

Anexo A

As Redes Neurais Artificiais

“A Coruja não tem exatamente um Cérebro, mas ela Sabe Coisas”

A. A. Milne (1882-1956): *Winnie-the-Pooh* (1926)

A.1 Introdução

A evolução natural deu ao cérebro humano muitas características desejáveis que não estão presentes na máquina de von Neumann (os computadores atuais) tais como (Jain e Mao, 1996):

- Paralelismo massivo
- Representação e computação distribuídas
- Habilidade de aprendizado
- Habilidade de generalização
- Adaptabilidade
- Processamento de informação contextual inerente
- Tolerância a falhas
- Baixo consumo de energia

É desejável que os dispositivos computacionais baseados nas redes neurais biológicas possuam algumas destas características. Veja na tabela a seguir (de Jain e Mao, 1996), a comparação entre o computador de von Neumann e o sistema neural biológico.

	<i>Computador de von Neumann</i>	<i>Sistema neural biológico</i>
<i>Processador</i>	Complexo Alta velocidade Um ou poucos	Simples Baixa velocidade Um grande número
<i>Memória</i>	Separado do processador Localizado Não-endereçável pelo conteúdo	Integrada com o processador Distribuída Endereçável pelo conteúdo
<i>Computação</i>	Centralizada Sequencial Programas armazenados	Distribuída Paralela Auto-aprendizado
<i>Confiabilidade</i>	Muito vulnerável	Robusta
<i>Especialidade</i>	Manipulações numéricas e simbólicas	Problemas perceptuais
<i>Ambiente operacional</i>	Bem definido, bem restrito	Pobrementemente definido, irrestrito

Cérebros e computadores digitais realizam tarefas bem diferentes e têm propriedades diferentes. Veja a tabela abaixo (de Russell e Norvig, 1995) que mostra uma comparação entre cérebros e computadores digitais (de 1994):

	<i>Computador</i>	<i>Cérebro humano</i>
<i>Unidades computacionais</i>	1 CPU, 10^5 portas	10^{11} neurônios
<i>Unidades de armazenamento</i>	RAM de 10^9 bits, disco de 10^{10} bits	10^{11} neurônios, 10^{14} sinapses ¹³
<i>Tempo de ciclo</i>	10^{-8} seg.	10^{-3} seg.
<i>Bandwidth</i>	10^9 bits/seg.	10^{14} bits/seg.
<i>Atualizações de neurônio/seg.</i>	10^5	10^{14}

A.2 O Neurônio Biológico

O neurônio típico (figura A.1A) tem muitos dendritos, usualmente ramificados, que recebem informação de outros neurônios e um único axônio que fornece como saída a informação processada, usualmente através da propagação de um *spike* ou *potencial de ação*¹⁴. O axônio se divide eventualmente em vários ramos que fazem sinapses com os dendritos e corpos celulares de outros neurônios.

A.2.1 Variantes do Neurônio Clássico

Este quadro simples se torna complicado nas seguintes situações:

- um neurônio pode não ter axônios, mas apenas *processos* que servem tanto para receber como para transmitir informação (figura A.1B).
- axônios podem formar sinapses em outros axônios (figura A.1C);
- dendritos podem formar sinapses em outros dendritos (figura A.1D);

¹³ As junções entre as células nervosas são chamadas de sinapses. São os locais através dos quais as células transferem sinais.

¹⁴ O potencial de ação é um impulso numa fibra nervosa, que se move rapidamente ao longo do nervo.

