétrica.

Um dos resultados mais importantes apresentados por Hopfield é o de que qualquer rede recorrente desta natureza converge para um estado de equilíbrio, ou seja, um estado onde nenhuma unidade muda mais de valor. Isto é mostrado através de uma analogia entre a rede conexionista e um sistema termodinâmico, estabelecendo-se uma função *energia* da rede que diminui a cada atualização. O estado de equilíbrio atingido pela rede é um estado de energia mínima local, ou seja, um estado cujos vizinhos têm todos energia maior ou

igual à dele. Este padrão normalmente é um dos padrões usados para treinamento, embora ocorram padrões espúrios.

Esta concepção da rede como um minimizador de energia estimulou sua exploração em um novo ramo de aplicações, o da otimização combinatória. Para resolver um problema em que houvesse um custo a ser minimizado, desenhava-se uma rede onde o estado das unidades representasse os valores das variáveis envolvidas, e determinava-se os pesos das conexões de forma que as restrições entre os valores das variáveis fossem expressas por elas. Além disso, a função custo deveria estar codificada de maneira que quanto maior o custo da solução representada, maior a energia da rede. Ao deixar o modelo partir de um estado em que apenas as condições que definissem determinada instância do problema estivessem representadas nos estados das unidades, a energia diminuía gradativamente até atingir um mínimo.

Esta abordagem possui algumas desvantagens. Pode acontecer de a rede violar algumas restrições em favor de outras, ou em favor do custo mais baixo. Por isso, suas restrições devem ser consideradas como *suaves*, ou seja, restrições que podem ser violadas em pequeno número. Naturalmente, nem todo problema pode ser especificado através de restrições suaves. Para aqueles que são definidos através de restrições *rígidas*, entretanto, pode-se usá-la até que uma solução válida seja encontrada, o que muitas vezes é viável.

Outra desvantagem é que a rede pode atingir um mínimo local ao invés de global, significando que, mesmo que a solução encontrada seja válida, pode não ser a melhor. Entretanto, há casos em que apenas uma boa solução é necessária.

2.4.3 Máquina de Boltzmann

Embora auto-associadores como BSB e Hopfield estabilizem-se em estados de baixa energia, nem sempre eles serão pontos de energia mínima global. Isto pode ser problemático em determinados casos. Além disso há ainda, na utilização de auto-associadores como a BSB e Hopfield, o problema de não haver unidades ocultas na rede, ou seja, unidades que não pertencem aos padrões de entrada e saída, mas auxiliam o processamento. Isto impossibilita o aprendizado completo de um padrão se, entre suas unidades de saída, alguma necessite calcular uma função não linearmente separável em função de todas as demais.

Os problemas do mínimo local e o da falta de unidades ocultas nestes auto-associadores forneceram a motivação para o desenvolvimento de uma sofisticada variante da rede de Hopfield, a máquina de Boltzmann. Sua principal diferença em relação à precursora é que suas unidades são *estocásticas*. Isto significa que o que seria o valor de ativação na rede de Hopfield torna-se aqui a *probabilidade* de a unidade estar ativa (valor 1, em oposição ao valor 0 quando inativa). Uma das implicações disto é que esta rede nunca está em equilíbrio estático, mas alterna-se permanentemente entre as diversas configurações possíveis (embora algumas configurações sejam muito mais prováveis do que outras).

A máquina de Boltzmann inspirou-se na termodinâmica para realizar um processo conhecido como *simulated annealing*. Este processo é análogo ao processo de uma barra de metal resfriando-se. Durante este resfriamento, os átomos do metal mudam constantemente sua organização, passando por estados de diferentes energias. A altas temperaturas, estas mudanças de estado são muito rápidas e é relativamente fácil passar de um estado de baixa energia para um de alta. Conforme a temperatura vai baixando, porém, torna-se cada vez mais difícil mudar de um estado de baixa energia para um de mais alta, até chegar um ponto em que a organização passa a apenas diminuir de energia e estabilizar-se no mínimo local. Entretanto, se a velocidade do resfriamento foi suficientemente baixa, há boas chances deste mínimo local ser também o mínimo global, pois a temperaturas mais altas o metal terá passado rapidamente por diversos estados e terá apresentado a tendência de permanecer naqueles de energia mais baixa.

Um bom exemplo para visualizar este processo é imaginar uma bola sobre um daqueles suportes de papelão onde comumente se vendem ovos. Suponha que este suporte possua diversas cavidades de diferentes profundidades. Imagine que inicialmente alguém passe a agitar violentamente o suporte, de modo que a bola pule de cavidade em cavidade freneticamente, e depois comece gradualmente a diminuir esta agitação. A bola irá sucessivamente perder a facilidade de passar a outra cavidade; em particular, terá mais dificuldade de escapar das cavidades mais profundas. Se ela escapar de uma cavidade mais profunda e a agitação mantiver-se no mesmo nível por algum tempo, então ela provavelmente terá tempo suficiente para voltar àquela cavidade ou encontrar outra ainda mais profunda. No final do processo, haverá grande probabilidade de ela estar na mais profunda cavidade do suporte, tão grande quanto menor a velocidade de decréscimo da agitação empregada (Maren, 1990).

Na máquina de Boltzmann procuramos também uma energia mínima, de acordo com um conjunto inicial de unidades cujo valor é fixado, onde a energia corresponde ao nível de insatisfação das restrições representadas pelos pesos das conexões. Para utilizar o princípio de simulated annealing precisamos de algo similar ao efeito de resfriamento, o que é conseguindo adicionando-se um parâmetro ao sistema chamado, de modo intuitivo, *temperatura*. Para que a analogia funcione, é preciso que a altas temperaturas a probabilidade de uma unidade mudar de um estado de menor para maior energia é relativamente mais alta. Dos dois estados possíveis de uma unidade, o de maior energia será aquele que estará “contrariando” o seu potencial de ativação (o produto escalar entre seus pesos e suas entradas), ou seja, 0 se o potencial for positivo e 1 se for negativo. Para que a altas temperaturas seja mais fácil uma unidade contrariar seu potencial de ativação, convencionou-se sua probabilidade de estar ativa como sendo

$$P_{ij} = \frac{1}{(1 + e^{-v/T})}$$

onde v é o potencial de ativação da unidade, seu produto escalar entre seus pesos e suas entradas, e T é o parâmetro temperatura. De fato, esta é a mesma função sigmóide usada

na rede backpropagation, mas usada sobre v/T ao invés de simplesmente v . Pode-se interpretar esta aplicação como a aplicação usual sobre o potencial de ativação, mas com a inclinação da função dependendo de T . Quanto mais alta a temperatura, menos inclinada a sigmóide, e maior a probabilidade de a unidade assumir um valor contrário ao seu potencial de ativação (figura 2-10). Conforme a temperatura é suavemente baixada, as variações do estado da rede vão tornando-se menores, de acordo com a analogia anteriormente citada, e os estados de menor energia (de acordo com o sinal do potencial de ativação) vão tornando-se cada vez mais prováveis. Idealmente, a rede inteira atinge a situação de alternar-se uniformemente entre os estados de menor energia. Embora as condições ideais não possam ser atingidas nas implementações, observa-se que estados de baixa energia são de fato obtidos (Aarts, 1995).

figura 2-10: A função sigmóide é usada para determinar a probabilidade de uma unidade estar ativa na máquina de Boltzmann. A altas temperaturas as probabilidades são relativamente menores, tornando mais freqüentes as mudanças para estados mais energéticos.

Note que se nunca utilizarmos um subconjunto das unidades para entrada ou saída, elas funcionarão como unidades ocultas. Isto possibilita que representações internas úteis sejam usadas para contornar possíveis não linearidades presentes no problema. O aprendizado na máquina de Boltzmann comporta a presença de tais unidades, fazendo com que elas passem a ser usadas para representações internas necessárias. Para tanto, ele não poderia ser tão simples quanto o produto externo usado em Hopfield e BSB, que exige o valor de todas as unidades para ser efetuado. Também não poderia utilizar o princípio da backpropagation, pois a máquina de Boltzmann pode apresentar laços, o que não pode ocorrer para que o cálculo das derivadas do erro pelos pesos esteja bem definido.

Para compreender melhor o aprendizado na máquina de Boltzmann, façamos algumas considerações. O objetivo é que a máquina aprenda sobre um determinado ambiente, do qual ela recebe informação a partir de suas unidades visíveis (ou, neste contexto, *ambientais*). Este ambiente exhibe, ao longo do tempo, uma determinada distribuição de valores sobre estas unidades ambientais. Esta distribuição é independente dos pesos das

conexões, pois a ativação das unidades visíveis parte do ambiente. A máquina terá aprendido sobre este ambiente se, na situação em que nenhuma unidade ambiental esteja com valor fixado (uma situação que poderíamos chamar de *livre*), ela exiba ao longo do tempo a mesma distribuição sobre as unidades visíveis que o ambiente exibe. Se ela atingir este estado, funcionará também quando algumas unidades ambientais tiverem os valores fixados, alternando-se entre os vetores ambientais (os vetores formados pelas unidades ambientais) compatíveis com a informação fixada (Hinton & Sejnowski, 1986).

Portanto, nosso objetivo é que a rede, em situação livre, exiba a mesma distribuição de vetores ambientais que o ambiente. Há uma medida, proveniente da teoria da informação, devida a Kullbach (Kullbach, 1959) e que fornece a distância entre as duas distribuições:

$$G = \sum_{\alpha} P^{+}(V_{\alpha}) \ln \frac{P^{+}(V_{\alpha})}{P^{-}(V_{\alpha})}$$

onde $P^{+}(V_{\alpha})$ é a probabilidade do estado α das unidades visíveis durante a fase chamada *positiva*, quando elas são fixadas nos valores de treinamento com duração proporcional à sua probabilidade no ambiente que se quer modelar, e $P^{-}(V_{\alpha})$ é a probabilidade do estado α das unidades visíveis durante a fase chamada *negativa*, quando deixa-se a rede alternando-se livremente através dos diversos estados, sem unidades visíveis fixas.

G é de tal forma que nunca será negativa e será igual a zero somente se as duas distribuições forem idênticas. Portanto, o processo de aprendizado procura diminuir esta distância alterando cada peso w_{ij} da rede na direção oposta a $\partial G / \partial w_{ij}$.

Calcular a derivada de G para um determinado peso, porém, não é simples, pois ela depende dos valores de todos os outros pesos. É possível mostrar, porém, que

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{T} (p_{ij}^{+} - p_{ij}^{-})$$

onde p_{ij}^{+} é a probabilidade das unidades i e j estarem ativas ao mesmo tempo durante o equilíbrio termal na chamada *fase positiva*, quando a rede é governada pelos vetores do ambiente sendo fixados nas unidades visíveis, e p_{ij}^{-} é a mesma probabilidade durante o equilíbrio termal na chamada *fase negativa*, quando a rede é deixada a percorrer livremente seus estados, sem unidades visíveis fixadas.

Uma das interpretações para esta equação é que estamos contabilizando quanto das ativações da rede são devidas ao ambiente (influyente na fase positiva) e quanto estas ativações não são devidas à própria configuração interna atual da rede (dominante na fase negativa), minimizando os efeitos da segunda para cada conexão e tornando a rede cada vez mais semelhante ao ambiente.

2.5 Outros modelos

Os modelos que apresentamos até aqui são os mais comuns ou que representaram avanços importantes no campos das redes conexionistas. Entretanto, há muitas outras formas de conectar unidades processadoras simples, com parâmetros, funcionamento e topologia peculiares, visando a algum problema ou vantagem em especial.

Em geral, as variações das redes conexionistas buscam melhorar seu desempenho em algum problema específico, aproveitando as restrições existentes em determinado domínio, ou melhorar a modelagem de determinado sistema biológico, quando é esta modelagem o principal objetivo do pesquisador.

Unidades processadoras funcionando em tempo contínuo, utilizando funções de transferência especiais, recebendo entradas espaço-temporais, redes divididas em módulos especializados e outras são apenas algumas das variações que buscam uma melhoria em determinados aspectos.

Para as redes auto-associativas, por exemplo, há uma variação importante chamada BAMs (Bidirectional Associative Memory). Desenvolvidas por Bart Kosko em meados da década de 80, elas funcionam como um auto-associador típico, com uma importante diferença: o padrão que restauram é formado por dois sub-padrões, um de entrada e um de saída, e não há conexões entre duas unidades de um mesmo sub-padrão. Ao receber um deles, a rede completa o outro a partir de uma matriz de conexões (aprendida da mesma maneira que nas redes de Hopfield ou BSB, mas zerando o peso das conexões que ligam unidades de um mesmo sub-padrão) e em seguida faz o mesmo no sentido contrário, recalculando e eventualmente corrigindo o sub-padrão inicial. Esta ressonância prossegue até a convergência para um dos pares de sub-padrões aprendidos. As BAMs podem ser estudadas com o auxílio de uma função de energia similar à de Hopfield, e pode ser mostrado que um conjunto inicial de estímulos vai convergir para o par pertencente ao conjunto de treinamento de que mais se aproxime.

Um exemplo típico de aplicação de uma BAM, por exemplo, seria a do mapeamento de imagens em um radar para um vetor de características dos objetos reconhecidos. Uma imagem inicial, distorcida e incompleta gera o correspondente vetor de características e o par converge para uma configuração consistente.

As BAMs são um claro exemplo de uma variação em função de um determinado domínio, no caso aqueles em que os padrões são naturalmente divididos em dois sub-padrões. Este domínio poderia ser tratado em uma rede de Hopfield ou BSB; entretanto, as BAMs incorporam o que se sabe a priori sobre o ambiente, exigindo menos memória e aumentando a velocidade e a estabilidade da rede.

2.5.1 Learning Vector Quantization (LVQ)

A classificação de padrões é uma das mais importantes e freqüentes aplicações das redes conexionistas, e não deixou também de dar origem a arquiteturas especiais. Além das redes já vistas, uma importante rede voltada para classificação de padrões é a *Learning Vector Quantization* (LVQ) (Kohonen, 1995; Ripley, 1996). Seu funcionamento é bastante distinto das outras redes, pois ao invés de aprender através de correções graduais baseadas nas saídas desejadas, as unidades da LVQ recebem já no início do aprendizado atribuições de qual classe representarão. A partir daí têm seus pesos ajustados de maneira a definir as fronteiras entre as classes, utilizando um conjunto de exemplos de padrões. O vetor dos pesos das conexões de uma unidade é também chamado seu *codebook*.

É importante notar que os codebooks das unidades estão no mesmo espaço que os padrões apresentados à rede e que se todos os vetores (os de entrada e os codebooks) forem tratados de forma a terem norma 1, a unidade que mais fortemente responderá a determinado padrão será aquela cujo codebook é o mais próximo ao padrão em questão, de acordo com as propriedades do produto escalar.

Na LVQ, um conjunto de unidades recebe codebooks aleatórios ou provenientes do conjunto de treinamento. Quando um padrão é apresentado à rede, apenas a unidade que mais fortemente responde a este padrão é analisada, no que se chama comumente *winner-takes-all*. Se a classe a ela atribuída for a mesma que a da entrada, haverá um reforço do codebook da unidade de maneira que ele se torne um pouco mais próximo à entrada apresentada, de acordo com uma taxa de aprendizado. Se por outro lado as classes da unidade e da entrada forem diferentes, o codebook será ajustado de forma que a unidade se distancie da entrada. Com isso, torna-se gradualmente mais provável que as unidades certas reajam àquela entrada da próxima vez.

O resultado disto é que os codebooks ficarão dispostos pelo espaço dos padrões de forma a cobrirem a área que reconhecem. Codebooks das unidades de uma mesma classe estarão agrupados e distribuídos uniformemente pelo volume que compõe sua classe. Fronteiras entre as classes serão definidas pela distância entre codebooks de classes distintas e vizinhas. Quando um novo padrão for apresentado, o codebook mais próximo a ele (e portanto o que tornar-se-á ativo) será um dos codebooks que cobrem a área de sua classe.

É sempre possível verificar a que tipo de padrão uma determinada unidade responderá mais fortemente simplesmente observando seu codebook. O codebook pode portanto ser considerado um *protótipo* da classe que a unidade representa, ou seja, o padrão “típico” daquela classe. Se a classe for representada por mais de uma unidade, os codebooks destas unidades serão protótipos distribuídos de forma uniforme pelo volume da classe.

A LVQ recebeu várias modificações, de maneira a melhorar seu desempenho. Ela se mostra bastante rápida e eficiente, embora se aplique apenas nos casos em que as classes sejam conexas em seu espaço. Casos mais complexos, em que estejam presentes fatores

de terceira ou maior ordem, necessitam de unidades intermediárias ou ocultas, que a LVQ não provê.

2.5.2 Classificação não-supervisionada

Dentro do domínio da classificação, temos um sub-domínio bastante peculiar, a *classificação não-supervisionada*, que também dispõe de modelos próprios para realizá-la. Até agora foram apresentadas aplicações de *classificação supervisionada*, quando a rede deve aprender a que classe um determinado padrão pertence baseada em exemplos daquela classe. Cada padrão do conjunto de treinamento estava portanto já definido como pertencente a uma determinada classe. A classificação não-supervisionada, por outro lado, é o particionamento de um conjunto de padrões onde cada um deles não está a priori inserido em nenhuma classe. Estas serão determinadas, ao contrário, de acordo com as similaridades presentes entre os padrões deste conjunto, de forma que os padrões pertencentes a cada classe sejam similares entre si.

A classificação não-supervisionada pode ser realizada por uma rede conexionista com *aprendizado não-supervisionado*. O aprendizado não-supervisionado, ao contrário do supervisionado, não ocorre pela adaptação dos parâmetros a uma resposta desejada, mas sim pela adaptação sucessiva de *unidades classificadoras*, de maneira que passe a haver uma unidade para representar cada classe em que os padrões sejam similares entre si.

Uma abordagem usada para fazer com que cada unidade represente um conjunto de padrões similares, sem que esteja definida a classe de cada padrão no conjunto de treinamento a priori, é o *aprendizado competitivo* (Rumelhart, 1986a). Esta técnica é bastante parecida com o LVQ, mas as unidades não estão definidas para uma determinada classe. O reforço é feito na unidade que mais fortemente reagiu à entrada (winner-takes-all), deixando as demais inalteradas. Com isto, cada agrupamento de padrões atrairá uma determinada unidade e será rotulado por ela, que passa a ser a única a reagir mais intensamente aos seus exemplares.

2.5.2.1 Topology-Preserving Map (TPM)

Outra modelo que realiza o aprendizado não-supervisionado e também foi idealizado por Kohonen é o Topology-Preserving Map (TPM) (Kohonen, 1983; Ritter, 1995). Como na LVQ, no TPM também há uma camada de elementos classificadores. O que se busca, porém, não é definir codebooks para as unidades classificadoras de maneira a cobrir classes de entradas pré-definidas, mas sim definir codebooks para as unidades de forma que as relações topológicas do espaço de entrada sejam preservadas no espaço de classificação (ou de saída), mesmo que a dimensão deste seja menor que a do espaço de entrada.

Também aqui há um mecanismo semelhante ao winner-takes-all, pois quando uma entrada é apresentada, apenas uma unidade, e seus vizinhos em menor escala, tornam-se

ativados. Isto é obtido através de conexões excitatórias entre uma unidade e seus vizinhos mais próximos, seguidas de conexões inibitórias entre esta mesma unidade e vizinhos um pouco mais distantes, no que se chama usualmente “função do chapéu mexicano.”

Supondo que os elementos classificadores p , q e r são os mais fortemente ativados respectivamente por entradas u e v e x distintas, a relação topológica que há entre estas entradas será preservada se, para determinadas medidas de distância d_e para o espaço das entradas e d_c para a camada classificadora,

$$d_e(u, v) \leq d_e(v, x) \Rightarrow d_c(p, q) \leq d_c(q, r)$$

Isto é obtido no TPM com um algoritmo semelhante ao do LVQ: há um elemento classificador que responde mais fortemente para determinado padrão e por isto ele é reforçado. A diferença aqui é que seus vizinhos topológicos também são reforçados, em medida diretamente relacionada à proximidade com ele. Temos, desta forma,

$$m_u(t+1) = m_u(t) + \alpha(t)(x(t) - m_u(t)), \text{ se } d_c(u_{x(t)} - u) < N(t)$$

$$m_u(t+1) = m_u(t), \text{ caso contrário}$$

onde $m_u(t)$ é o codebook da unidade classificadora u no momento t , $x(t)$ é a entrada apresentada neste momento, $u_{x(t)}$ é a unidade mais fortemente ativada pela entrada x no momento t , $\alpha(t)$ é o passo das alterações e $N(t)$ define uma vizinhança ao redor de $u_{x(t)}$ de acordo com d_c . Em aplicações práticas, é útil que $\alpha(t)$ e $N(t)$ decresçam monotonicamente.

Com isso, obtém-se um mapeamento do conjunto dos padrões para um espaço de menor dimensão cuja representação pode ser usada topologicamente da mesma maneira que a representação dos próprios padrões. Kohonen considera esta computação uma das mais importantes no processamento de informações, pois permite que *representações reduzidas* dos dados mais relevantes sejam usadas sem que as relações entre eles sejam perdidas.

2.5.2.2 Adaptive Resonance Theory (ART)

Um modelo de rede de aprendizado não-supervisionado desenvolvido em várias por Stephen Grossberg desde os anos 70 é a Adaptive Resonance Theory (ART) (Carpenter & Grossberg, 1995). A rede ART combina as características de reconhecimento de padrões dos perceptrons com um controle do processo que procurar equilibrar duas necessidades em sistemas que aprendem: *estabilidade e plasticidade*. A plasticidade é a capacidade de aprender novos padrões, se eles diferem muito daqueles já aprendidos. A estabilidade é a capacidade de manter o que já se aprendeu quando novos padrões são incorporados.

A rede ART é composta basicamente de duas camadas: a de entrada e a de classificação. Os vetores a serem aprendidos ou reconhecidos são representados na camada de entrada, enquanto a camada de classificação é formada pelas unidades que indicam a que classe pertence o padrão presente na camada de entrada.

A ressonância está presente na ART porque quando um padrão é apresentado, ativa uma das unidades da camada de classificação e, após isto, esta unidade é usada para “testar” o padrão de entrada corrigindo-o de acordo com suas expectativas. Estas expectativas correspondem ao protótipo representado pelos pesos das conexões da unidade ativa. O processo continua até uma estabilização (quando o padrão sucessivamente corrigido na camada de entrada continuar ativando a mesma unidade de classificação), que comprova que o padrão apresentado condiz com o que já foi aprendido sobre a classe em questão. Quando este reconhecimento ocorre, a entrada é usada ainda para refinar o protótipo da classe, com o reforço das conexões da unidade na direção da entrada.

Se não houver a estabilização no reconhecimento da ART, a rede utiliza uma unidade da camada de classificação que ainda não tenha sido usada para representar nenhuma classe, e reforça suas conexões para que o padrão apresentado seja por ela reconhecido da próxima vez, constituindo assim uma nova classe.

Todo este procedimento é complexo e efetivado apenas através de sinais de controle. Também é necessário um parâmetro chamado *vigilância*, que determina quão próximo deve estar um padrão do protótipo de uma classe para ser considerado pertencente àquela classe. Um parâmetro de vigilância muito alto leva a classes estreitas e não muito generalizadas, que aceitam apenas padrões muito próximos de seu protótipo, enquanto um mais baixo permite a formação de classes mais generalizadas.

Entretanto, a ART tem algumas desvantagens. Entre elas, o fato da ordem de apresentação dos padrões alterar quais classes são formadas e a maneira empírica como se deve escolher o parâmetro de vigilância. Trata-se de um modelo bastante específico, projetado levando-se em consideração o fato de ser destinado à classificação.

2.6 Síntese

No estudo de um certo número de modelos conexionistas como o realizado acima, percebe-se uma grande diversidade em relação a alguns detalhes. Algumas redes usam unidades contínuas, outras discretas; algumas são recorrentes, outras unidirecionadas; várias possuem unidades ocultas, outras apenas as visíveis. Os algoritmos de aprendizado podem ser baseados na correlação entre as componentes dos exemplos ou na adaptação em relação a um sinal de erro. Há mesmo modelos específicos que buscam reproduzir um padrão no tempo, enquanto a maioria busca gerar um padrão estático final. As representações podem ser binárias (valores das unidades no intervalo $[0, 1]$) ou bipolares (no intervalo $[-1, 1]$).

Há muitas variações possíveis para redes conexionistas, na verdade tantas que não é possível seu agrupamento em uma única formalização teórica. Isto torna difícil o estudo de suas propriedades de modo formal. Entretanto, pode-se estudar subclasses melhor delineadas, como no caso de redes recorrentes, compostas por neurônios de Pitts-McCulloch unidas por conexões simples e funções de transferência iguais ao sinal do valor de ativação, e com atualização síncrona (o tipo mais comum de rede). Para esta subclasse, de fato, está provado que as redes são equivalentes a autômatos finitos. Se for usada uma função de transferência que assuma valores contínuos em um intervalo, então a equivalência passa a ser em relação à máquina de Turing (Sontag, 1995). Esta equivalência, porém, é apenas teórica, por exigir uma precisão infinita (como é também apenas teórica a fita infinita da máquina de Turing).

De qualquer forma, pode-se identificar variações mais comuns e importantes na literatura. Algumas delas são presença ou não de unidades ocultas, presença ou não de laços na rede (recorrência), utilização de aprendizado supervisionado ou não supervisionado, unidades bipolares ou binárias.

Embora todas estas variações sejam importantes, apenas a presença de unidades escondidas realmente altera significativamente o conjunto de problemas que podemos abordar com as redes conexionistas (seção 2.2). Sem elas, é impossível tratar problemas não linearmente separáveis.

Em relação à opção quanto ao aprendizado ser supervisionado ou não supervisionado, pode-se perceber que eles não são mutuamente exclusivos. Sua distinção concerne principalmente ao desempenho. O aprendizado não supervisionado exige que um certo subconjunto das unidades identifique os padrões apresentados e que aquela que mais fortemente responder torne-se a única unidade ativada. Isto pode ser obtido através de conexões laterais inibitórias entre estas unidades em uma rede recorrente, pois quando uma delas receber um mínimo de ativação antes das outras, tornará ainda mais difícil que a ativação das concorrentes cresça e passará gradativamente a ser a única unidade ativada no conjunto. Estas conexões inibitórias, por sua vez, podem ser facilmente treinadas através de aprendizado supervisionado. Para formar uma conexão inibitória entre duas unidades, basta apresentar vários exemplos em que elas estejam com valores opostos a uma rede com aprendizado supervisionado por reforço. Por outro lado, se apresentarmos padrões formados pela concatenação de pares de entrada e saída a uma rede conexionista com aprendizado não-supervisionado, podemos obter a saída a partir da entrada, ainda que de forma não muito eficiente. Estes padrões serão armazenados (incidentalmente associados a uma das unidades classificatórias). Ao apresentarmos um dos pares completado apenas com a entrada, a unidade classificatória correspondente será usada e seu protótipo causará o completamento do padrão formando, de modo provável, a saída correspondente.

A diferenciação entre unidades bipolares ou binárias também não é fundamental. A escolha de uma ou de outra baseia-se normalmente apenas na representação mais ou

menos direta dos dados do problema tratado. Se uma unidade tem o potencial de ativação representado por

$$x_1w_1 + x_2w_2 + x_3w_3 + \dots + x_nw_n + t$$

na representação bipolar $(x_1, x_2, \dots, x_n)$ de um padrão, o mesmo potencial pode ser obtido para a representação binária $(y_1, y_2, \dots, y_n)$ com

$$(2y_1 - 1)w_1 + (2y_2 - 1)w_2 + \dots + (2y_n - 1)w_n + t =$$

$$y_1(2w_1) + y_2(2w_2) + \dots + y_n(2w_n) - (w_1 + w_2 + \dots + w_n) + t.$$

Outra variação é a de recorrência ou de feedforward (sem laços, como as redes que usam backpropagation). Uma rede recorrente (em um número limitado de passos) pode ser representada por uma feedforward com uma camada para cada etapa da recorrência. Uma rede feedforward pode ser implementada em uma rede “recorrente,” executando-se somente um passo ou fixando-se certos pesos de forma que os valores de ativação sejam propagados em apenas uma direção.

Se as redes conexionistas não podem ser agrupadas em uma única formalização teórica, podem no entanto ser agrupadas sob o fato de todas apresentarem três características principais que fundamentam seu funcionamento. No capítulo posterior sobre alto nível em redes conexionistas, estas características serão a base da argumentação, que apresentará assim generalidade quanto às redes conexionistas. Estas três características são o *processamento paralelo de unidades simples*, a *representação distribuída* e a *interpretação das redes como conjuntos de hipóteses com restrições fracas entre si*.

2.6.1 Processamento paralelo de unidades simples

O processo realizado pelas redes conexionistas é essencialmente paralelo. Todas as unidades podem calcular simultaneamente seus novos valores de ativação. Mesmo quando a atualização de suas unidades é assíncrona (ou seja, não ocorrem mudanças simultâneas de valores de ativação), esta atualização não tem uma ordem definida e necessária e nada impede que as unidades que não tenham sido atualizadas não estejam já há algum tempo realizando cálculos. Além das implicações quanto à implementação das redes, que fica facilitada em hardware paralelo, esta característica reflete-se também no fato de que durante o processamento efetuado pelas redes vários fatores são considerados ao mesmo tempo, tornando-as aptas à resolução de problemas que naturalmente exigem a integração de diversas restrições simultaneamente.

Outro fator de unificação é o fato de as unidades dos modelos conexionistas, bem como as conexões entre elas, serem relativamente *simples*. A saída das unidades são determinadas a partir de poucas operações básicas sobre as entradas (como é o caso das unidades mais comuns que multiplicam as entradas pelos seus respectivos pesos, somam

os resultados e aplicam uma função de transferência monotônica). As conexões possuem apenas a informação de inibição ou excitação entre duas unidades. Qualquer complexidade envolvida na tarefa é deixada mais a cargo da *interação* entre as unidades e conexões do que destes elementos propriamente. Isto traz efeitos importantes, como por exemplo a degradação suave; se nenhum elemento processador carrega consigo parte significativa do processamento, a alteração, ruído ou ausência de seu valor de ativação terá um efeito menor sobre o resultado final. Há vantagens também em relação ao aprendizado. A simplicidade inerente às unidades facilita os algoritmos de aprendizado que utilizem regressão linear, tornando possível o cálculo de suas derivadas em relação a algum sinal de feedback.

A simplicidade dos elementos dos modelos conexionistas, entretanto, podem ser vistas como um fator limitante. Não é claro se e como a interação entre eles pode levar a processos cognitivos mais complexos, como veremos no capítulo 3.

2.6.2 Representação distribuída

Em uma rede conexionista, o conhecimento sobre os elementos *no nível do problema* (conceitos, entidades, relações) pode estar representado em diversas unidades diferentes, *no nível da rede*. Por outro lado, uma unidade da rede pode contribuir para a implementação de mais de um dos elementos *no nível do problema*. Este tipo de representação é o que se chama de *representação distribuída*. Embora em certos modelos haja alguns elementos em representação local (uma unidade exclusiva para aquele elemento), redes com representações *apenas* locais dificilmente podem ser chamadas conexionistas, estando mais próximas de redes semânticas (Barnden, 1995b) em suas propriedades.

O compartilhamento de unidades entre diversos elementos e a presença de um elemento sobre várias unidades conduz à chamada *degradação suave*, ou seja, o desempenho da rede varia apenas gradualmente conforme são inseridas inexatidões em sua configuração ou entradas. Ela ocorre justamente porque muitas unidades no nível da rede representam parcialmente muitos elementos do problema, e portanto uma pequena alteração em um pequeno conjunto de unidades será compensado pelo grande número de unidades que restarem intactas.

A degradação suave manifesta-se sob a forma de mais de uma vantagem. Uma delas é a generalização automática, a capacidade de processar adequadamente entradas novas, apenas similares a entradas já processadas. Outra vantagem é a de as redes poderem recuperar padrões que não correspondem exatamente aos padrões fornecidos, o que é muito importante nas aplicações em ambientes imprecisos. Finalmente, nos casos de implementações diretas em hardware, a degradação suave torna a rede uma máquina bastante robusta, que pode ser danificada até uma certa extensão sem uma diminuição drástica de seu desempenho.

Os componentes dos padrões com representação distribuída são algumas vezes chamados de micro-características (*micro-features*). São elementos em um nível mais detalhado do que o dos objetos envolvidos no sistema, que os compõe quando reunidos em número suficiente para caracterizá-los. Os objetos em uma sala, por exemplo, podem ser portas, mesas, janelas, lustres etc. Estes objetos podem ser decompostos em características mais detalhadas, como “é plano,” “é opaco,” “é pesado,” “tem dobradiças” e assim por diante. Algumas destas características são as mesmas para mais de um objeto. A partir de uma delas não se pode em geral saber de qual objeto se fala; para caracterizar qualquer um deles é necessário fornecer um certo número destas micro-características. Esta decomposição em detalhes compartilhados por vários objetos propicia o estabelecimento de relações mais ricas entre eles, tais como similaridade, através do compartilhamento de micro-características. Objetos que seriam em uma abordagem tradicional representados por um símbolo atômico passam a ser representados a partir das interações ocorridas em um nível “sub-simbólico” (Smolensky, 1986).

A representação nas redes torna-se distribuída devido aos métodos de aprendizado utilizados. Neles, a adaptação ou o reforço empregados para cada exemplo são aplicados, em pequena medida, sobre todas as unidades, embora não na mesma intensidade (Hinton, 1986a). Desta maneira, com o passar do aprendizado cada uma delas incorpora informações sobre cada elemento do problema.

A principal desvantagem da representação distribuída é sua ininteligibilidade. O significado de cada unidade, por ser o resultado da integração de diversos elementos do problema, é bastante obscuro. Por isso o conhecimento presente na rede é normalmente implícito e não pode ser checado ou extraído facilmente.

2.6.3 Interpretação como conjuntos de hipóteses com restrições suaves entre si

Todas as redes podem ser interpretadas como um conjunto de hipóteses com restrições suaves entre si. O valor de cada unidade pode ser interpretado como um grau de certeza de determinada hipótese, baseado na entrada apresentada correntemente. As próprias unidades de entrada podem ser interpretadas como hipóteses sobre o ambiente. As unidades de saída representam estimativas feitas pela rede para hipóteses também sobre o ambiente, baseando-se no treinamento recebido. Nos casos das unidades de entrada e saída, a cada uma delas corresponde uma hipótese determinada a priori pelo próprio esquema de representação adotado, o que não ocorre com as unidades ocultas, cujo significado é definido ao longo do aprendizado e pode não ter correspondência com hipóteses diretamente formuladas no nível do problema.

As restrições entre as hipóteses são do tipo inibitória ou excitatória, ou seja, valores positivos nas conexões entre duas unidades indicam que as hipóteses que elas representam tendem a ocorrer juntas e a reforçar-se mutuamente. Valores negativos indicam que uma tende a inibir a outra. Quanto maior o valor absoluto da conexão, mais fortemente aquela restrição se aplica. São as restrições das hipóteses sobre o ambiente

representadas pelas unidades de entrada em relação a outras hipóteses que farão com que o estado deste ambiente se reflita no estado da rede, e que por sua vez fará com que este estado possa ser interpretado como um conjunto de hipóteses mais plausível frente à entrada corrente (Hinton & Sejnowski, 1986). Em um sistema feedforward, as hipóteses sobre a entrada são propagadas até que se estabeleçam os valores das hipóteses da saída. Em um sistema recorrente ocorre o mesmo, com a vantagem de todas as hipóteses poderem ser utilizadas, potencialmente, para refinar os valores umas das outras sucessivamente.

As restrições entre as hipóteses das redes conexionistas são chamadas *suaves* porque elas não são necessariamente obedecidas. Uma unidade pode ter seu valor de ativação calculado em concordância com a *maioria* de suas conexões, mas não *todas*. Isto ocorre em contraposição com restrições *rígidas*, em que o problema em questão só é considerado resolvido quando todas as restrições são atendidas. Nas redes conexionistas pode-se ter algo próximo também das restrições rígidas, representadas através de conexões bem mais fortes do que as outras, mas apenas no nível prático.

O fato das redes poderem encontrar soluções que violam algumas de suas restrições pode parecer à primeira vista uma desvantagem, mas quando empregadas em determinados problemas é de fato vantajoso. Uma boa parte dos problemas de Inteligência Artificial envolvem restrições suaves. Valores *default* ou utilização de contexto em resoluções são exemplos de fatores que podem ser caracterizados com restrições suaves. O problema de reconhecimento de padrões envolve muitas restrições suaves, pois é em geral raro que um padrão corresponda precisamente aos requisitos de sua classe. Os esquemas, ou frames, muito utilizados na década de 70 mas difíceis de caracterizar já naquela época, são mais facilmente tratados se considerados como construtos de restrições suaves (Rumelhart *et al.*, 1986b).

As unidades ocultas são uma questão à parte na interpretação de seus valores como a plausibilidade de uma hipótese. Enquanto as unidades visíveis têm uma clara interpretação, muitas vezes estabelecida pelo usuário da rede, as unidades ocultas não possuem uma definição a priori de significado. Isto no entanto não invalida a interpretação da rede como um conjunto de hipóteses. O que ocorre é que cada unidade oculta adquire uma definição de seu comportamento *durante* o treinamento, e sua interpretação fica condicionada *unicamente* a este comportamento, e não a um significado determinado a priori. Muitas vezes é possível, principalmente em redes pequenas, estabelecer qual hipótese uma unidade oculta passou a representar observando os pesos de suas conexões. Na maioria das vezes, entretanto, o significado da unidade não corresponde a nenhum conceito no nível do problema, embora seja útil à sua resolução.

Em suma, o que os modelos conexionistas oferecem ao apresentar estas três principais características é um tratamento da cognição com uma propriedade fundamental: os objetos envolvidos na resolução dos problemas não são fixos, com significado imutável. Ao contrário, são objetos que mudam sutilmente seu comportamento de acordo com o

contexto, e que podem estar presentes em variados graus, dependendo de quão nítidas estão as condições para seu surgimento. São objetos “emergentes,” ou seja, que surgem como resultado da interação de componentes muito mais detalhados. Suas definições exatas e fixas seriam apenas aproximações úteis à compreensão e interpretação, mas não fornecem um tratamento adequado. Para obter este tratamento adequado, segundo os connexionistas, é necessário descer ao nível mais detalhado das micro-características.

3. Alto nível em redes conexionistas

Embora uma rede conexionista possa aprender a realizar um mapeamento de um conjunto de padrões de entrada em um conjunto de padrões de saída apenas baseada em exemplos quando este mapeamento é de baixo nível, não é simples fazê-lo para pares de entrada e saída em problemas de nível mais alto. Parece improvável, por exemplo, apresentar diversos pares de listas onde a segunda lista é igual a primeira, porém ordenada, a um modelo conexionista e esperar que a rede possa passar a ordenar corretamente listas jamais recebidas.