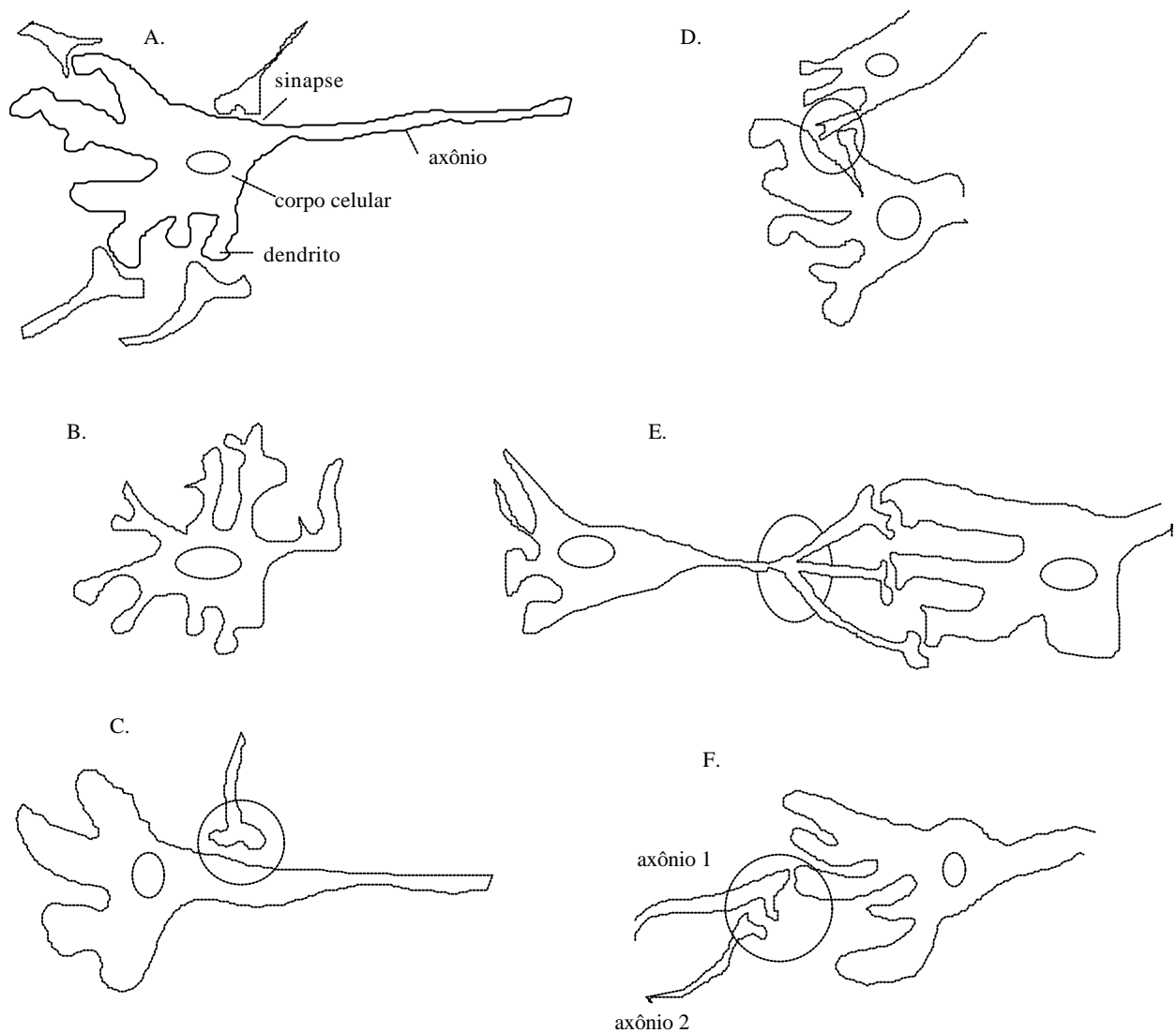


Figura A.1. Diagramas esquematizados do neurônio clássico (A) e algumas de suas variantes (B-F) (Crick e Asanuma, 1986).

- um axônio pode não propagar um *spike* mas produzir um potencial graduado (*graded potential*). Por causa da atenuação, deve-se esperar que esta forma de sinalização da informação não ocorra através de distâncias longas (figura A.1E). Estes potenciais graduados podem ocorrer em outro nível. Por exemplo, um terminal de axônio formando uma sinapse em uma dada célula pode receber uma outra sinapse (figura A.1F). A sinapse pré-sináptica pode exercer apenas uma mudança de potencial local que é portanto restrito àquele terminal de axônio.

A.2.2 Sinapses: Junções entre Células Nervosas.

O tipo predominante de sinapse no cérebro do mamífero é a sinapse química, que opera através de liberação de uma substância transmissora do terminal pré-sináptico para o terminal pós-sináptico (figura A.2).