Entretanto, seria altamente desejável proceder assim e obter bons resultados. Várias tarefas de alto nível, como por exemplo a compreensão de sentenças de linguagem, ou ainda a determinação de parentesco em uma árvore genealógica, exibem características que tornam desejável sua abordagem pelas redes conexionistas: aprendizado baseado em exemplos, integração simultânea de múltiplas restrições, degradação suave etc.

Embora a dificuldade das redes conexionistas no tratamento de problemas de alto nível seja largamente admitida e tida como clara, as razões para isto não o são. Conforme já dissemos, as redes conexionistas com valores de ativação contínuos são equivalentes à máquina de Turing. Isto poderia levar à conclusão de que elas são capazes de resolver problemas como o da ordenação das listas. Resta saber, porém, se é possível que elas executem qualquer processo computacional exibindo também as propriedades que as tornam interessantes. A discussão deste tópico, na verdade, está em andamento, e se dá em diversos níveis. No entanto, é possível levantar diversos pontos críticos, o que será feito nas próximas seções.

Um passo importante é procurar caracterizar melhor o que são processos de alto nível. A seção 3.1 discute algumas características normalmente atribuídas a estes processos que serão consideradas nas seções seguintes.

Todo um conjunto de autores, chamados geralmente de *Classicistas*, defendem a idéia de que as características da cognição de alto nível só podem ser reproduzidas através de sistemas de símbolos, e que os modelos conexionistas não podem fazê-lo a não ser *implementando* um sistema de símbolos (compondo, através de módulos, componentes do sistema como registradores, memória por endereço etc). Os Conexionistas dividem-se a respeito da questão em alguns graus. A seção 3.2 aborda esta discussão.

A seguir, na seção 3.3, será apresentado um importante instrumento na abordagem deste problema, o conceito de *níveis de descrição*.

A seção 3.4 apresentará os argumentos de uma e outra abordagem na discussão sobre qual é a mais apropriada à modelagem do nível cognitivo.

Os problemas para atender às necessidades do nível cognitivo por parte dos modelos conexionistas são apresentados na seção 3.5, bem como algumas das soluções usadas para contorná-las.

Finalmente, seguir-se-á na seção 3.6 uma síntese de tentativas conexionistas neste sentido que busca mostrar quais seus verdadeiros sucessos e o que elas deixaram de resolver.

O capítulo 4 buscará exibir um exemplo de como modelos conexionistas podem abordar problemas de alto nível.

3.1 O que é alto nível?

A discussão sobre o que é essencialmente um processo de “alto nível” é extensa e não será tratada aqui em todo o seu detalhe. Entretanto, há diversos pontos de consenso sobre no que consistem estes processos.

Barnden (1995) discute algumas das características que parecem ser intrínsecas a processos de alto nível. Sendo um conexionista, ele aponta aquelas mais dificilmente realizadas nestes modelos. Segundo este autor, eles parecem necessariamente envolver os seguintes tópicos:

1. Codificações altamente temporárias. Por exemplo, conclusões temporárias ou diferentes hipóteses devem ser rapidamente criadas ou destruídas durante um processo de raciocínio;
2. As técnicas de codificação devem permitir que itens de informação possam ser combinados, mesmo que tal combinação nunca tenha ocorrido antes nem mesmo de uma maneira similar. Por exemplo, em uma sentença de linguagem diversos conceitos podem surgir combinados de uma maneira nunca vista antes por aquele que a interpreta;
3. A complexidade das combinações dos itens de informação deve poder variar arbitrariamente. Sentenças de linguagem ainda são um bom exemplo também neste caso;
4. Em particular, as estruturas codificadas podem ser arbitrariamente aninhadas, como na frase “João espera que Maria acredite que José sabe de tudo.” Neste exemplo, uma afirmação sobre conhecimento está aninhada em uma afirmação de crença que está aninhada em uma afirmação de esperança;
5. Um determinado tipo de informação pode estar aninhada em diversos níveis de uma mesma estrutura. Por exemplo, uma estrutura representando uma sala pode conter a representação de um quadro que exibe uma outra sala;

6. Um mesmo tipo de informação pode estar instanciada de diferentes maneiras, como por exemplo em algum problema em que três instâncias de relacionamentos de amor devam ser consideradas simultaneamente.

Fodor & Pylyshyn (1988) levantam outros tópicos importantes que, segundo eles, devem estar presentes na cognição. São eles *produtividade*, *sistematicidade*, *composicionalidade* e *coerência inferencial*. Apesar de os autores não vincularem estes itens a alto nível necessariamente, trata-se de características que colocam dificuldades aos conexionistas quando estes querem atacar problemas de alto nível com seus modelos.

Produtividade é a propriedade que sistemas exibem quando são, *em princípio*, capazes de manipular um número infinito de elementos a partir de um conjunto finito de elementos primitivos. Esta propriedade pressupõe a distinção entre competência e desempenho: a primeira refere-se à capacidade infinita enquanto o último explica a limitação observada no mundo real em função da limitação de recursos. Falar da capacidade em princípio de lidar com um número infinito de elementos, separadamente dos recursos, equivale a dizer que o mesmo sistema poderia lidar com um número arbitrariamente grande destes elementos se recebesse recursos suficientes. Embora os autores acreditem que sistemas cognitivos são produtivos, admitem que esta é uma questão controversa; há os que acreditam que mentes não podem, nem mesmo em princípio, lidar com uma capacidade infinita de conceitos.

Sistematicidade é a propriedade que sistemas cognitivos devem apresentar de que, sendo capazes de manipular algumas representações, torna-se imperativo que manipulem outras. Um exemplo é “João ama a garota.” Um sistema capaz de manipular corretamente a representação de um tal conceito (seja ela qual for) deve, necessariamente, manipular a representação do conceito “A garota ama João.” Isto tem importantes consequências sobre as propriedades da representação escolhida (uma delas é que as representações devem ser formadas de *constituintes*), como será visto na seção 3.4.

Composicionalidade é uma propriedade exigida de sistemas cognitivos que caminha paralelamente à sistematicidade. Ela dita que representações ligadas pela sistematicidade devem ser próximas semanticamente. Embora possa à primeira vista parecer que o princípio da sistematicidade já traz este fato consigo, isto não é verdade. O princípio da sistematicidade é essencialmente sintático, determinando que certos conjuntos de representações são formadas por um conjunto básico de constituintes arranjadas de diferentes maneiras. A composicionalidade, entretanto, restringe as regras de composição destes elementos básicos de forma que o significado das composições levem a conceitos semelhantes e coerentes. Se o conceito “João ama a garota” estabelece uma relação unilateral entre João e a garota, então “A garota ama João” deve necessariamente levar ao estabelecimento desta *mesma* relação da garota em relação a João. Indo mais adiante, a composicionalidade tem como consequência a independência de contexto dos constituintes da representação *molecular* (a combinação dos constituintes, em oposição a representações *atômicas* sem constituintes internos). “João” deve ser isento

semanticamente do contexto em que estava inserido em cada uma das composições apresentadas, contribuindo da *mesma* forma nos dois casos. Este constituinte não poderia estar contribuindo por si só, na primeira sentença, com algo relacionado ao contexto, informando algo sobre a relação de amor de João em relação à garota ou informando algo sobre a própria garota. Isto deve ocorrer apenas em função da composição deste constituinte com os demais. Se ele informasse algo sobre a relação *por si só*, então a segunda sentença não seria análoga à primeira semanticamente, pois informaria sobre *duas* relações: a da garota para João e a de João para a garota (esta última, da mesma maneira que fazia na primeira composição), enquanto a primeira sentença só trata de *uma* relação.

Coerência inferencial é o último princípio apontado por Fodor & Pylyshyn. Este princípio diz respeito ao fato de que as inferências entre estados mentais devem ser coerentes com sua estrutura. Se a inferência de P&Q para P é realizada devido ao conhecimento de P e de Q, então a inferência de P&Q&R para P deve ser realizada pelos mesmos meios. O argumento é o de que não se pode ter mentes que possam deduzir P de P&Q mas não possam deduzir P de P&Q&R.

A partir destas observações, pode-se perceber três grandes necessidades no tratamento de problemas de alto nível: a necessidade de estruturas, a necessidade de processos temporais e de controle, e a necessidade de múltiplas instâncias simultâneas de objetos do mesmo tipo. Pode-se ainda procurar e encontrar outras necessidades, evidenciadas por determinados tipos de problemas e modelos, mas estas são as que mais se destacam.

O que há de comum entre estes mecanismos é que eles constituem *ferramentas para controle de complexidade*. Ao usar-se estruturas, pode-se falar independentemente das propriedades de uma estrutura e das propriedades de seus constituintes, não sendo preciso por exemplo saber quais elementos estarão em uma pilha ou árvore para descrever suas propriedades; quebrando-se descrições de procedimentos ou funções em passos temporais básicos comuns a várias delas, tem-se um conjunto de passos que podem ser combinados sem grande acréscimo de informação; criando-se uma classe que possa ser usada na descrição de mais de um objeto, concentra-se informação em um único ponto e diminui-se a complexidade.

Entretanto, isto não parece dizer tudo sobre alto nível pois, na verdade, estas características são importantes e úteis também em problemas de baixo nível. Pode-se identificar e utilizar estruturas em tarefas de baixo nível sobre imagens ou ondas sonoras, por exemplo. Padrões de baixo nível podem em alguns casos ser melhor reconhecidos se for possível realizar alguma operação temporal sobre eles. Outras tarefas de reconhecimento de padrões podem ser melhoradas se for possível reconhecer dentro de um padrão diferentes instâncias de uma classe de sub-padrões. Se é assim, então o que verdadeiramente caracteriza alto nível?

A questão de alto nível está ligada à idéia de níveis de descrição, tratada em detalhe na seção 3.3. Por ora, basta notar que um processo de alto nível é descrito em termos de

níveis mais baixos. Isto significa que as propriedades de todo um conjunto de termos já estão definidas a priori quando a descrição de um processo de alto nível é introduzida. Também isto é uma medida de controle de complexidade, pois os termos definidos em um nível mais baixo são usados repetidamente e de várias maneiras na descrição no nível mais alto. Se tivessem que ser definidos em todo seu detalhe a cada vez que fossem usados, a complexidade do processo seria aumentada grandemente. Em descrições de objetos de alto nível, os elementos da descrição por si só contêm pouca informação. Eles são informativos à medida que são examinados à luz de suas relações com níveis mais baixos.

Os processos de alto nível, portanto, dependem por definição de ferramentas de controle de complexidade como as apontadas acima. São essas ferramentas que fazem muitas vezes a ligação entre dois níveis: a lista, por exemplo, é uma estrutura; quando se fala de uma lista e da remoção de um certo elemento desta lista, está-se fazendo referência a um certo processo temporal; pode-se fazer referência a várias listas que compartilham propriedades por pertencerem à mesma classe.

Desta maneira, compreende-se melhor a função de tais ferramentas e a preocupação dos autores em delas dispor no tratamento de problemas da cognição de alto nível. Porém, como indicado acima, estas ferramentas não são tudo em alto nível. Elas podem ser usadas de forma proveitosa dentro de um mesmo nível para auxiliar o processamento de certo sistema, mesmo que de baixo nível, como no caso de tratamento de imagens ou ondas sonoras. Os processos de alto nível certamente contam com estas ferramentas, mas para serem considerados de alto nível devem ser descritos de forma estanque a partir de um nível mais baixo.

A importância da separação em níveis chama a atenção para mais uma característica do processamento de alto nível. Baseando-se em um nível mais baixo, a descrição de alto nível baseia-se em conhecimento prévio sobre este nível mais baixo. Um sistema que recebe uma descrição de alto nível terá que receber, em algum momento, informação sobre outros níveis. Isto supõe um aprendizado (ou pelo menos recepção de informação) por parte deste sistema em várias fases, visando níveis diferentes. Adiante mostrar-se-á que este é um fator importante no tratamento de processos de alto nível em redes conexionistas, já que estes modelos tradicionalmente não contam com aprendizado em mais de uma fase, ou seja, fases tratando de diferentes conjuntos de elementos e suas propriedades.

Assim, pode-se resumir que um problema de alto nível é aquele que em sua descrição e tratamento utiliza ferramentas de controle de complexidade para basear-se em um nível mais baixo de descrição, previamente conhecido.

Note que nada do que foi dito nesta seção sobre a cognição e o alto nível envolve necessariamente sistemas de símbolos. Entretanto, muitos defendem esta necessidade para a reprodução das propriedades apontadas, incluindo-se aí Fodor & Pylyshyn. Este tópico é tratado na próxima seção.

3.2 A questão da representação simbólica

Durante boa parte de sua existência a maioria das ciências que estudam a cognição, e entre elas a IA, baseou-se na hipótese de que representações simbólicas e operações sobre elas são o meio mais adequado para reproduzir processos inteligentes. Por isso, este grupo é comumente chamado *Classicista*, quando em comparação com correntes com outro ponto de vista, como é o caso dos *Conexionistas*. Este grupo defende, naturalmente, os modelos conexionistas como mais adequados a este propósito, colocando-os como uma alternativa aos modelos simbólicos. Entretanto, a maneira como fazem isto se dá em graus variáveis.

Pinker & Prince (1988) traçam três diferentes visões conexionistas em relação a esta questão. A primeira, chamada por eles de “conexionismo implementacional,” consiste da idéia de que modelos conexionistas ofereceriam um meio de implementar os processos simbólicos já conhecidos, através de módulos com funções específicas construídos de acordo com as restrições conexionistas. Assim, ao invés de substituir os modelos atuais por algo radicalmente novo, os modelos conexionistas iriam complementá-los, lançando nova compreensão sobre as primitivas simbólicas disponíveis e seus custos, bem como disponibilizando primitivas até então inexistentes. Os algoritmos e toda a pesquisa já desenvolvidos até hoje poderiam, portanto, ser plenamente aproveitados ou aperfeiçoados.

Outra possibilidade é o chamado “conexionismo eliminacionista,” que defende a idéia de que modelos conexionistas reproduzindo aspectos da cognição não teriam correspondência com nenhum processo simbólico que conduzisse ao mesmo resultado. Todo o sistema seria explicado unicamente em torno das regularidades e superfície de energia em que o modelo estivesse envolvido, ou seja, em termos puramente conexionistas. Ainda seria possível falar de símbolos, estruturas e outras abstrações, mas estes não passariam de meras aproximações do verdadeiro processo subjacente.

Finalmente, haveria um terceiro grupo chamado “conexionismo revisionista do processamento de símbolos.” Este grupo consideraria o processo que se passa em um modelo conexionista como uma sequência de representações de entidades com propriedades semânticas formais tais como estruturas, valores verdade, relações de consistência e sistematicidade, de modo a serem previsíveis apenas em função destas propriedades. Um observador da implementação destas entidades em uma rede conexionista, entretando, encontraria estruturas e processos bem diferentes das normalmente postuladas por sistemas simbólicos. Os autores consideram que estas entidades, apesar da estrutura e primitivas de operação radicalmente diversas das encontradas no processamento simbólico, ainda merecem o nome de “processamento de símbolos” por exibirem suas qualidades semânticas, daí o nome deste grupo.

Algumas vezes encontra-se o argumento de que as abordagens simbólica e conexionista são simplesmente equivalentes. Afinal, pode-se implementar redes neurais a partir de símbolos e vice-versa. Este argumento porém não procede, como será melhor

compreendido quando colocarmos o problema em termos de níveis de análise, vistos na próxima seção.

3.3 Níveis de análise

Praticamente todos os ramos da atividade científica utilizam-se há muito da possibilidade de descrever os seus domínios em termos de *níveis de análise*. Trata-se de uma das maneiras de dividir problemas em partes independentes (embora relacionadas), aumentando as possibilidades de entendimento e análise de sistemas complexos.

A Ciência da Computação e a Inteligência Artificial não têm sido exceções. Linguagens de programação e sistemas operacionais são normalmente descritos nestes termos. Na Inteligência Artificial, níveis de análise têm sido utilizados proveitosamente não apenas na descrição de sistemas, mas também na análise do que um sistema cognitivo deve ser.

Um exemplo disto é o *nível do conhecimento* (*knowledge level*) proposto por Allen Newell (1982). Newell postula a existência deste nível em uma tentativa de separar, na IA Clássica, a noção do conhecimento dos símbolos que o representam. Outro exemplo são os argumentos de Fodor & Pylyshyn (1988) em defesa da visão Clássica da IA, que serão apresentados em maiores detalhes mais adiante.

Para basear os argumentos que serão apresentados, serão explicitados alguns conceitos sobre sistemas e seus níveis de análise nos parágrafos seguintes.

Um *sistema* é um processo através do tempo, um conjunto de objetos que mudam de estado constantemente, como por exemplo um ecossistema ou um computador digital. Um *elemento* do sistema será alguma entidade cujo estado é definido através de um conjunto de medidas sobre o sistema. Um elemento do ecossistema poderá ser a umidade do ar, um animal ou uma molécula. Um elemento do computador digital pode ser um bit de memória, um programa ou a corrente em determinado condutor elétrico. O estado de um elemento do sistema pode ser *explicado* a partir de outros elementos se houver relações fixas entre o estado do primeiro em um dado momento e os estados dos últimos nos momentos anteriores.

Os elementos de um sistema podem estar divididos em diversos *níveis de análise*, com cada um deles devendo estar presente em apenas um nível. Cada nível de análise deve exibir a propriedade de seu estado (o estado de seus elementos) ser explicado *apenas* em função de outros elementos do *mesmo* nível. Isto não significa, entretanto, que seu estado deva ser completamente explicado; há espaço para a aleatoriedade e o não-determinismo. Note que um nível é por sua vez um sistema, e portanto pode ser igualmente dividido em sub-níveis de análise.

Um nível de análise *baseia* outro se existe uma função que mapeia estados do primeiro nível em estados do segundo. *Baseia* é uma relação transitiva, assimétrica e reflexiva. Note porém que ela não é uma relação de ordem, pois dois níveis de análise podem

basear-se mutuamente, embora sejam muitos os casos em que uma ordem surge naturalmente, como no caso do ecossistema mais facilmente analisado no nível quântico, atômico, molecular, de organismos e assim por diante. Em outros casos uma ordem não se apresenta, como por exemplo na geometria analítica onde há o nível algébrico e o geométrico, sem uma ordem específica ou “natural.”

Uma fator de ordenação bastante comum em sistemas projetados é a da *implementação*. Considere-se que um nível de análise pode ser definido como *básico* ou não, quando se pode dizer que ele está *disponível em grau 0*. Um nível está *disponível em grau n* se existe um nível de grau $n - 1$ que o baseia. Um nível N *implementa* um nível M se N baseia M e se N tem um grau de disponibilidade menor do que M . Neste caso diz-se que N é de nível *mais baixo* que M (e naturalmente M é de nível *mais alto* do que N). Um exemplo evidente deste caso é o dos níveis computacionais em que o sistema físico está disponível em grau 0 e outros níveis são implementados a partir dele.

Embora nos casos mais tradicionais a relação de implementação especifica uma única ordem dos níveis de um sistema, ela pode na verdade especificar várias ordens. Se um nível básico A implementa B , B implementa C , C implementa D e C baseia B , então pode-se ter a ordem A, C, B, D (B implementa D , pois esta relação é transitiva). Na verdade, pode-se conceber sistemas com um único nível básico e todos os outros em uma ordem absolutamente arbitrária no que se refere à implementação. Porém serão poucas, senão apenas uma, as ordens de implementação desejáveis em um sistema.

Para que se possa falar da desejabilidade de uma determinada divisão de níveis e ordem de implementação, é preciso um critério. Este critério é dado em função do objetivo dos níveis de análise. Embora se possa observar e, em muitos casos, prever completamente o comportamento de um sistema apenas através dos elementos de seu nível mais básico, é necessário prever seu comportamento em termos de elementos que não pertencem a este nível. Por exemplo, quando se quer saber se um determinado algoritmo percorre todos os nós de uma determinada árvore, é preciso prever o comportamento do sistema em termos dos elementos de interesse, ou seja, nós e árvores. Por isso os termos deste nível de representação de dados têm de ser relacionados com os termos do nível mais básico, cujas causalidades se conhece. Desta maneira as causalidades envolvidas no nível superior podem ser conhecidas.

Entretanto o mapeamento entre dois níveis pode ser complexo demais, impondo uma decomposição do sistema em mais níveis intermediários até que os mapeamentos se tornem suficientemente simples. Em sistemas onde a ordem de implementação dos níveis apresenta opções, haverá normalmente ordens de implementação em que a simplicidade dos mapeamentos é muito maior do que em outras.

A simplicidade dos mapeamentos surge assim como um fator de ordenação ainda mais importante do que a da implementação, pois é a simplicidade dos mapeamentos entre os níveis que conduzem à escolha da ordem de implementação. A ordem pela simplicidade,

por sua vez, não é conduzida por qualquer ordem de implementação. Ela é conduzida pelo *problema* em si, pelas propriedades a que se quer chegar no nível mais alto.

Em sistemas de Inteligência Artificial, os níveis de análise são normalmente explícitos e considerados no projeto. Sempre existe um nível disponível em grau 0 (usualmente o nível simbólico disponibilizado por uma linguagem equivalente à Máquina de Turing) e uma sequência de níveis onde o primeiro é implementado pelo nível simbólico e os demais implementados pelo anterior. Segundo Fodor & Pylyshyn, os chamados *Representacionistas* defendem que o último nível desta sequência é o *nível cognitivo* (ou *do conhecimento*, na terminologia de Newell). Este é o nível cujos elementos são as entidades relacionadas ao raciocínio, tais como objetivos, lembranças, imagens, ações, percepções etc. Os *Reduccionistas*, por outro lado, negam a necessidade de tal nível, afirmando que um sistema inteligente pode ser definido e compreendido em termos completamente distintos (usualmente nos termos de um dos níveis mais baixos do que o cognitivo na abordagem representacionista).

Tanto Classicistas quanto Conexionistas são Representacionistas, pois identificam no último nível de seus modelos elementos diretamente relacionados aos elementos do nível cognitivo, em um mapeamento muito simples e direto. Portanto eles objetivam o nível cognitivo, com características e propriedades próprias conhecidas até certo ponto, e partem do mesmo nível básico, a Máquina de Turing (mesmo que um conexionista desenvolva modelos implementados em hardware, ele dificilmente vai negar que o mesmo modelo não possa ser simulado em um computador digital). A divergência se encontra *em quais níveis devem ser postulados, e em que ordem*. Conexionistas afirmam que o nível simbólico apenas implementará redes conexionistas, enquanto Classicistas defendem o ponto de vista de que, se for *necessário* um nível conexionista (no caso do cérebro constituir um), ele apenas estará implementando um nível simbólico.

Como se disse, são as propriedades do nível a que se quer chegar que determinam a simplicidade dos mapeamentos entre os níveis escolhidos. Já foram apresentados na seção 3.1 algumas das propriedades que se acredita serem inevitáveis no nível cognitivo. Veremos na seção seguinte razões que fazem alguns pesquisadores defenderem um nível ou outro logo abaixo do cognitivo.

3.4 Quem está mais próximo do nível cognitivo?

Tendo como instrumento os níveis de análise, é possível abordar melhor diversos aspectos da questão do alto nível nos modelos conexionistas. Um deles é a equivalência dos modelos simbólicos, modelos conexionistas e a Máquina de Turing. Algumas vezes isto é usado para basear a opinião de que a distinção entre eles é menor do que se alega (como por exemplo em Honavar & Uhr, 1995). Isto não se sustenta se considerarmos que uma rede conexionista não é apenas um conjunto de unidades simples com conexões mediadas por pesos. É necessário que também se apresente as propriedades dos modelos conexionistas, como a representação distribuída, a degradação suave, entre outros.

Implementar um sistema simbólico usando as unidades de modelos conexionistas não leva a um modelo que se possa chamar conexionista. A questão é, portanto, se os modelos conexionistas seriam equivalentes à Máquina de Turing enquanto continuassem apresentando as citadas propriedades. O reverso vale para os sistemas simbólicos, isto é, como eles podem apresentar as capacidades desejáveis dos modelos conexionistas sem perder as suas próprias.

Colocando a questão de outra forma, pode-se dizer que se houvesse um sistema implementando o nível cognitivo através de um modelo simbólico, ter-se-ia consequentemente um sistema fazendo o mesmo através de um modelo conexionista, bastando implementar os símbolos necessários a partir da rede. Mas neste caso um nível simbólico continuaria intermediando os níveis cognitivo e conexionista. O contrário também seria verdadeiro, pois qualquer modelo conexionista pode ser implementado por um nível simbólico. Isto não torna o modelo simbólico no que tange ao nível cognitivo, pois um nível conexionista continuaria existindo entre os níveis simbólico e cognitivo. A questão portanto é: *qual dos dois níveis é o mais próximo do nível cognitivo?* Se for o simbólico, de nada serve a equivalência das redes; será necessário estudar esquemas simbólicos para chegar ao nível cognitivo. Se for o conexionista, os símbolos apenas implementarão a rede de unidades, que terá de ser compreendida de qualquer maneira. Será apresentado a seguir as razões apontadas por um lado e por outro em defesa de sua perspectiva.

3.4.1 Argumentos a favor da maior proximidade do nível simbólico em relação ao nível cognitivo

Fodor & Pylyshyn (1988) fazem uma defesa do nível simbólico como a mais adequada base de implementação do nível cognitivo de duas formas: refutando o nível conexionista como apto a apresentar várias propriedades necessárias ao nível cognitivo e relacionando diversas propriedades desejáveis em um sistema de IA que são frequentemente associadas a modelos conexionistas, mas que podem ser realizadas também em sistemas simbólicos.

Depois de apresentar as qualidades do nível cognitivo já expostas na seção 3.1, os autores consideram que símbolos possuem estas características a partir de sua própria definição. Símbolos sempre podem ser combinados, e suas estruturas podem ser apreciadas sem que o conteúdo de seus nós sejam relevantes, possibilitando assim processos sensíveis a estrutura. Os constituintes das estruturas simbólicas seriam os correspondentes mais naturais para os constituintes das estruturas cognitivas.

Os modelos conexionistas, porém, não apresentariam estas características. Uma máquina Clássica que realiza a inferência de 'A&B' para 'A' o faz de acordo com a estrutura de 'A&B', que explicitamente atribui o papel de constituintes a 'A' e 'B.' É baseado nestes papéis e na estrutura de & que ela realiza a inferência, que será correta também quando houver 'D&E' ou '(P&Q)&R.' Neste último caso, inclusive, seria também realizada, pelo mesmo princípio, a inferência 'P&Q' para 'P' e 'Q.' Um modelo Conexionista pode fazer

o mesmo. Porém seria, segundo os autores, um modelo da forma expressa na figura 3-1. Este modelo realizaria corretamente a inferência de ' $(P \& Q) \& R$ ' para ' $P \& Q$,' ' R ,' ' P ' e ' Q ,' mas não o estaria fazendo *baseado na estrutura* de ' $\&$.' (Os rótulos dos nós poderiam fazer pensar que a estrutura está sendo levada em conta, mas eles não têm nenhuma influência no processo.) Isto violaria uma necessidade do nível cognitivo, isto é, que uma tal inferência *seja* baseada na estrutura.

figura 3-1: Uma rede conexionista típica que realiza inferências de ' $P \& Q \& R$ ' para ' $P \& Q$ ' e ' R ' e assim por diante não o está fazendo baseada na estrutura de ' $\&$ ' (que não está representada separadamente). O conhecimento que deriva ' P ' e ' Q ' de ' $P \& Q$ ' deveria ser o mesmo que deriva ' $P \& Q$ ' e ' R ' de ' $P \& Q \& R$,' mas estas duas inferências estão sendo realizadas por conjuntos completamente distintos de conexões, não mantendo assim o princípio da coerência inferencial.

O centro do problema, segundo os autores, está no fato de que as representações em modelos conexionistas serem sempre atômicas, não importa quão complexas elas sejam. Normalmente realizadas através de padrões sobre um conjunto de unidades, as representações conexionistas não estabelecem entre os componentes destes padrões qualquer relação estrutural. A única relação existente é a de causalidade. Não há outras relações que possam compor componentes em representações moleculares.

Eles acreditam que este fato tenha passado despercebido para muitos, principalmente porque os componentes dos padrões, as chamadas micro-características (seção 2.6), dariam a impressão de serem constituintes das representações no mesmo sentido que os constituintes das representações simbólicas o são de suas respectivas representações moleculares. Quando uma sala é representada em um modelo conexionista como a união de diversos sub-padrões representando mesas, janelas, sofás e portas, pode-se facilmente confundir uma tal representação como molecular, quando na verdade não o é, já que a única relação que une todos estes itens é a de causalidade.

Dirigindo-se ao outro lado da questão, Fodor & Pylyshyn procuram negar alguns dos argumentos contra modelos simbólicos que sustentam a preferência de muitos pelos modelos Conexionistas. Alguns deles são:

Dificuldade de modelos clássicos realizarem reconhecimento de padrões de alta capacidade e recuperar itens de memória pelo conteúdo. Humanos parecem recuperar com facilidade enormes quantidades de dados. O número de restrições sobre o conteúdo do item de memória a ser localizado parece apenas facilitar a busca. Isto seria muito complexo se feito serialmente como computadores convencionais o fazem, portanto o conhecimento das pessoas deve ser realizado de modo muito diferente do que o usual em modelos simbólicos.

Explicitude de regras. Em modelos simbólicos as regras do comportamento estão normalmente estabelecidas de modo explícito em uma representação similar à linguagem. Isto cria dificuldades para se lidar com processos altamente implícitos como guiar uma bicicleta ou lembrar-se do nome de alguém. Ainda nestes modelos, situações típicas e as respectivas exceções são tratadas de modo completamente diverso através de regras muito diferentes, embora alguns processos cognitivos apresentem um *continuum* entre os dois extremos.

Sensibilidade aguda dos modelos convencionais a danos e ruído. É largamente divulgado o fato de que danos cerebrais em extensão limitada não comprometem completamente o funcionamento mental das pessoas. Danos locais não eliminam memórias ou habilidades específicas, mas parecem diminuir o desempenho como um todo. Também é assumido que o *hardware* cerebral não possui grande precisão e que portanto uma considerável quantidade de ruído existe sem comprometer o funcionamento mental. Isto oferece um claro contraste em relação a modelos computacionais convencionais, onde as representações são altamente locais e a memória é uma sequência de registros.

O armazenamento em modelos convencionais é passivo. Em modelos simbólicos os itens da memória só têm alguma influência quando acessados explicitamente por um controlador central. Na cognição, pelo contrário, itens da memória são trazidos à baila sem um acesso explícito, reagindo à situação presente de acordo com sua definição, como, por exemplo, nas ocasiões em que detalhes vistos alguns dias antes e aparentemente sem importância são trazidos ao processo de resolução de algum problema atual.

Modelos tradicionais não tratam a aplicação de regras e conceitos em graus variáveis. Diz-se muitas vezes de modelos tradicionais que tratam regras e conceitos de uma maneira ‘tudo-ou-nada’, ou seja, muitas vezes mostram-se inaptos a aplicar regras a situações em que as condições para isto não são completamente atendidas, quando poderiam fazê-lo proveitosamente.

Falha em demonstrar degradação suave. Ao estarem incapacitados de realizar uma tarefa perfeitamente, as pessoas fazem em geral algo razoável. Modelos convencionais, por

outro lado, tendem a produzir um comportamento sem sentido quando confrontados com uma situação deteriorada.

Modelos computacionais simbólicos da cognição não levam em consideração fatos da neurologia. Modelos simbólicos raramente vêm acompanhados das formas como eles poderiam estar implementados no cérebro. São planejados de forma a resolver um problema sem nenhuma consideração às restrições neurológicas. Isto talvez leve ao desenvolvimento de modelos tão distintos do nível neurológico que não possam estar implementados de nenhuma maneira no cérebro e não possuam nada em comum com a cognição humana.

De acordo com Fodor & Pylyshyn, todos estes argumentos estão simplesmente errados. Eles os dividem em dois grupos básicos: aqueles que se sustentam em características que não são intrínsecas aos modelos Clássicos, podendo haver modelos deste tipo sem aquelas características, e aqueles que apontam problemas mais direcionados à implementação, estando portanto voltados ao nível errado, pois a disputa entre modelos Conexionistas e Clássicos diz respeito ao nível cognitivo, e não como o nível escolhido para implementá-lo deve ser implementado.

Algumas das respostas a argumentos do primeiro grupo são:

Explicitude de regras. Nada impede que um sistema Clássico possua uma grande quantidade de regras implícitas (na verdade, todos eles o fazem em alguma medida, através do código do programa que os implementa). Apenas as regras que devem ser manipuladas precisam ser explicitamente representadas (e o mesmo deve provavelmente aplicar-se a sistemas conexionistas).

Falha em demonstrar degradação suave. Uma boa parte de problemas com a degradação suave, segundo eles, é a de uma falta de conhecimento em geral; o sistema pode simplesmente não saber o que fazer quando uma série de métodos conhecidos falha. Modelos simbólicos futuros podem incorporar este conhecimento.

Modelos tradicionais não tratam a aplicação de regras em graus variáveis. Novamente trata-se de um tópico que pode ser tratado em uma abordagem simbólica. As condições de regras podem ser definidas de maneira a permitir sua detecção em graus variáveis. Nada impede em princípio que várias regras sejam ativadas de uma só vez em graus ou prioridades diferentes, com seus efeitos interagindo entre si ou sendo selecionados de acordo com a prioridade.

Os argumentos que se encaixam no segundo grupo são:

Dificuldade de modelos clássicos realizarem reconhecimento de padrões de alta capacidade e recuperar itens de memória pelo conteúdo. A alta capacidade exibida por um sistema estará, via de regra, relacionada à implementação. Limites de passos seriais para um determinado processo podem ser contornados por sistemas simbólicos

implementados de forma paralela (aqui, porém, Fodor & Pylyshyn parecem não levar em conta que a implementação em hardware mais rápido e em paralelo levam a melhorias apenas constantes, o que pode não ser suficiente se os algoritmos usados são de ordem maior que linear). A recuperação de memória por conteúdo é um tópico que diz respeito à representação escolhida para os símbolos. O problema pode ser tratado no próprio nível simbólico (escolhendo-se representações simbólicas apropriadas para certos conceitos) ou no nível que implemente o nível simbólico, fazendo-o de maneira a tornar automática a recuperação de qualquer estrutura através de seu conteúdo.

O armazenamento em modelos convencionais é passivo. Embora as representações no nível simbólico sejam passivas, o comportamento de um agente será definido não só em função destas representações mas também em função da máquina que as manipula. Neste sentido elas podem comportar-se de modo ativo de acordo com o processo cognitivo em questão. Se este comportamento deve ser levado em conta apenas no nível simbólico ou já no nível de implementação é uma questão em aberto a ser respondida empiricamente.

Sensibilidade aguda dos modelos convencionais a danos e ruído. Embora altamente localizadas no nível simbólico, as representações envolvidas em um modelo clássico podem estar distribuídas no nível que as implementa, tornando-as resistentes a danos localizados. Da mesma maneira, é responsabilidade dos dispositivos físicos levar em conta o ruído que possa haver para que ele não influencie o funcionamento do nível superior.

Modelos computacionais simbólicos da cognição não levam em consideração fatos da neurologia. Este é o argumento que mais claramente refere-se ao nível de implementação ao invés do nível cognitivo. Para um Classicista, a compatibilidade com o cérebro é uma necessidade apenas do nível que implementará o nível simbólico sobre o qual ele teoriza. O nível simbólico em si não precisa ser similar ao nível neurológico, como rochas e rios não são similares aos prótons que os “implementam.” A preocupação de chegar a modelos simbólicos não-implementáveis no nível neurológico não é um argumento a favor do Conexionismo, pois se os modelos conexionistas podem ser implementados neurologicamente, os simbólicos também podem (ainda que isto tenha que ocorrer via modelo conexionista).

Finalmente, Fodor & Pylyshyn concluem que se há um lugar para modelos conexionistas, este lugar é apenas como o de um nível de implementação para os modelos simbólicos, seja para conferir-lhes robustez, seja para estabelecer uma ligação entre modelos simbólicos e modelos neurológicos.

3.4.2 Argumentos a favor da maior proximidade do nível conexionista em relação ao nível cognitivo

A argumentação de Fodor & Pylyshyn em favor de sistemas simbólicos, portanto, divide-se em duas partes: a primeira procura mostrar que os modelos conexionistas são inadequados, declarando-os inaptos a realizar aspectos importantes da cognição: a

manipulação sistemática de e sensibilidade a estruturas, sua composicionalidade etc. A segunda parte diz que os sistemas simbólicos podem igualmente realizar os aspectos da cognição que os modelos conexionistas tratam adequadamente (ainda que com dificuldades, pode-se adicionar, posto que não são propriedades “naturais” aos sistemas simbólicos nem usuais nos sistemas conhecidos). Eles estariam assim aptos a realizar o que os modelos conexionistas fazem e, além deles, outros aspectos essenciais à cognição.

Curiosamente, pode-se traçar uma linha de argumentação simétrica em favor de modelos conexionistas. Da mesma forma que há partes da cognição a que modelos simbólicos prestam-se particularmente bem (vistas na seção 3.1), há partes da cognição a que modelos conexionistas adaptam-se de modo bastante adequado e direto. Estas partes são aquelas apontadas por eles na segunda parte de sua argumentação: necessidade de reconhecimento de padrões através de processamento paralelo e endereçamento por conteúdo, degradação suave, aplicação de regras em vários graus etc. Embora eles apontem possibilidades de modelos simbólicos realizarem as mesmas tarefas, isto não é simples. As entidades que apresentassem tais características não estariam diretamente relacionadas a símbolos, posto que apresentariam propriedades bastante diversas; o mais provável é que fossem compostas por eles. Modelos conexionistas implementam estas entidades muito mais diretamente.

A simetria estende-se à segunda parte de seu argumento, em que eles afirmam que modelos simbólicos podem realizar as partes de cognição que usualmente acredita-se não poderem. O mesmo se pode dizer dos modelos conexionistas. Nada impede em princípio que eles apresentem as características necessárias à cognição mais diretamente realizadas por símbolos, ou seja, sistematicidade, composicionalidade, coerência inferencial etc. Certamente isto não é realizado pelos modelos *típicos* de redes conexionistas, em que as representações são quase sempre atômicas de fato, da mesma maneira em que as críticas a modelos simbólicos apresentadas na segunda parte de sua argumentação aplicam-se apenas aos modelos *típicos* de modelos simbólicos.

De fato, caracterizamos em 2.6 os modelos conexionistas como apresentando uma interpretação do valor de ativação de suas unidades como a plausibilidade de hipóteses, que estariam relacionadas por conexões excitatórias ou inibitórias. Nos modelos conexionistas *típicos* estas hipóteses dizem respeito a um único objeto do problema. Mas nada impede que estas hipóteses sejam sobre quaisquer outros dados relevantes, incluindo-se aí dados sobre estruturas, sobre constituintes e sobre ligações de constituintes às respectivas estruturas. Neste caso, pode-se ter modelos conexionistas que atendam às exigências do nível cognitivo. A estrutura ‘P&Q’, por exemplo, pode estar codificada na rede de forma a manter separados os dados sobre ‘P’, ‘&’ e ‘Q’ e suas ligações, de forma a processá-los de forma sistemática. Se ‘P’ mostrar-se formada por ‘R&S’, por exemplo, a parte ‘&’ de ‘R&S’ deverá ser a mesma que era de ‘P&Q’, possibilitando a inferência de ‘R’ e ‘S’ do mesmo modo em que se inferiu ‘P’ e ‘Q.’