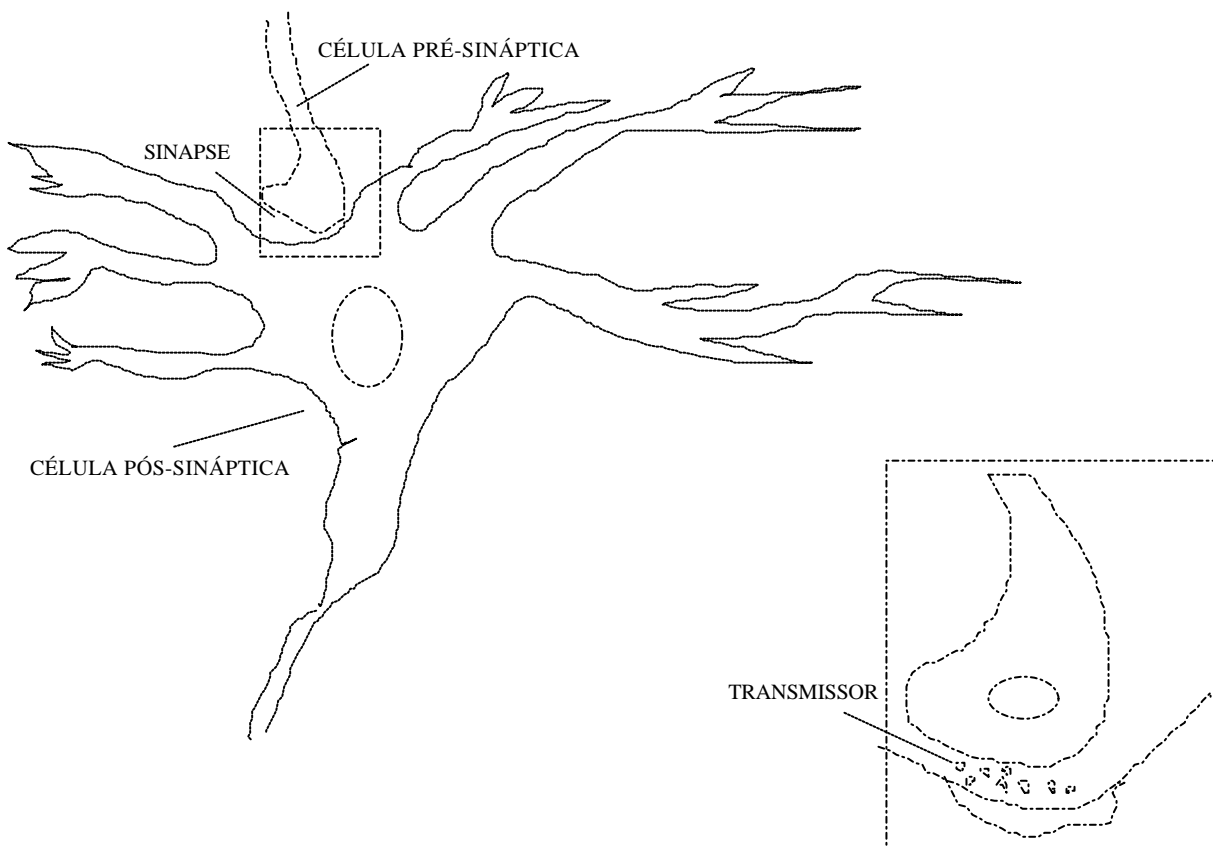


Figura A.2. Na maior parte das sinapses, o terminal pré-sináptico libera uma substância química, o transmissor, em resposta a uma despolarização. (Kuffler *et al.*, 1984).

A acetilcolina (um neurotransmissor) é difundida por uma distância curta até a membrana pós-sináptica e age nas moléculas receptoras de acetilcolina específicas naquela membrana. Então, a acetilcolina é enzimaticamente dividida e parte dela é levada novamente à síntese de um novo transmissor.

As vesículas usadas fundem-se com a membrana do terminal pré-sináptico e novas vesículas são formadas da membrana nas margens do terminal.

A.2.2.1 As Sinapses são Químicas e não Elétricas

Como já foi mencionado, a maior parte das sinapses que ocorrem no córtex cerebral são químicas e não elétricas. Os contatos sinápticos podem ser classificados morfológicamente em dois tipos básicos (Crick e Asanuma, 1986; Kandel *et al.*, 1995):

- tipo I (figura A.3A): estas sinapses têm especializações de membrana assimétricas (a espessura da membrana é maior no lado pós-sináptico) e o processo pré-sináptico contém vesículas sinápticas redondas bastante grandes (50 nm), onde acredita-se que existam pacotes de neurotransmissores.
- tipo II (figura A.3B): estas têm especializações de membrana simétricas. As vesículas sinápticas são menores e com os fixativos usuais usados pela microscopia eletrônica, são frequentemente elipsoidais ou achatados. (A forma das vesículas depende dos detalhes de fixação e não é sempre um critério completamente confiável quando compara-se resultados relatados por diferentes pessoas.) A zona de contato é usualmente menor que da sinapse tipo I.

A.2.2.2 As Sinapses Podem Excitar ou Inibir

A importância da classificação nos dois tipos morfológicos é que as sinapses do tipo I parecem ser excitatórias, ao passo que as sinapses do tipo II parecem ser inibitórias¹⁵.

¹⁵ As células nervosas influenciam outras por (a) excitação, ou seja, elas produzem impulsos em outras células e (b) inibição, ou seja, elas previnem a liberação de impulsos em outras células.

Existe um outro critério possível para determinar o caráter das sinapses: o transmissor que elas usam. Em geral, assume-se que um dado transmissor fará usualmente a mesma coisa em lugares diferentes, apesar de haver exceções, dependendo da natureza dos receptores pós-sinápticos.