Note que ao realizar estes aspectos requeridos pelo nível cognitivo, os modelos conexionistas não precisam fazê-lo simplesmente como os modelos simbólicos. Todos estes elementos podem estar distribuídos, podem ser processados em paralelo, apresentando degradação suave etc. Mesmo nos problemas em que parte de suas estruturas sejam necessariamente simbólicas, ou seja, exibam as mesmas características que um modelo simbólico ofereceria, seria possível “emular” esta parte no modelo conexionista através do estabelecimento das mesmas relações que estariam presentes no modelo simbólico (*funcionando* como no modelo simbólico, mas com base no modelo conexionista), mas preservando as representações puramente conexionistas no restante das estruturas que não exigiriam isto. Se por exemplo um problema é sobre estruturas simbólicas, o modelo teria que aprender sobre estas estruturas e suas relações simbólicas, devendo ser capaz de simular seu comportamento. Entretanto, outros aspectos do problema, como os métodos para tomar decisões em direção a um determinado objetivo, poderiam continuar conexionistas, estando bem integrados à parte “simbólica” do problema. Este seria, na verdade, um caso típico na abordagem da cognição superior, pois são muitos os problemas que exigem estruturas equivalentes às simbólicas. Mas estes mesmos problemas possuem também componentes melhor abordadas por características conexionistas.

A simetria com os modelos Clássicos ainda se apresenta aqui. Da mesma forma que modelos conexionistas poderiam “emular” estruturas similares às simbólicas quando necessário, modelos clássicos poderiam “emular” aspectos conexionistas quando necessário. Nos dois casos, isto daria origem a um nível mais poderoso que contivesse as características positivas das duas abordagens, uma generalização de ambos os modelos. Estas “emulações” de um modelo por outro são porém dispendiosas e pouco compreendidas. Saber qual das duas abordagens permite este salto com maior facilidade dependerá apenas dos avanços futuros de cada área. O final da disputa pode estar ainda longe de uma conclusão.

A próxima seção aborda o lado conexionista deste esforço, apresentando alguns dos problemas envolvidos. Também serão apresentados os trabalhos que procuraram contornar algumas destas dificuldades.

3.5 Problemas e abordagens conexionistas às necessidades do nível cognitivo

Barnden (1995) aponta como desfavoráveis ao tratamento de alto nível as seguintes restrições sobre a arquitetura dos modelos conexionistas:

1. Não há criação e destruição rápida e dinâmica de nós ou conexões na rede. Portanto, informações temporárias não podem ser codificadas em mudanças topológicas da rede;
2. Conexões nas redes conexionistas não podem ser diferenciadas por rótulos, e por isso relações entre conceitos devem ser codificadas através de ativações de unidades, pesos

etc. Segundo Barnden, isto aumenta significativamente a dificuldade em tratar a informação;

3. Os valores das unidades nas redes conexionistas não têm resolução suficiente para conter informação complexa, representando normalmente apenas níveis de confiança;
4. Apontadores e programas não podem ser armazenados e utilizados, pelo menos de uma forma direta, dificultando o tratamento de tarefas que naturalmente os utilizam.

Estas restrições têm como base a manutenção das características positivas das redes conexionistas, como por exemplo o treinamento baseado em regressão linear. Alguns pesquisadores também consideram importante sua manutenção com vistas à plausibilidade biológica (neurônios não são criados ou destruídos rapidamente, seus disparos não têm alta resolução etc). Devido a estas restrições, os pesquisadores têm procurado resolver os principais problemas em termos das ativações das unidades.

Estas restrições dos modelos conexionistas dificultam a reprodução de fatores importantes no tratamento de problemas de alto nível, as chamadas ferramentas de controle de complexidade apresentadas na seção 3.1: o tratamento de estruturas, processos temporais e controle e manutenção de múltiplos objetos da mesma classe. As próximas sub-seções apresentam as dificuldades envolvidas quando se procura tratar estas ferramentas com modelos conexionistas. No caso de tratamento de estruturas, este problema é normalmente referenciado na literatura conexionista em termos de um problema um pouco mais geral, o de *ligação de variáveis*. Por isso ele será referenciado a partir deste ponto desta maneira.

A sub-seção seguinte apresenta uma solução parcial comum para estes problemas: modelos conexionistas estruturados.

3.5.1 Múltiplos objetos de uma mesma classe

Uma dificuldade importante na modelagem conexionista de problemas em alto nível vem do fato de muitos deles envolverem diversos objetos da mesma classe, ou diversas instâncias da mesma relação. Em modelos típicos cada entidade permanece fixa em um certo conjunto de unidades cujas conexões representam o conhecimento envolvido em objetos de sua classe. Na representação de um cômodo de uma casa, por exemplo, os diversos elementos que o compõe possuem posições fixas que são completadas gradualmente de acordo com as demais informações, através do intermédio das conexões entre estas posições, às quais se chama comumente *slots* (Rumelhart *et al.*, 1986b). Neste tipo de tratamento, múltiplos objetos de uma mesma classe teriam de ser representados cada um em seu slot. Entretanto, além disto não fornecer a possibilidade de representar um número arbitrário de objetos, exigiria que as propriedades da classe com várias instâncias estivesse replicada em cada slot. Seria necessário o treinamento separado de cada um dos slots ou a cópia dos pesos das conexões de um slot para os demais. O primeiro caso resultaria em um treinamento pouco natural e ineficiente, pois em geral

aprende-se as propriedades de uma determinada classe apenas uma vez. O segundo violaria a restrição das conexões da rede serem formadas apenas através de ajustes, além de exigir que se conheça o conjunto de unidades sobre o qual a rede está representando determinada classe, o que não é possível muitas vezes.

Hinton (1990) trata esta questão e aponta algumas medidas necessárias em sua abordagem. Ele reconhece a importância de representar diversos objetos e suas relações no tratamento de muitos problemas cognitivos, mas considera ingênua a idéia de representá-los todos ao mesmo tempo sobre vários conjuntos de unidades. Argumenta que na maioria dos problemas da vida real existe uma enorme quantidade de informação e conhecimento a ser compartilhado entre vários conceitos. Isto significa que se este conhecimento for realizado em um modelo conexionista pelos pesos das conexões entre certos conjuntos de unidades, então os conceitos, ao menos enquanto forem processados à luz deste conhecimento, deverão estar representados sobre este conjunto de unidades (um processamento similar ao processamento por referência não seria possível, já que não há algo como apontadores ou endereços em um modelo conexionista). Como existem diversos conceitos compartilhando este conhecimento, será necessário haver um compartilhamento no tempo (*timesharing*) destes conjuntos de unidades pelos diferentes conceitos.

Este mecanismo seria válido mesmo quando duas ou mais entidades do mesmo tipo tenham que ser processadas sob o mesmo conhecimento em um problema. Neste caso, todas elas seriam mapeadas no conjunto de unidades correspondente a este conhecimento para o tratamento necessário, uma de cada vez. Uma integração final poderia ocorrer, por exemplo, através dos resultados temporários armazenados em unidades próprias para isto.

Um exemplo de um processo como este seria o de reconhecimento de palavras. Em um modelo conexionista para o reconhecimento de palavras de McClelland & Rumelhart, o conhecimento sobre a imagem correspondente a cada letra era replicado para cada posição da palavra em que ela pudesse aparecer. Isto entretanto poderia ser feito com apenas uma instância das conexões necessárias para reconhecer cada letra. Neste caso, cada posição teria de ser processada separadamente e em sequência para que se descobrisse de qual letra se tratava. O resultado seria armazenado em unidades específicas que serviriam para o reconhecimento de palavras inteiras.

Pode-se imaginar que o cenário anterior, com várias entidades da mesma classe mapeadas sucessivamente sobre o mesmo conjunto de unidades, não impede que cópias inteiras destas mesmas entidades estejam armazenadas em outras partes da rede, sendo copiadas quando necessário. Mesmo se constituírem partes de outras entidades de outro tipo, pode-se imaginar que elas estariam inteiramente representadas, sob a pena talvez leve de esta entidade de outro tipo que as contém ter uma representação maior e mais complexa. Entretanto, esta concepção falharia em pelo menos dois casos. O primeiro é quando o mesmo objeto faz parte de diferentes entidades de outros tipos. Ter duas cópias seria custoso já que alterações em uma delas teriam de se refletir na outra automaticamente. O

segundo é quando a entidade que envolve um objeto é um objeto da mesma classe que a sua. Isto implicaria que a representação de uma não poderia ser maior do que a outra, mas deveria contê-la e aos demais dados que a compusessem, o que é impossível.

Um exemplo mais concreto de hierarquias de objetos do mesmo tipo envolvidas em um problema é o de um problema envolvendo uma árvore genealógica. A visão mais simples poderia supor que todas as pessoas deveriam estar representadas em todo o seu detalhe ao mesmo tempo. Como se viu acima, porém, isto não é viável, pois deve haver apenas um conjunto de unidades codificando o conhecimento sobre pessoas no nível total de detalhe. Entretanto, o sistema deve estar apto a recuperar esta descrição detalhada de qualquer uma das pessoas envolvidas. Pode-se por exemplo procurar um filho com certa idade da pessoa “corrente” (aquela que ocupa o conjunto unidades que codifica pessoas em todo o detalhe) para uma consulta. Embora este filho não esteja representado totalmente em parte alguma no momento, deve haver informação suficiente para recuperá-lo. Esta informação, que serve como uma chave de busca para este filho, deverá estar naturalmente associada à pessoa corrente (pai ou mãe), fazendo parte de sua própria representação no nível mais baixo de detalhe. Portanto, a informação capaz de recuperar a descrição completa de uma pessoa deve poder ser menor que a representação completa daquele tipo, e deve poder estar no mesmo conjunto de unidades sobre a qual será restituída a entidade completa.

Note que não se estabelece nenhuma restrição a esta informação que serve como chave, portanto ela pode ser usada de forma bem mais proveitosa do que simplesmente como uma chave arbitrária. Hinton propõe a consideração deste tópico através da diferenciação entre “inferência intuitiva” e “inferência racional.” A primeira é o tipo de cálculo realizado em redes conexionistas através de seu equilíbrio em um determinado estado, enquanto a segunda é o tipo de cálculo serial realizado na computação convencional. Segundo ele, a grande inovação da abordagem conexionista foi trazer ao repertório da Inteligência Artificial o tipo de inferência intuitiva, pois a IA simbólica abordava todos os problemas através da inferência racional.

A inferência intuitiva realiza-se como um mapeamento entre uma representação a outra por intermédio das regularidades de um determinado domínio. Como este mapeamento é relativamente direto, é necessário que todas as informações envolvidas na inferência estejam presentes simultaneamente, sem a necessidade de trocas e cópias, comuns na inferência racional. A troca de entidades sobre um conjunto de unidades é um passo típico de inferência racional, e realmente necessário em muitos casos. Entretanto, não é necessário que *todas* as inferências envolvendo objetos do mesmo tipo relacionados entre si sejam racionais. Para ser uma inferência intuitiva, basta que as informações relevantes estejam disponíveis. É neste ponto que a liberdade do conteúdo da chave para outros objetos entra na questão. Ela pode ser usada para conter, além do necessário para uma recuperação unívoca do objeto correspondente, as informações necessárias para as inferências mais comuns ou que devam ser mais rápidas. Neste caso, mais do que ser uma chave de busca, esta informação constitui uma *representação reduzida* do objeto.

Contendo informações significativas dos objetos, as representações reduzidas tornariam possível que uma grande parte das inferências que antes seriam do tipo racional possam ser intuitivas. Assim, se no exemplo da árvore genealógica a representação reduzida de cada pessoa inclui sua idade e seu nome, e se a consulta ao filho de certa idade da pessoa corrente é apenas para saber seu nome, esta consulta poderá ser realizada através de uma inferência intuitiva.

Este problema relaciona-se com o fato de que normalmente é útil referenciar objetos e algumas de suas propriedades relevantes em um processo ao invés de usar suas representações inteiras, no mais completo nível de detalhe. É o problema a que se chama comumente *acesso distal* (Touretzky, 1990). Ao se raciocinar sobre uma viagem com um determinado carro, por exemplo, usa-se algumas de suas propriedades como velocidade, autonomia etc, mas não se chega a detalhes do funcionamento de seu motor. Estas propriedades desnecessárias em um momento, entretanto, podem estar disponíveis se for preciso considerar sobre elas.

Hinton não estipula um método para determinar qual representação reduzida deve ser usada, mas outros o fizeram. Uma das mais conhecidas é a proposta por Pollack (1990), Recursive Auto-Associative Memory (RAAM), formada pelo próprio método de aprendizado de uma rede conexionista. Treinando uma rede com backpropagation para reproduzir o padrão de entrada na camada de saída através de uma camada intermediária com a metade do comprimento, Pollack forçava o desenvolvimento de uma representação reduzida adequada nesta camada intermediária. Tendo esta representação de tamanho reduzido, é possível codificar estruturas recursivas de comprimento arbitrário em um vetor de largura fixa, usando as conexões da camada de entrada para a intermediária como codificadoras e as da camada intermediária para a de saída como decodificadoras. Para isto basta codificar a estrutura de acordo com alguma representação e, dentro desta representação, usar a representação reduzida para estruturas aninhadas do mesmo tipo. Uma pilha, por exemplo, pode ser obtida a partir de átomos de tamanho n da seguinte maneira: uma pilha vazia é representada por um átomo especial *nil*. Pode-se empilhar um novo átomo em qualquer pilha concatenando-se o padrão correspondente ao átomo com o padrão correspondente à pilha e usando-se a rede para, a partir do padrão de comprimento $2n$ obtido, calcular qual o padrão de tamanho n correspondente à nova pilha. A mesma rede pode ser usada para decompor o padrão de comprimento n de uma pilha nos padrões de seu átomo do topo e no padrão da pilha sem este topo.

A RAAM de Pollack é ainda um pouco mais do que apenas um meio de comprimir dados. Em seu trabalho ele mostra como a codificação desenvolvida a partir de um conjunto de sentenças de uma linguagem L definida por uma gramática permite que outras sentenças de L possam ser codificadas com mais facilidade do que sequências arbitrárias, mostrando portanto que a codificação foi sensível a regularidades desta linguagem. Também mostra que as representações podem ser usadas diretamente em procedimentos que usem a separação da estrutura e seus constituintes. Uma rede foi

treinada por exemplo para mapear as RAAMs de sentenças da forma “X ama Y” nas RAAMs de “Y ama X,” exibindo certa generalização.

Outro tipo de representação que pode ser usada como a RAAM de Pollack é a Holographic Reduced Representation (HRR) de Plate (1995). A HRR também se ocupa do problema de representar estruturas de tamanho variado em vetores de largura fixa, mas o faz de maneira mais consistente em relação à similaridade de estruturas e proporciona um tratamento para o problema da ligação de papéis. Tanto o problema de ligação de papéis quanto as HRRs serão vistas com maior detalhes na próxima sub-seção.

3.5.2 Ligação de variáveis

Como foi dito anteriormente, nada impede em princípio que informações sobre estruturas e a forma como elas se compõem estejam representadas nas unidades de uma rede conexionista. Isto já é feito implicitamente em modelos típicos quando as relações de adjacência na representação escolhida para a entrada ou saída do problema possuem interpretações estruturais. Hinton (1990) vai um pouco além disto quando mostra como um conjunto extra de unidades pode representar a relação familiar entre duas pessoas representadas em certos slots (conjuntos determinados de unidades da rede). Ao instanciar-se uma das pessoas e a relação familiar, a memória de uma árvore familiar inteira codificada nas conexões recupera a pessoa correta. Ao instanciar as duas pessoas, as mesmas conexões produzem a representação da relação entre elas.

Pode-se imaginar como estruturas complexas, formadas por vários constituintes relacionados de diferentes maneiras, poderiam ser representadas em redes conexionistas. Cada um dos constituintes seria representado em determinado slot. Cada par de slots, por sua vez, poderia corresponder a um outro slot que representasse a relação entre eles. Esta é porém uma concepção ingênua que deixa de abordar diversos tópicos. O mais óbvio é que se torna extremamente dispendioso determinar um slot para cada par de conceitos. É típica a situação em que a maioria dos pares não está relacionada de forma relevante e por isso dispense qualquer representação a este respeito. Um par de entidades pode estar relacionada de múltiplas maneiras, necessitando de mais de uma representação de relações entre elas. Novas entidades são incorporadas durante a realização de tarefas, e dois ou mais objetos relacionados podem formar algo que também é um objeto. Estes novos objetos devem poder estar relacionados de diversas maneiras aos objetos anteriores ou entre si, talvez formando outros ainda mais complexos. Naturalmente o esquema fixo proposto não proveria slots para estes novos objetos.

Este é, na verdade, um dos mais importantes problemas para o tratamento do alto nível em redes conexionistas: o da ligação de variáveis (*variable binding*) (Bienenstock, 1995). Sua forma mais simples já foi discutida acima como o problema de estabelecer ligações entre valores representados simultaneamente na rede. Se representamos ao mesmo tempo os conceitos círculo, quadrado, azul e verde ao mesmo tempo em uma rede, isto significa

que tem-se um quadrado azul e um círculo verde ou que se tem um quadrado verde e um círculo azul? É necessário representar de algum modo as ligações entre estes conceitos.

O problema da ligação de variáveis aparece em outras diferentes formas mais complexas. Pode-se tê-lo em relação a papéis (*roles*) e seus respectivos valores (*fillers*) na representação de uma estrutura. Por exemplo, duas atividades de uma mesma pessoa estão representadas na rede e deve-se estabelecer a ligação de cada uma delas com os papéis *hobby* e *profissão* daquela pessoa, ou ainda em relação a uma determinada pessoa há diversos papéis preenchidos por outras: cônjuge, mãe, patrão etc. Pode haver até mesmo um único item para mais de um papel diferente. Relacionar cada valor com o respectivo papel é o problema da *ligação de papéis*.

Talvez a forma mais complexa do problema da ligação de variáveis surja quando um determinado conceito envolvendo algumas variáveis é aprendido. É necessário que este conhecimento seja usado para *qualquer* conjunto de variáveis. Por exemplo, se a rede infere que quando “X ama Y, Z ama Y e X, Y, Z são pessoas distintas então X tem ciúmes de Z,” é necessário que a rede esteja apta a aplicar este conhecimento para José no lugar de X, Maria no lugar de Y e João no lugar de Z, mas também apta a aplicá-lo para Joaquim no lugar de X, Mariana no lugar de Y e João no lugar de Z. Isto é simples de realizar se houver um slot para cada variável que é preenchido de acordo com o caso. Mas é importante que a rede seja capaz de realizar isto dinamicamente, no decorrer da solução de um mesmo problema. Por exemplo, se outra parte do conhecimento afirma que “se X tem ciúme de Z e Z têm ciúme de X, então há um duelo entre X e Z,” uma rede que precise prever a possibilidade de um duelo entre Joaquim e José baseada apenas no conhecimento dos indivíduos e as relações de amor entre eles deverá, durante o mesmo processo de resolução, aplicar o conhecimento sobre ciúmes para Joaquim e José alternadamente nos papéis de X e Z e procurar um indivíduo que possa ocupar o lugar de Y. Durante cada uma destas aplicações devem ser usadas as propriedades da variável correta.

Embora seja possível implementar um procedimento assim em uma rede connexionista (de um modo extremo, planejando um procedimento simbólico e implementando-o em forma de rede connexionista), é difícil fazê-lo através de treinamento por exemplos. Os exemplos com os slots das variáveis X, Y e Z e um slot indicando se X tem ciúme de Z preenchidos corretamente podem ser facilmente aprendidos por uma rede connexionista; este conhecimento estará expresso das conexões entre as unidades destes slots. A rede também pode receber exemplos dos padrões representando as pessoas envolvidas, recuperando qualquer uma delas a partir de algumas características, em um slot próprio para isto. Assim, o conhecimento sobre pessoas estará expresso nas conexões das unidades deste slots. Entretanto, o slot reservado à representação de pessoas não pode ser o mesmo usado para as variáveis X, Y e Z. No primeiro caso, há apenas um slot contendo o conhecimento sobre indivíduos e representando uma pessoa de cada vez. No segundo caso, há três slots contendo o conhecimento sobre o ciúme, preenchidos com pessoas diferentes e que devem ser considerados ao mesmo tempo. É necessário, portanto, algum

mecanismo de cópia ou de referência entre os slots de um tipo e de outro, de acordo com as necessidades do processamento. Entretanto, isto parece exigir um mecanismo de controle complexo de difícil treinamento através apenas de exemplos.

O problema da ligação de variáveis pode ser tratado de várias maneiras diferentes. Uma delas é através da escolha da representação adequada. Nesta vertente uma maneira é dedicar certa partes da representação à codificação de ligações. Isto acaba se tornando muito semelhante a ter unidades dedicadas de ligação, como no caso dos slots próprios para codificar relações no modelo “ingênuo” apresentado acima. Outro dos métodos mais utilizados é o de armazenar determinado valor em uma posição que caracterize a ligação envolvida. Por exemplo, o padrão representando uma atividade pode estar armazenado em uma posição dedicada ao hobby ou à profissão de uma pessoa, dependendo do caso.

Smolensky (1990) formaliza estas duas maneiras de ligar variáveis sob o conceito de *produto tensor*, um esquema geral para representar estruturas decompostas em duplas papel/valor. Basta considerar que o papel tem uma certa representação e o valor outra, e que a representação completa é formada com a inserção de um padrão determinado por seu produto externo (o mesmo usado para realizar o aprendizado Hebbiano na rede BSB, visto na seção 2.4.1). Ele aponta que a maioria das aplicações do produto tensor é do tipo local (quando as representações dos papéis e valores são locais) ou semi-local (quando apenas uma das duas o é), mas nada impede que ele seja usado de forma completamente distribuída (ver figura 3-2), beneficiando-se das qualidades destas representações. Os exemplos mais comuns também têm sido em apenas duas dimensões, quando é possível uma generalização para três ou mais dimensões, simplesmente usando o produto tensor em outros produtos tensores.

figura 3-2: a representação das ligações papel-valor pode ser obtida pelo produto tensor entre as representações de papéis e valores. Na figura, (a) é um produto tensor semi-local, pois a representação do papel é local enquanto o do valor é distribuído, enquanto (b) é totalmente distribuído, pois as representações de papel e valor são distribuídas.

Dispondo das representações do produto tensor de pares papel/valor de uma estrutura, pode-se utilizá-la em uma rede pelo simples aprendizado destas representações, obtendo os valores a partir dos papéis ou vice-versa. A recuperação de ligações apresentará as mesmas propriedades de recuperações comuns nas redes connexionistas, como por exemplo a degradação suave.

Uma das mais recentes representações com os mesmos objetivos que o produto tensor de Smolensky é a Holographic Reduced Representation (HRR), de Plate (1995). A HRR faz praticamente o mesmo que o produto tensor, com a vantagem de também poder realizar o papel de uma representação reduzida. Trata-se de uma técnica de codificação de estruturas através do produto de convolução circular entre representações de papéis e valores. A diferença é que representações de papel e valor de dimensão n produzem um HRR de dimensão também n , ao invés de n^2 como era o caso do produto tensor. A partir de vetores randômicos (com componentes reais de distribuição normal de média 0 e variância $1/n$, onde n é sua dimensão) representando papéis e valores atômicos, forma-se estruturas somando-se os produtos de convolução entre papéis e valores. As estruturas, codificadas desta maneira em vetores de dimensão também n , podem ser usadas como constituintes de outras estruturas, possibilitando estruturas recursivas. As estruturas podem ser decodificadas através de inversos (em relação ao produto de convolução) dos vetores dos papéis. Convolucionados com a estrutura, estes inversos produzem uma versão com ruído de média 0 do constituinte correspondente. Se necessário, este constituinte pode ser limpo (através de uma memória associativa) e por sua vez decodificado (se for também uma estrutura codificada).

Plate aponta algumas propriedades desejáveis das HRRs. Além de fornecer uma maneira de codificar estruturas de complexidade variável em vetores de largura fixa, elas permitem que estruturas com objetos e relações similares tenham sua similaridade computada simplesmente através do produto escalar. Embora estruturas com poucos objetos em comum exibam menor sensibilidade à similaridade estrutural, uma técnica chamada *contextualização*, que envolve a mudança dos vetores representando elementos atômicos (normalmente arbitrários) para que contenham informação (por exemplo sobre os papéis que ocupam mais frequentemente), pode contornar este problema em vários casos.

Ele compara ainda as HRRs favoravelmente em relação ao produto tensor de Smolensky e RAAMs de Pollack. As RAAMs requerem um longo tempo de treinamento, e estruturas semelhantes muitas vezes não têm representações semelhantes. Além disso, após o treinamento as RAAMs ficam restritas a um repertório limitado de elementos atômicos. O produto tensor, por sua vez, embora funcionalmente muito semelhante a HRRs, tem a dimensionalidade de sua representações aumentando exponencialmente quando há estruturas aninhadas. Isto limita severamente sua utilização prática em muitos problemas.

Barnden (1995a), através de seu sistema Conposit, utiliza duas maneiras distintas de ligar variáveis, o *RPE* (*Relative-Position Encoding*, ou Codificação por Posição Relativa) e o *PSA* (*Pattern-Similarity Association*, ou Associação por Similaridade de Padrões), ilustradas da figura 3-3. No primeiro caso, RPE, estruturas são armazenadas em espaço de registros (compostos por unidades conexionistas) com determinada relação de adjacência, e estruturas relacionadas são dispostas em registros vizinhos. No PSA, estruturas são relacionadas de acordo com a similaridade de um certo sub-padrão que contém. Embora sua utilização mais óbvia seja aceitar apenas a igualdade dos sub-padrões de duas estruturas para considerá-las relacionadas, esta técnica permite a interessante possibilidade de haver relacionamentos em diferentes graus, de acordo com a maior ou menor similaridade dos sub-padrões envolvidos.

figura 3-3: Dois modos de realizar a ligação de variáveis são o RPE (Relative-Position Encoding) (a) e o PSA (Pattern-Similarity Association) (b). O primeiro usa a posição relativa de certos registros em uma matriz de unidades para associá-los. As figuras geométricas representam os papéis dos registros em suas respectivas estruturas. O segundo usa a similaridade (ou identidade) de um sub-padrão dos registros para associá-los. Ambos, especialmente o PSA, requerem controle externo para serem efetivamente utilizados.

Tanto o RPE quanto o PSA, entretanto, exigem uma manipulação procedural mais complexa do que a usual. Suas propriedades de ligação de variáveis não ocorrem apenas pela propagação de ativação normal nas redes conexionistas, mas através de manipulações externas da rede. No caso do Conposit, por exemplo, há regras que detectam variáveis relacionadas e ações que têm que considerar explicitamente as relações de adjacência e identidade dos sub-padrões do PSA. Ou seja, estas técnicas *representam* a ligação das variáveis, deixando seu *processamento* a cargo de processos simbólicos.

Shastri (1995) propõe a realização da ligação de variáveis através de *sincronia temporal*. Neste caso, unidades que representem parâmetros de determinados procedimentos ou predicados representados em uma rede conexionista oscilam em fase com unidades que representam as entidades correspondentes a estes parâmetros em um dado momento. Assim, as mesmas unidades paramétricas podem ser usadas para realizar o procedimento ou predicado para diferentes entidades, bastando para isto fazê-las oscilar em fase com as que forem mais convenientes. Seu sistema SHRUTI utiliza a sincronia temporal e é também capaz de criar um número limitado de diferentes instâncias de cada predicado.

Para atingir seus objetivos, entretanto, o sistema SHRUTI deve ser estruturado, mantendo clara separação entre as unidades usadas para predicados e as usadas para entidades. A utilização desta estrutura exige por sua vez controle externo, do tipo convencional. De fato, a necessidade de controle externo parece tanto maior quanto mais complexas são as tarefas desempenhadas por uma rede. A questão do controle será apresentada na próxima sub-seção.

3.5.3 Controle

Como já foi dito acima, as tarefas cognitivas parecem poder ser analisadas em termos de inferências intuitivas e inferências racionais. As primeiras exigem que todos os dados necessários à computação efetuada estejam disponíveis e representados de maneira que suas relações possam ser expressas de forma muito simples. As segundas envolvem dados codificados de maneira mais complexa, exigindo uma manipulação temporal das informações para serem realizadas. As redes conexionistas são especialmente apropriadas para o primeiro tipo de inferência, mas pouco se fez para que possam realizar também inferências racionais.

Problemas cognitivos raramente são compostos por apenas um tipo de inferência mas, ao contrário, parecem poder ser divididos em passos, ou componentes, melhor tratados por inferências intuitivas ou racionais isoladamente. Um destes componentes usualmente identificado com inferências intuitivas é a representação e recuperação de dados. A manipulação temporal destes dados, por outro lado, é outra componente que, por definição, está ligada à inferência racional.

O campo dos modelos conexionistas sempre exibiu sucessos em relação a estruturas de dados que não exigissem inferências racionais para serem manipuladas. Durante a última década muita pesquisa tem se direcionado para as estruturas que as exigem em algum grau, procurando tratá-las em *alguns* aspectos com inferências intuitivas (somente alguns aspectos, pois trata-se justamente de estruturas que só podem ser completamente manipuladas com inferências racionais). As RAAMs de Pollack (1990) são um exemplo típico. Embora ele tenha procurado tratar a codificação das estruturas em representações reduzidas de forma conexionista, assim como algumas inferências envolvendo estas representações, a manipulação dos diversos componentes de uma estrutura deve ser feita por um executor externo no caso de inferências e composições mais complexas.

Outro exemplo é o sistema BoltzCONS (Touretzky, 1990), que procura representar listas como as da linguagem LISP em um modelo conexionista. Na verdade, o que o sistema faz é considerar a lista como um grafo e representar seus nós na rede em triplas de nós adjacentes naquele grafo. O fato desta representação ser conexionista produz algumas propriedades interessantes, como a busca de qualquer um dos nós em tempo constante, buscas baseadas em similaridade ao invés de simples identidade, e passeios bidirecionais por grafos onde isto só seria possível em uma direção. Entretanto, somente o armazenamento e recuperação dos nós é conexionista neste modelo; o processamento da

lista deve ser feito de modo sequencial por um controlador externo. A suposição que parece estar por traz disto é de que o principal problema é tratar tudo o que pode ser tratado de forma conexionista nestas estruturas, deixando a parte necessariamente racional como um outro tópico de pesquisa (modelos conexionistas executando também esta parte) ou a ser tratado por estruturas simbólicas definitivamente (quando o pesquisador acredita que esta é a alternativa mais adequada). Entretanto, soluções que incluem a suposição de um controle externo em separado (seja ele conexionista ou não) tomam certas características indesejáveis de modelos simbólicos, como por exemplo a passividade das representações, em contraposição com alternativas mais atraentes em que as representações não fossem puramente passivas, mas “reagissem” ao ambiente, em paralelo e de uma maneira independente e descentralizada.

Há muito poucos modelos que abordam a questão de um controle puramente conexionista. A principal razão parece ser de que controle envolve uma espécie de conhecimento distinto do conhecimento do domínio. Para realizar um determinado processo sequencial sobre um conjunto de entidades de um certo domínio, é preciso o conhecimento sobre outras entidades, tais como operadores, resultados, objetivos, e saber como aplicá-los. Este é de fato um componente a mais no conhecimento, tanto que é comum seu compartilhamento entre domínios diferentes: diversas estratégias usadas em um jogo de xadrez, por exemplo, podem ser empregadas na demonstração de um teorema. Este tipo de informação, porém, raramente está presente de alguma forma em lotes de treinamento provenientes de um determinado domínio. É preciso que ele seja adquirido por outras vias.

A utilização de modelos conexionistas estruturados é a mais utilizada destas vias para inserir no sistema, ainda que implicitamente, conhecimento sobre o controle necessário na resolução de problemas. Mas ela se presta ainda à solução de outros problemas, como por exemplo a ligação de variáveis. Modelos conexionistas estruturados são discutidos na próxima sub-seção.

3.5.4 Modelos conexionistas estruturados

Uma maneira de contornar dificuldades do conexionismo como ligação de variáveis ou controle são os *modelos conexionistas estruturados* (Shastri, 1995). Estes modelos compõe-se de diferentes módulos conectados entre si. Cada um deles pode ser composto de maneira completamente diferente dos demais. É possível até mesmo que alguns deles não sejam conexionistas, ou que sejam realizados por modelos conexionistas muito diferentes dos demais. Nestes modelos, a estrutura traduz de alguma maneira as necessidades do problema. O modelo SHRUTRI, de Shastri, é um exemplo de modelo conexionista estruturado, pois possui módulos explicitamente separados para entidades e predicados. Quando necessário, ele cria diferentes instâncias de um único predicado, para que sejam usadas simultaneamente com diferentes variáveis. Ou seja, o que seria normalmente realizado através de um procedimento temporal, pela aplicação do mesmo predicado para cada conjunto de variáveis um após o outro, é realizado como uma

inferência intuitiva, criando módulos que disponibilizem toda a informação necessária simultaneamente.

O sistema DISCERN (Miikkulainen, 1995) também faz amplo uso de módulos para realizar sua tarefa. Trata-se de uma implementação conexionista da compreensão de histórias baseadas em scripts (Schank, 1977). Tendo aprendido uma série de histórias-modelo (os scripts), o sistema os aplica no reconhecimento de novas histórias que venha a receber. Assim, ele pode receber algumas histórias sobre jantares em restaurantes e desenvolver um script típico destas ocasiões. Depois pode receber a informação sobre uma pessoa que vá a um certo restaurante, e inferir como isto deverá ter sido. Ainda que as instâncias dos scripts conttenham um número reduzido de informações em relação aos scripts completos, o sistema as reconhece e completa as informações ausentes. A tarefa exige controle temporal, porque as palavras são recebidas temporalmente, compondo-as em sentenças que também são compostas temporalmente para formar as histórias. O caminho inverso também é realizado: a partir da representação das histórias forma-se sentenças e palavras correspondentes. Embora alguns módulos realmente realizem um reconhecimento temporal de padrões, o controle geral do processo está claramente expresso na estrutura do sistema formado pelos módulos: o fato de que forma-se sentença por sentença a partir das palavras, e as sentenças por sua vez formam cada episódio reflete-se no fato de que há um módulo para cada uma destas tarefas, organizados sequencialmente. Além disso, há camadas de unidades para diversos níveis de detalhamento da história, contidas em diferentes módulos.

Sistemas estruturados também são usados para contornar o problema da ligação de variáveis. Através de um controle mais elaborado, é possível interpretar as variáveis de várias maneiras, estabelecendo ligações arbitrárias entre elas.

Sun (1995), por exemplo, propõe a organização de uma rede conexionista que cria construtos (assemblies) de unidades representando, cada um deles, um frame ou predicado. Um frame representa uma entidade com uma série de atributos (Minsky, 1975), como por exemplo um carro que possui atributos tais como marca, dono, ano de fabricação etc. No esquema proposto por Sun, um frame é formado por vários slots, que podem ser sub-redes conexionistas, preenchidos com seus respectivos valores. Um predicado é formado por slots contendo seus argumentos e seu valor verdade. Diversas conexões entre os componentes de frames e predicados fazem valer as relações entre eles de acordo com uma base de conhecimento prévia. Neste modelo, esta estruturação é usada, entre outras coisas, para estabelecer a ligação de variáveis. Se um frame possui sub-frames, por exemplo, eles são representados *dentro* do slot correspondente. Sua posição na rede, portanto, indica qual o frame a que pertencem. Da mesma maneira, funções empregadas em predicados devem estar representadas inteiramente dentro de um certo slot deste predicado. Naturalmente, um tal esquema não dá conta do fato de que a mesma função pode ser usada em dois predicados diferentes; ela deverá ser replicada em duas instâncias.

O modelo de Sun provê também um exemplo de controle conexcionista. Cada frame ou predicado tem em sua representação um *slot de consistência*. Este slot tem a função de controlar a propagação dos padrões apenas quando eles tiverem se estabilizado (já que o completamento de um padrão de um slot pode levar vários ciclos) e de permitir apenas inferências válidas (slots só recebem entrada de outros se estes outros slots exibirem um padrão consistente com o que o slot de consistência foi programado para aceitar). Trata-se de um controle descentralizado, independente de um executor externo, mas dependente da estrutura, montada a partir da especificação deste controle.

Uma abordagem semelhante é a do sistema μ KLONE (Derthick, 1990). O autor aponta a utilidade de redes conexcionistas para encontrar modelos mais plausíveis de uma determinada teoria lógica, ao invés de procurar manter-se fiel a todos os modelos possíveis. A partir de uma teoria, constrói-se uma rede que representa seus modelos possíveis e ruma para o mais plausível dentre eles a partir de uma parte inicial de sua especificação. Para conseguir esta inferência intuitiva, entretanto, a rede possui módulos específicos para representar o sujeito analisado e para representar pares de papéis e constituintes ativos, uma unidade para cada par. Apesar de construída a partir de uma teoria lógica, a rede oferece diversos recursos tipicamente conexcionistas, como a generalização automática de entidades e consideração paralela de várias restrições suaves.

A utilização de estruturas provê um meio eficiente de realizar aplicações e testar hipóteses sobre representações, mas ainda não oferece uma teoria totalmente coerente com os processos tratados. Para tanto, os modelos deveriam ser flexíveis, aceitando mudanças de acordo com aprendizado e apresentando a capacidade de combinar elementos arbitrariamente, mesmo os adquiridos durante o aprendizado ou desempenho da tarefa. As redes estruturadas, ao contrário, são rígidas em sua constituição e têm capacidade limitada em princípio (por exemplo quando oferecem uma unidade para cada par papel/constituente, ou conjuntos de unidades representando determinadas entidades).

Uma alternativa à estruturação dos modelos de acordo com o domínio é a formação dos recursos necessários através dos métodos conexcionistas usuais de treinamento. Bienenstock (1995) aponta que certos pesquisadores como von der Malsburg sugerem uma abordagem desenvolvimental, onde mecanismos capazes de prover controle e composicionalidade (ligação de variáveis) surgem como resultado de um processo de auto-organização. Se, como comentou-se em 3.5.3, o controle baseia-se em um outro tipo de conhecimento que não o do domínio, pode-se supor que este conhecimento pode ser adquirido por um modelo mais simples pelas vias usuais de treinamento, para ser usado em aprendizado mais específico de um determinado domínio.

Naturalmente, é sempre necessária uma estrutura prévia. Isto é ainda mais verdadeiro no caso de problemas de alto nível, que baseiam-se em níveis mais baixos que devem ser previamente conhecidos. Os próprios modelos conexcionistas atuais possuem a sua estrutura prévia, presente em seu modo de funcionamento e conectividade. Entretanto, deve-se escolher a estrutura que melhor disponibilize soluções sem restringir demais as

possibilidades dos modelos. A estrutura presente nos modelos conexionistas estruturados disponibilizam soluções ao custo de uma rigidez excessiva, falta de escalabilidade e dificuldade de aproveitamento em outros domínios. Inserir estruturas prévias até o ponto exato de equilíbrio entre poder computacional e flexibilidade é uma questão essencialmente empírica.

3.6 Síntese

Determinados problemas cognitivos, descritos como sendo de “alto nível,” parecem poder beneficiar-se das características positivas de redes conexionistas, tais como generalização automática, aprendizado por exemplos e degradação suave. Entretanto, não é fácil empregar estes modelos diretamente em sua resolução. Analisando-os mais de perto, distingue-se um número de características a eles inerentes que parecem desafiadoras aos modeladores conexionistas que não querem perder as vantagens de seus modelos: a composicionalidade, a sistematicidade, a coerência inferencial. Confrontando-se estas características às propriedades mais marcantes dos modelos conexionistas, surgem certos obstáculos a serem vencidos: múltiplas instâncias de uma mesma classe, ligação de variáveis, controle em sequências temporais na resolução de problemas. Estes problemas podem ser apenas contornados por alguns artifícios, como a utilização de estruturas *ad-hoc* e rígidas. Ainda não se conhece soluções gerais. A necessidade de uma estrutura prévia é real, mas busca-se uma que não invalide as propriedades positivas dos modelos conexionistas. Nesta direção, um caminho promissor é o de uma abordagem desenvolvimental, mas resta saber se a estrutura atual é suficiente para suportá-la.

Quanto à oposição entre IA conexionista e simbólica, pode-se notar que a idéia de que uma das abordagens pode representar uma solução completa é um tanto enganadora. A cognição contém problemas que exigem o melhor de cada um. Cada um pode, com custo relativamente elevado, realizar o que o outro faz. Assim, trata-se de uma questão de verificar como cada um pode fazer o que o outro faz, e descobrir qual pode fazê-lo com o menor custo. Uma possibilidade é que isto os leve a encontrar-se no meio do caminho, ainda que um possa chegar mais rapidamente do que o outro. Outra possibilidade é que um ou ambos os caminhos não cheguem muito longe. É o esforço de desenvolvimento dos dois lados que vai responder a estas questões.

No lado conexionista deste processo, o lado desenvolvimental é uma trilha promissora. Ele se baseia na idéia de o aprendizado de uma rede conexionista ser separado em várias fases, onde cada uma prepara o terreno para as seguintes. O próximo capítulo exemplifica um aprendizado em várias fases, exibindo um modelo que o empregue.

4. Um modelo conexionista com aprendizado de múltiplas fases

4.1 *Alto nível e a questão do conhecimento prévio*

Normalmente os elementos em um nível alto de processamento são considerados mais complexos e significativos do que os de níveis mais baixos. Esta complexidade e significado são obtidos através de sistemas de interpretação previamente conhecidos que são aplicados sobre estes elementos. Por exemplo, pode-se considerar que elementos em níveis altos de descrição são normalmente tipados, isto é, pertencem a uma determinada classe. O sistema de classes utilizado em certo contexto é usualmente conhecido a priori e densamente relacionado: conhece-se propriedades de cada uma das classes assim como relações existentes entre elas, como as de inclusão, similaridade etc. Assim, quando há uma referência a um objeto de uma das classes, não apenas o objeto como entidade atômica, mas também suas propriedades conferidas por seu tipo e relações são levadas em conta em qualquer tratamento que ocorra.

Desta maneira, problemas de alto nível sempre dependem de estruturas de fundo conhecidas previamente. Em sistemas classicistas, a estrutura de fundo normalmente necessária à interpretação correta de elementos em um problema de alto nível está presente como característica indissociável deste sistema. Pode-se por exemplo contar com a noção de símbolo, de conjuntos, de regra e assim por diante. Isto tem sido fonte das principais críticas aos modelos conexionistas, pois estas estruturas de fundo são por vezes consideradas essenciais ao tratamento dos problemas relacionados à cognição e os modelos conexionistas, não dispondo destes recursos a princípio, seriam incapazes de tratá-los. A resposta conexionista à esta crítica é a de que estas estruturas não devem ser fixadas rigidamente mas desenvolvidas de acordo com o problema tratado. Desta maneira as técnicas de aprendizado e características positivas do funcionamento das redes podem ser utilizadas tanto nesta estrutura de fundo quanto nos elementos de um problema em particular. Nos problemas em que a estrutura de fundo é muito complexa, entretanto, arca-se com um custo adicional de aprendizado (que pode ser muito alto e até proibitivo).

No treinamento de uma rede conexionista, a estrutura de fundo para um tipo de problema pode ser aprendida ao mesmo tempo em que a rede é treinada para resolver uma instância deste tipo de problema. Muitas vezes, porém, isto não será tão conveniente, pois nada impede que uma mesma estrutura de fundo possa ser usada diversas vezes como ponto de partida do aprendizado de instâncias específicas de problemas. Pode-se introduzir em um modelo, por exemplo, categorias de veículos e fatos sobre elas em um sistema que servirá de base no tratamento de diversos problemas envolvendo veículos. Desta maneira não será necessário realizar o aprendizado dos fatos básicos novamente a cada aplicação do sistema.

Realizar o aprendizado em fases distintas traz também a vantagem de obter conhecimento mais correto e geral. Tentar realizar o aprendizado de uma estrutura básica e uma certa

instância de aplicação ao mesmo tempo pode levar a simplificações devidas a peculiaridades daquela determinada instância. Em um problema envolvendo fatores climáticos em países no hemisfério sul, por exemplo, fatos básicos que serviriam como estrutura de fundo, como por exemplo o conhecimento das estações do ano, poderiam ser mal aprendidos: o verão poderia ser identificado erroneamente como indissociável dos meses de dezembro a março. Se estes fatos básicos fossem treinados em uma fase distinta e dedicada, poderia ficar bem estabelecido a definição correta de verão e apenas mais tarde usar-se a especialização para o caso do hemisfério sul.

Outras vantagens da divisão por fases diz respeito a redundâncias. Se dois corpos de conhecimento possuem um certo grau de redundância, ela pode ser representada duplamente em uma rede conexionista se o aprendizado destes dois corpos realizar-se ao mesmo tempo. Entretanto, se apresentada em uma fase anterior de modo independente, esta parte redundante pode mais tarde integrar-se a cada um deles em aprendizados posteriores. Ao aprender determinados fatos que mantêm redundância com outros já conhecidos, um sistema pode absorver apenas aquilo que é realmente informação nova, mantendo o máximo de generalidade e parcimônia em sua descrição do que recebeu. Se a redundância não é um simples acaso, mas sim consequência de uma real relação entre dois corpos de conhecimento, então é útil que seja representada de forma unificada pelo sistema pois, caso sofra alterações, estas devem refletir-se da mesma maneira sobre ambos.

Neste capítulo é apresentado uma aplicação de uma rede conexionista organizada de maneira que um corpo de conhecimento básico é apresentado a uma rede e usado posteriormente para aprender novos fatos. Com isso, mostra-se que:

1. é possível utilizar uma rede conexionista com aprendizado em fases que se integram;
2. este conhecimento é melhor formado por ser mais correto e flexível;
3. há aumento de desempenho em determinadas situações;
4. as características positivas de redes conexionistas, como a degradação suave, são preservadas neste tipo de aplicação.

4.2 O modelo conexionista utilizado

Nesta seção, esboça-se algumas características necessárias a um modelo para que se possa realizar o aprendizado em múltiplas fases.

Cada fase de treinamento mantém a característica usual de modelos conexionistas de aprender através da apresentação de exemplos ou pares de entrada e saída. Além deste tipo de aprendizado ser o mais natural em muitos tipos de problemas cognitivos, pode-se desta maneira preservar os métodos conhecidos para realizar o aprendizado sobre a rede de elementos processadores simples de forma distribuída, apresentando degradação suave e criando representações internas de acordo com a necessidade do sistema.

Para integrar as diferentes fases de aprendizado, entretanto, é necessário que não haja partes da rede destinadas necessariamente a representar entradas e outras a representar saídas, pois muitas vezes o que foi saída em uma das fases será entrada em outras. O tipo de rede que apresenta esta versatilidade é o de redes recorrentes, e será portanto deste tipo o modelo utilizado no exemplo dado.

Para manter a capacidade altamente desejável de desenvolver representações internas e não limitar-se a problemas linearmente separáveis, é útil que o modelo possua camadas ocultas que possam conter representações internas úteis. Para tanto, pode-se utilizar backpropagation como método de aprendizado.

A recorrência é normalmente observada em modelos com uma única camada. No caso de mais de uma camada, ela é obtida simplesmente igualando os valores de ativação das unidades da primeira camada aos valores de ativação das unidades da última camada após cada iteração. Este ciclo é repetido até que o valor destas unidades se estabilize.

É interessante que no modelo utilizado as unidades possam influenciar-se umas às outras tanto positivamente quanto negativamente. Por isso, a representação bipolar, em que cada unidade pode assumir valores de ativação na faixa de -1 a 1, é utilizada. Por outro lado, é importante que a rede seja formada por unidades com alguma estabilidade ao redor do valor de ativação zero. Caso contrário, devido à sua recorrência, mesmo as conexões com valores muito próximos de zero teriam uma influência aumentada a cada iteração que tornar-se-ia determinante para o valor de várias unidades. Isto pode ser favorável em diversos modelos, como no BSB, por exemplo, mas no caso de treinamento em múltiplas fases, com diversos sub-domínios com pouca ou nenhuma relação entre si, seria muito desfavorável. Devido a isso, foi escolhida uma função de transferência que combina os dois pólos de unidades bipolares e estabilidade ao redor de zero. Trata-se de uma sigmóide obtida através da “colagem” de duas sigmóides binárias, cada uma delas cobrindo uma das faixas $(-1, 0)$ e $[0, +1)$ (figura 4-1). Esta “colagem” foi realizada de maneira que a derivada da função fica bem definida em todo o domínio da função, possibilitando a utilização do método de backpropagation normalmente.

figura 4-1: a função sigmóide dupla é usada concatenando-se duas sigmóides binárias, cada uma cobrindo uma das faixas $(-1, 0)$, $[0, +1)$. Assim, ela pode funcionar com unidades bipolares mas mantendo uma região de estabilidade em torno do valor 0.

Em redes recorrentes é interessante que a entrada apresentada fique presente por tempo suficiente sobre as unidades correspondentes para que possa influenciar as demais. Quando se busca um completamento de padrões a partir de uma certa entrada, como é o caso no exemplo apresentado, é interessante que esta entrada permaneça até o final do processamento. Isto é realizado fazendo-se com que os valores das unidades de entrada sejam repassadas através de todas as camadas, até a última. Isto é possível se, quando da inicialização dos pesos, cada unidade seja fortemente conectada à unidade correspondente na camada acima e tornando muito grande a chance desta reproduzir o seu valor. A criação de representações internas diferenciadas, entretanto, é assegurada com a introdução de mais unidades ocultas do que unidades nas primeira e última camadas. As unidades ocultas além daquelas correspondentes às unidades de entrada não são fortemente conectadas a nenhuma unidade em particular e ficam livres para desenvolver a representação interna que for necessária.

Desta maneira, a rede pode ser esboçada como na figura 4-2. Alimentada com qualquer trecho de entrada, propagará esta informação através das várias camadas, mantendo a entrada através das várias iterações, e ao mesmo tempo usando as conexões determinadas durante o aprendizado e as representações internas desenvolvidas para completar sucessivamente as unidades que podem ter seu valor determinado, enquanto aquelas que não podem ter este valor determinado através da entrada presente ficam estabilizadas no valor de ativação zero.

figura 4-2: A rede conexionista utilizada para o exemplo apresentado possui um conjunto de unidades que mantém o padrão a ser completado em recorrência enquanto conta também com um conjunto de unidades ocultas que podem desenvolver representações internas convenientes.

O método de backpropagation utilizado neste modelo é primariamente o referente a um único passo. Isto significa que os pesos são alterados levando-se em conta os valores de ativação das unidades apenas na última iteração. Portanto, garante-se que o erro vai diminuir apenas na situação em que uma única iteração for utilizada e por isso, para garantir a convergência da rede durante o aprendizado, este é normalmente realizado utilizando-se uma única iteração por padrão e corringindo-se os pesos em seguida. Embora garantindo a convergência para um erro mínimo (local), o aprendizado realizado desta maneira apresenta a desvantagem de não utilizar conhecimento já obtido que leve mais de uma iteração para exibir resultados e com isso corre o risco de realizar aprendizado de associações que na verdade já aprendeu.

Por exemplo, suponha que ao receber o valor +1 em certa unidade u a rede leve a unidade v a conter o valor de ativação -1 utilizando conhecimento já obtido, mas que este processo leve duas iterações para ocorrer. Se durante aprendizado posterior a rede recebe um padrão com u igual a +1 e deseja uma saída em v igual a -1, isto não ocorrerá na única iteração que o aprendizado utiliza e o método de aprendizado mudará os pesos para que v receba valor -1 a partir de u igual a + 1 em apenas uma iteração, quando isto não era absolutamente necessário.

Pode-se entretanto tomar a decisão de permitir que a rede itere por um número limitado de vezes durante o aprendizado para utilizar conhecimento prévio. Entretanto o método de backpropagation usado continua sendo o de um único passo. Como as correções para uma rede que tenha iterado algumas poucas vezes mantêm uma forte relação com as

correções para uma rede que tenha iterado apenas uma vez, este método de aprendizado de um único passo serve na grande parte das ocasiões, mas menos eficientemente. Ou seja, pode-se realizar o aprendizado utilizando-se conhecimento prévio, mas paga-se um preço por isto. Convém lembrar ainda que existem vários métodos na literatura para aprendizado de redes recorrentes (por exemplo em Rumelhart *et al.*, 1986; Hertz *et al.*, 1991), cada um apresentando vantagens e desvantagens específicas em termos de desempenho, memória ou tipo de rede.

4.3 O exemplo apresentado

O domínio escolhido para exemplo é o de controle de máquinas em um ambiente industrial. Determinadas peças, designadas pelas letras de 'A' a 'G', devem passar por certo tratamento. Este tratamento será constituído de uma dentre quatro operações possíveis (numeradas de 1 a 4), determinada de acordo com o formato e material da peça usada. O formato de cada peça pode ser o de um cubo, cilíndrico ou o de um disco. O material pode ser aço ou alumínio. As operações são realizadas por certa máquina, que para isto recebe um código correspondente. No nosso exemplo, as operações de 1 a 4 possuem códigos de W, X, Y e Z respectivamente.

Embora todos os dados estejam já determinados, o exemplo é concebido tendo-se em mente que várias relações podem mudar. Novas peças podem ser introduzidas, bem como a operação a ser aplicada a determinado formato ou material. A máquina executora pode ser trocada e uma nova codificação das operações fazer-se necessária.

O objetivo do sistema será o de, fornecido o nome de uma peça, fornecer o código a alimentar a máquina executora. Um lote com exemplos de nomes de peças e os códigos correspondentes a ser utilizados será treinado com e sem treinamento prévio e os resultados correspondentes analisados.

O conhecimento envolvido neste domínio pode ser esquematizado como na figura 4-3. Note que a partir de determinado nome de peça, pode-se deduzir qual o código a ser fornecido à máquina através de dois caminhos diferentes: através de um mapeamento direto usando os padrões fornecidos para treinamento, e através do conhecimento das relações entre os formatos e materiais das peças, a operação necessária para tratá-las e o código correspondente a esta operação. Entretanto, o código diretamente associado a uma determinada peça no lote de treinamento deve prevalecer. O conhecimento mais indireto que fala a respeito da relação entre o formato e material e a operação necessária deve ser encarado como algo que guia mas não determina com certeza a decisão tomada.

figura 4-3: o domínio usado como exemplo possui diversos tipos de informação com relações distintas entre si.

A parte mais complexa do domínio trata exatamente da relação entre formatos, materiais e operações necessárias. Trata-se de um conhecimento bastante independente do restante, e por isto tratá-lo separadamente pode ser útil. Mais importante, sua presença na rede pode ser usada quando do aprendizado de novos exemplos de peças. Mediante a definição do formato e material destas peças, o modelo pode deduzir a operação e consequentemente o código provável para ela. Desta maneira, só é necessário dispendir tempo de mudanças nas conexões para as peças que contrariarem estas previsões e o conhecimento a respeito destas relações fica unicamente representado.

4.4 Ferramentas para o treinamento

Cada situação ou solução do problema tratado é representado como um padrão de valores de ativação sobre as unidades de entrada da rede. Inicialmente alguns destes valores são iguais a zero; isto significa que seu valor no domínio do problema não está definido.

Cada variável pertencente ao domínio ocupa uma ou mais posições no padrão que representa uma situação. Este conjunto de posições é chamado de *slot*. Para facilitar a leitura dos lotes de treinamento foi estabelecida uma linguagem de representação destes padrões a partir de seus slots. Uma linguagem é definida estabelecendo quais os slots

disponíveis e seus valores possíveis, bem como os valores de ativação das unidades correspondentes a cada valor do slot. No domínio escolhido para o exemplo, a linguagem usada é a seguinte:

```
linguagem l (slot ("formato",
                  "cubo",      "-1 -1",
                  "disco",     "-1 +1",
                  "cilindro", "+1 +1",
                  ),
  slot ("material",
        "aço",      "+1",
        "alumínio", "-1",
        ),
  slot ("operação",
        "1", "-1 -1",
        "2", "-1 +1",
        "3", "+1 -1",
        "4", "+1 +1",
        ),
  slot ("código",
        "W", "-1 -1",
        "X", "-1 +1",
        "Y", "+1 -1",
        "Z", "+1 +1",
        ),
  ...
  slots A, B, C, D, E, F, G definidos todos como:
  slot("A",
        "V", "+1",
        "F", "-1",
        ),
  ...
  );
```

Definida a linguagem e os slots possíveis, os padrões podem ser representados por sentenças desta linguagem. Um exemplo é composto pelos valores dos slots fornecidos e os desejados. Os slots fornecidos são acompanhados pelo caractere `:`. Os desejados são acompanhados por `?`. Assim, o exemplo

`formato:cubo material:aço operação?2`

significa que o valor desejado para o slot `operação` é 2 quando forem fornecidos os valores `cubo` e `aço` para os slots `formato` e `material` respectivamente.

Slots como o slot `A` apresentados acima, ou seja, com valores `V` e `F` somente, são considerados booleanos e recebem uma notação especial nas sentenças: `!A` é equivalente a `A:F`, `A` a `A:V`, `A?` a `A?V` e `!A?` a `A?F`. No exemplo dado, o slot `A` ter valor `V` significa que a peça sendo tratada é a `A`, o slot `B` ter valor `V` significa que a peça sendo tratada é a `B` e assim por diante.

Os lotes de treinamento usados são os seguintes:

operação:

formato:cubo	material:alumínio	operação?1
formato:cubo	material:aço	operação?4
formato:cilindro	material:alumínio	operação?4
formato:cilindro	material:aço	operação?3
formato:disco	material:alumínio	operação?2
formato:disco	material:aço	operação?4

código:

operação:1	código?W
operação:2	código?X
operação:3	código?Y
operação:4	código?Z

peças:

A	formato?cilindro	material?aço
B	formato?cubo	material?alumínio
C	formato?disco	material?alumínio
D	formato?cilindro	material?aço
E	formato?cilindro	material?alumínio
F	formato?disco	material?aço
G	formato?cubo	material?aço

direto:

A	código?Y
B	código?W
C	código?X
D	código?Y
E	código?Z
F	código?Z
G	código?Z

A execução da rede é demonstrada traduzindo os sucessivos estados das unidades para sentenças na linguagem definida e exibindo aqueles diferentes do anterior. Desta forma a rede começa exibindo a linguagem fornecida e prossegue até a estabilização, quando chega à sentença mais completa possível. Alguns exemplos de execuções são os seguintes:

```
***> A
A formato:cilindro material:aço
A formato:cilindro material:aço operação:3
A código:Y formato:cilindro material:aço operação:3

***> operação:2
código:X operação:2
```

A ordem dos slots nas sentenças traduzidas é alfabética.

4.5 Os testes realizados e o que eles indicam

O primeiro teste realizado é se o modelo conexcionista pode realmente receber os diversos lotes de treinamento de forma separada e integrá-los convenientemente. Treinando a rede com os lotes *operação, código, peças* (sem *direto*), já obtemos o funcionamento correto para todas as peças. Eis algumas execuções obtidas:

```
***> C
C formato:disco material:alumínio
C formato:disco material:alumínio operação:2
C código:X formato:disco material:alumínio operação:2

***> formato:cubo material:aço
formato:cubo material:aço operação:4
código:Z formato:cubo material:aço operação:4

***> F
F formato:disco material:aço
F formato:disco material:aço operação:4
F código:Z formato:disco material:aço operação:4
```

Além de reproduzir corretamente exemplos que estavam nos lotes de treinamento, a rede também reage de forma razoável na presença de entradas incompletas, corrompidas ou contraditórias:

```
***> B C
B C material:alumínio
B C material:alumínio operação:2
B C código:X material:alumínio operação:2

***> formato:cubo
formato:cubo operação:1
código:W formato:cubo operação:1
```

Quando confrontado com a ambiguidade de qual peça se trata (B ou C), o modelo infere que o material é o alumínio (comum às duas). Quando confrontado apenas com o formato ou com o material (no lote *operação* havia sempre as duas informações juntas), o modelo oferece uma operação que tenha sido treinada com aquele formato ou material em algum exemplo. Ou seja, a rede continua exibindo as propriedades de degradação suave e generalização típicos dos modelos conexcionistas.

Se a rede que já tenha sido treinada com os lotes *operação*, *código* e *peças* receber o treinamento para o lote *direto*, este treinamento é realizado com apenas uma passagem sobre o lote. Isto ocorre porque toda a informação do lote já está na verdade aprendida pela rede. Ela oferece os resultados de acordo com os resultados desejados do lote e nenhuma alteração é necessária.

Se o lote *direto* é substituído pelo lote abaixo:

direto *inexato*:

A código?Y
B código?W
C código?X
D código?W (resposta prevista é Y)
E código?Z
F código?Y (resposta prevista é Z)
G código?Z

e seu treinamento é realizado sobre o modelo que já tenha treinado *operação*, *código* e *peças*, então é realizado algum aprendizado e tem-se as seguintes execuções:

```
***> D
D formato:cilindro material:aço
D código:W formato:cilindro material:aço operação:3
```

```
***> F
F formato:disco material:aço
F formato:disco material:aço operação:4
F código:Y formato:disco material:aço operação:4
```

```
***> A
A formato:cilindro material:aço
A formato:cilindro material:aço operação:3
A código:Y formato:cilindro material:aço operação:3
```

Ou seja, as associações que não estavam de acordo com o conhecimento anterior (os códigos para as peças D e F) foram aprendidas sem destruí-lo (pois para outras peças ele continua sendo utilizado com proveito). Note que as peças A e D são iguais, ou seja, possuem o mesmo formato e são feitas do mesmo material; entretanto, o fato da peça ser D é suficiente para que o código W seja ativado a despeito do formato e material, enquanto que para A estas informações continuam a ser usadas para obter-se o código Y de acordo com o conhecimento prévio.

Um dos aspectos mais interessantes do treinamento em múltiplas fases é o de que o conhecimento prévio pode ser alterado de maneira que os outros itens continuem funcionando. Se o modelo que foi treinado com *operação*, *código*, *peças* e *direto* receber o treinamento do seguinte lote de uma única sentença:

alteração:

formato:disco material:aço operação:3 (antes era 4)

após algum aprendizado obtemos as seguintes execuções:

```
***> formato:disco material:aço  
formato:disco material:aço operação:3  
código:Y formato:disco material:aço operação:3
```

```
***> F  
F formato:disco material:aço  
F código:Y formato:disco material:aço operação:3
```

ou seja, o processamento para a peça F, que havia sido treinada após o treinamento para obter operação a partir de formato e material, reflete automaticamente a mudança ocorrida.

4.6 Limitações

Como em qualquer modelo conexionista, também o modelo empregado neste exemplo possui suas limitações quanto a tolerância à degradação e conhecimento incompleto ou inconsistente. Naturalmente, quando um novo conjunto é treinado sobre um modelo que já sofreu treinamento há a possibilidade de o novo aprendizado alterar os pesos das conexões de maneira a destruir o que foi aprendido antes. Isto é particularmente verdadeiro se o novo conhecimento for de alguma maneira inconsistente com o conhecimento anterior. Porém em treinamentos de múltiplas fases o caso típico é de dispor corpos de conhecimento bastante ortogonais em diferentes fases, o que facilita sua integração sem mútua anulação, como ocorreu no nosso exemplo. De qualquer maneira, pode-se evitar este tipo de efeito mantendo em algum tipo de memória auxiliar os padrões que já foram treinados e mantê-los nos treinamentos seguintes para que não se corra o risco de perder a capacidade de reproduzi-los.

Quando se opta pela utilização de conhecimento prévio, permitindo um número de iterações antes da correção de erros, a eficiência do backpropagation diminui consideravelmente. Isto ocorre em parte pelo aumento da dificuldade do problema em si, e em outra parte pela opção de continuar utilizando o método de backpropagation de um único passo mesmo nesta situação. Pode-se obter uma melhora desta eficiência através de métodos mais apropriados a redes recorrentes, ainda que a um maior custo em termos de espaço de armazenamento.

Além disto, este modelo ainda está sujeito à mais típica das limitações de modelos conexionistas, a de que um mínimo local da função de erro pode ser atingido ao invés de um mínimo global e certos padrões não sejam corretamente reproduzidos.

4.7 Desempenho

Pode-se perguntar se há ganho de desempenho com o treinamento do modelo em múltiplas fases, ou seja, se o número de ciclos de aprendizado da rede (ou seja, o número de vezes em que um padrão foi propagado e a rede corrigida em função de uma saída desejada) é menor quando se adota esta técnica.

De fato, existe uma economia de ciclos dentro de determinadas situações. Separando e antecipando uma parte mais básica e complexa do conhecimento, o número de ciclos necessários para aprender fatos complementares é bem menor. Em nosso exemplo, o corpo de conhecimento que relaciona formato e material com operação é a parte mais básica e complexa. Embora isto tenha variado de execução para execução de acordo com a disposição aleatória dos pesos iniciais, o número mínimo de ciclos necessários foi de 50.232. O restante do conhecimento, sobre as peças e os códigos das operações, levou sempre menos de 1.000 ciclos. Portanto, se uma cópia da rede após o treinamento mais básico for mantida, pode-se introduzir conhecimento complementar a um custo bem mais baixo. A economia será realizada nas situações em que realmente há um mesmo conhecimento básico para diversas situações de detalhes. Entretanto, mesmo no caso em que apenas uma situação de detalhes era objetivada, o treinamento em múltiplas fases não significou decréscimo de desempenho. Pelo contrário: ao treinar-se todos os lotes de treinamento reunidos em um único lote, não se obteve nenhuma vez um número de ciclos menor do que o menor obtido no treinamento em múltiplas fases. O menor número de ciclos obtidos foi de 240.686. Além disso, um maior número de erros de aprendizado foi observado.

Outro tipo de economia de ciclos que esta técnica produz é a referente à utilização de conhecimento anterior quando do aprendizado de um certo lote de treinamento. Como foi dito anteriormente, ao contemplar o conhecimento anterior a rede só precisa realizar o aprendizado para gravar os exemplos que o contrariam. Quando não há exemplos que contrariem o conhecimento anterior, então esta economia é máxima: não é necessário aprendizado algum.

Entretanto, se há exemplos que contrariam o conhecimento, não existe uma economia em todos os sentidos. O aprendizado utilizando mais de uma iteração, para que o conhecimento anterior possa ser melhor aproveitado, é mais caro, e por isso é mais barato treinar os novos exemplos sem preocupação com o conhecimento anterior; o treinamento é mais rápido e eles são gravados pela rede de qualquer maneira. Neste caso específico, o treinamento em múltiplas fases não representa um aumento de desempenho. Porém, continua representando uma melhoria de qualidade. Ao aprender os novos exemplos de acordo e integrados com o conhecimento prévio, a rede mantém uma descrição mais geral e parcimoniosa do conhecimento. Assim, se houver futuramente alguma alteração no conhecimento de base, os padrões aprendidos refletirão a influência disto; se tiverem sido simplesmente gravados, não.

4.8 Síntese

Com o exemplo de determinação de códigos a partir de peças em um ambiente industrial, mostrou-se como é possível que um modelo conexionista trate um certo domínio de forma modular e compartimentada sem que seja ele mesmo dividido em compartimentos a priori. Com isto pode-se beneficiar da compartimentação sem estar-se limitado a um esquema completamente fixo.

O exemplo mostra também como dividir um domínio em sub-domínios e treiná-los separadamente pode trazer benefícios à qualidade da representação formada, aumentando a generalização e corretude das respostas. Isto permite também que redes sejam treinadas em etapas e em situações em que mais de uma etapa final for possível ocorre uma economia de tempo de treinamento.

Com isto, obtém-se uma forma mais geral de aprendizado em redes conexionistas e passa-se a contar com o recurso de utilização de conhecimento prévio que, como foi visto anteriormente, é um dos elementos necessários ao tratamento de problemas de alto nível. Isto é obtido, ainda, sem prejuízo para outras características de modelos conexionistas, como a degradação suave, representações distribuídas, generalização automática e aprendizado baseado em exemplos.

5. Conclusão

O tratamento de problemas de alto nível por modelos conexionistas mostra-se como uma atraente expansão da aplicação destes modelos que têm recebido aperfeiçoamentos constantes desde a década de 70. Suas vantagens no tratamento de problemas de baixo nível parecem adequadas à abordagem de determinados problemas na Inteligência Artificial que trata de problemas de alto nível.

A expansão das capacidades destes sistemas nesta direção depende fundamentalmente da compreensão de suas propriedades e restrições e das características intrinsecamente relacionadas a problemas de alto nível. A primeira parte foi abordada no capítulo 2, que procurou oferecer uma visão geral dos modelos com suas diferentes capacidades, enquanto a segunda foi tratada no capítulo 3, quando os processos de alto nível e sua abordagem por modelos conexionistas foram examinados mais de perto.

No capítulo 2 o funcionamento de diversos tipos de modelos foi apresentada e uma síntese dos modelos conexionistas foi realizada com o objetivo de elucidar seus pontos em comum e os motivos que levam às suas vantagens e desvantagens, de forma a tornar mais claras as suas dificuldades com problemas de alto nível abordadas no capítulo seguinte.

No capítulo 3 foram abordadas as possibilidades e dificuldades de integração dos domínios dos modelos conexionistas e dos problemas de alto nível. Esta discussão passou ainda pela questão da necessidade de símbolos no tratamento destes problemas. Foi constatado que mais do que uma simples disputa entre duas abordagens excludentes, a oposição entre sistemas simbólicos clássicos e modelos conexionistas relaciona-se com o tratamento dos mesmos problemas através da valorização diferenciada de ferramentas disponíveis a ambas as partes, que podem continuar desenvolvendo-se em busca da maior ampliação de suas possibilidades dentro do menor custo possível.

Foi mostrado que fatores importantes no tratamento de alto nível são as assim chamadas, nesta dissertação, de ferramentas de controle de complexidade, tais como a manipulação de estruturas, de processos temporais e manutenção de múltiplos objetos da mesma classe. Ainda, é importante a divisão do conhecimento em níveis, formados a partir da utilização destas ferramentas e fornecimento do conhecimento em fases.

Pouco se sabe a respeito do tratamento destas características por parte de modelos conexionistas. Várias tentativas e abordagens foram apresentadas, considerando suas eventuais limitações. As principais delas são as que tratam da representação de estruturas mas que deixam sua manipulação a cargo de controle temporal externo, e aquelas que utilizam o conhecimento disponível do problema a ser tratado para realizar uma estruturação dos modelos em módulos *ad hoc* e fixa.

A possibilidade de sistemas que se estruturam convenientemente com aprendizado por exemplos em várias fases também foi contemplada e um modelo que utiliza esta abordagem foi apresentado. Este sistema foi capaz de tratar um certo número de subdomínios separadamente e integrá-los de forma proveitosa, inclusive durante o aprendizado, que pôde beneficiar-se de conhecimento já presente na rede. Estas vantagens foram obtidas sem perda das características positivas de modelos conexionistas como degradação suave e generalização automática.

Outros problemas, principalmente o da ligação de variáveis e controle de processos temporais, são cruciais, e a ausência de seu tratamento adequado limita fortemente a expansão das aplicações dos modelos conexionistas. Estes problemas devem continuar a ser objeto de pesquisa para que se conheça os limites do que sistemas conexionistas podem realmente oferecer e, com isto, para que se aumente a compreensão das necessidades envolvidas no tratamento da cognição.

6. Bibliografia

- Aarts, E. H. L., Korst, J. H. M. (1995) Boltzmann Machines. In *Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.). MIT Press. Cambridge, MA, EUA. pp 162-165.
- Anderson, J. A. (1995) *An Introduction to Neural Networks*. MIT Press. Cambridge, EUA.
- Barnden, J. (1995) Artificial Intelligence and Neural Networks. In *Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.). MIT Press. Cambridge, MA, EUA. pp 98-102.
- Barnden, J. (1995a) High Level Reasoning, Computational Challenges for Connectionism. *Applied Intelligence* 5, pp 103-135.
- Barnden, J. (1995b) Semantic Networks. In *Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.). MIT Press. Cambridge, MA, EUA. pp 854-857.
- Bienenstock, E., Geman, S. (1995) Compositionality in Neural Systems. In *Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.). MIT Press. Cambridge, EUA. pp 223-226.
- Carpenter, G., Grossberg, S. (1995) Adaptive Resonance Theory (ART). In *Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.). MIT Press. Cambridge, MA, EUA. pp 79-82.
- Derthick, M. (1990) Mundane Reasoning by Settling Down on a Plausible Model. *Artificial Intelligence* 46. pp 107-157.
- Fodor, J., Pylyshyn, Z. (1988) Connectionism and cognitive architecture: A critical analysis. *Cognition* 28. pp 3-71.
- Gardner, H. (1987) *The Mind's New Science*. Basic Books.
- Hertz, J., Krogh, A., Palmer, R. G. (1991) *Introduction to the Theory of Neural Computation*. Addison-Wesley. Redwood City, CA, EUA.
- Hinton, G. (1990) Mapping Part-Whole Hierarchies into Connectionist Networks. *Artificial Intelligence* 46. pp 45-75.
- Hinton, G. E., Sejnowski, T. J. (1986) Learning in Boltzmann Machines. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (D. E.

Rumelhart, J. L. McClelland and the PDP Research Group, Eds.), vol. 1, *Foundations*. MIT Press. Cambridge, EUA. pp 282-317.

Hinton, G. E., McClelland, J. L., Rumelhart, D. E. (1986a) Distributed Representations. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (D. E. Rumelhart, J. L. McClelland and the PDP Research Group, Eds.), vol. 1, *Foundations*. MIT Press. Cambridge, EUA. pp 77-109.

Honavar, V., Uhr, L. (1995) *Artificial Intelligence and Neural Networks*. Academic Press. Boston, EUA.

Kohonen, T. (1983) *Self-Organization and Associative Memory*. 2^a ed. Springer-Verlag. Berlin.

Kohonen, T. (1995) Learning Vector Quantization. In *Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.). MIT Press. Cambridge, EUA. pp 537-540.

Kullback, S. (1959) *Information theory and statistics*. Wiley. Nova Iorque, EUA.

Maren, A., Harston, C., e Pap, R. (1990) *Handbook of Neural Computing Applications*. Academic Press. San Diego, EUA.

Miikkulainen, R. (1995) Script-Based Inference and Memory Retrieval in Subsymbolic Story Processing. *Applied Intelligence* 5. pp 137-163.

Minsky, M., Papert, S. (1969) *Perceptrons*. MIT Press. Cambridge, EUA.

Minsky, M. (1975) A Framework for Representing Knowledge. In *Mind Design* (J. Haugeland, ed.).

Newell, A. (1982) The Knowledge Level. *Artificial Intelligence* 18. pp 87-127.

Pinker, S., Prince, A. (1988) On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition* 28. pp 73-193.

Plate, T. (1995) Holographic Reduced Representations. *IEEE Transactions on Neural Networks* 6, pp 623--641.

Pollack, J. (1990) Recursive Distributed Representations. *Artificial Intelligence* 46. pp 77-105.

Ritter, H. (1995) Self-Organizing Feature Maps: Kohonen Maps. In *Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.). MIT Press. Cambridge, EUA. pp 846-851.

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks*. Cambridge University Press. Cambridge, Grã-Bretanha.

Rosenblatt, F. (1962) *Principles of Neurodynamics*. Spartan. Nova Iorque, EUA.

Rumelhart, D. E., Hinton, G. E., e Williams, R. J. (1986) Learning Internal Representations by Error Propagation. In *Parallell Distributed Processing: Explorations in the Microstructure of Cognition* (D. E. Rumelhart, J. L. McClelland and the PDP Research Group, Eds.), vol. 1, *Foundations*. MIT Press. Cambridge, EUA. pp 318-362.

Rumelhart, D. E., Zipser, D. (1986a) Competitive Learning. In *Parallell Distributed Processing: Explorations in the Microstructure of Cognition* (D. E. Rumelhart, J. L. McClelland and the PDP Research Group, Eds.), vol. 1, *Foundations*. MIT Press. Cambridge, EUA. pp 151-193.

Rumelhart, D. E., Smolensky, P., McClelland, J. L., e Hinton, G. E. (1986b) Schemata and Sequential Thought in PDP Models. In *Parallell Distributed Processing: Explorations in the Microstructure of Cognition* (J. L. McClelland, D. E. Rumelhart and the PDP Research Group, Eds.), vol. 2, *Psychological and Biological Models*. MIT Press. Cambridge, EUA. pp 7-57.

Rumelhart, D. E., McClelland, J. L. (1986) *Parallell Distributed Processing: Explorations in the Microstructure of Cognition* (J. L. McClelland, D. E. Rumelhart and the PDP Research Group, Eds.), vols. 1 e 2. MIT Press. Cambridge, EUA.

Schank, R., Abelson, R. (1977) *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates. Hillsdale, NJ, EUA.

Shastri, L. (1995) Structured Connectionist Models. In *Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.). MIT Press. Cambridge, EUA. pp 949-952.

Smolensky, P. (1986) Information Processing in Dynamical Systems: Foundations of Harmony Theory. In *Parallell Distributed Processing: Explorations in the Microstructure of Cognition* (D. E. Rumelhart, J. L. McClelland and the PDP Research Group, Eds.), vol. 1, *Foundations*. MIT Press. Cambridge, EUA. pp 194-281.

Smolensky, P. (1990) Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems. *Artificial Intelligence* 46. pp 5-46.

Sontag, E. D., (1995) Automata and Neural Networks. In *Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.). MIT Press. Cambridge, EUA. pp 119-122.

Sun, R. (1995) Schemas, Logics, and Neural Assemblies. *Applied Intelligence* 5. pp 83-102.

Touretzky, D. (1990) BoltzCONS: Dynamic Symbol Structures in a Connectionist Network. *Artificial Intelligence* 46. pp 5-46.

Werbos, P. (1995) Backpropagation: Basics and New Developments. In *Handbook of Brain Theory and Neural Networks* (M. Arbib, Ed.). MIT Press. Cambridge, EUA. pp 134-139.