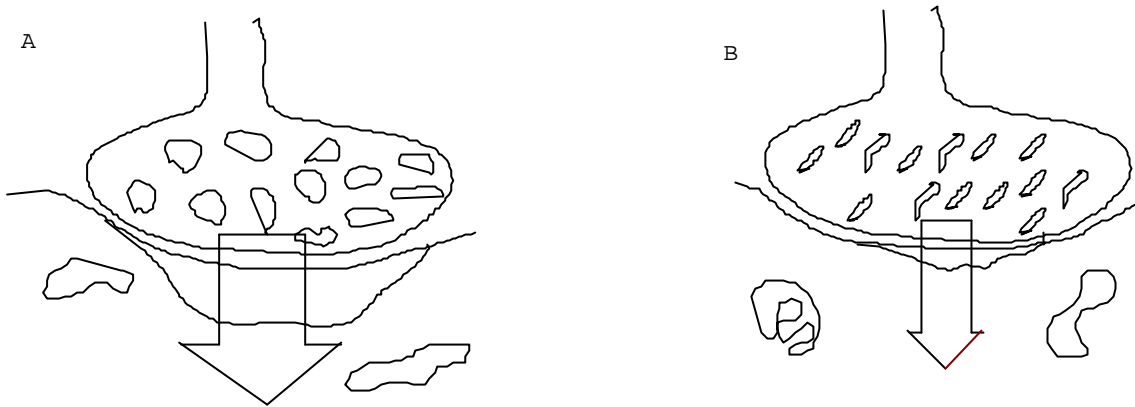


Figura A.3. Diagramas idealizados das sinapses tipo I (A) e tipo II (B). Veja o texto para esclarecimentos (Crick e Asanuma, 1986).

A.2.2.3 Generalizações sobre Sinapses

Vários métodos têm sido usados para identificar os neurotransmissores, mas cada técnica tem limitações. No momento, é difícil identificar os transmissores envolvidos e seus efeitos pós-sinápticos em muitas sinapses do sistema nervoso central. Pode-se fazer uma lista de tentativas de possíveis generalizações sobre sinapses:

- nenhum axônio faz sinapses tipo I em alguns locais enquanto faz tipo II em outros;
- nenhum axônio no cérebro de mamífero mostrou liberação de dois neurotransmissores diferentes não peptídeos. (Mas parece que muitos neurônios, incluindo neurônios corticais, podem liberar um transmissor “convencional” e um neuropeptídeo, ou em alguns casos, dois ou mais neuropeptídeos);

- não existe evidência no cérebro de mamífero que um mesmo axônio possa causar excitação e inibição em sinapses diferentes, mas isto é certamente possível já que o efeito de um dado transmissor depende dos tipos dos receptores presentes e de seus canais de íon associados.

A.2.2.4 *Peptídeos: Moduladores da Função Sináptica*

Ao longo dos últimos dez anos tem-se descoberto que existem muitos peptídeos distintos, de vários tipos e tamanhos, que podem agir como neurotransmissores. Há, no entanto, razões para suspeitar que os peptídeos são diferentes de muitos transmissores convencionais:

- peptídeos aparecem para *modular* a função sináptica ao invés de ativá-la;
- a ação de peptídeos, em poucos casos estudados, geralmente consiste em avançar vagarosamente e persistir por algum tempo, isto é, por segundos ou mesmo minutos, ao passo que os transmissores convencionais duram poucos milisegundos;
- em alguns casos foi mostrado que os peptídeos não agem onde foram liberados, mas a alguma distância. A difusão leva tempo. O tempo demorado de persistência seria compatível com os possíveis atrasos de tempo produzidos pela difusão;
- existem muitos exemplos agora conhecidos de um neurônio único produzindo, e presumivelmente liberando, mais de um neuropeptídeo.

A.2.2.5 *Peptídeo: Transmissor Lento ou Neuromodulador ?*

Foi mostrado que os peptídeos formam um segundo, mais lento, meio de comunicação entre neurônios, mais econômico do que usar neurônios extras para este propósito.

Os peptídeos têm papel de modulação principalmente em sistemas neurais, nos quais o modo de comunicação é o endereçamento químico.

Como transmissores os peptídeos agem em locais bem restritos, mesmo assim como um meio de condução muito lento, não sustentando as altas frequências dos impulsos. Como neuromoduladores da função sináptica, a sua atividade é mais intensa. Os efeitos excitatórios da substância P (um peptídeo) são muito lentos no início e prolongados na duração (mais de um minuto) e por si só não podem causar a despolarização¹⁶ suficiente para excitar as células. O efeito, entretanto, é tornar os neurônios mais prontamente excitáveis por outras entradas excitatórias – um claro exemplo de *neuromodulação*.

A.3 O Cérebro como Modelo

A idéia de simular o cérebro já era o objetivo de muitos trabalhos iniciais em Inteligência Artificial. O cérebro era visto como uma *rede neural*, ou seja, um conjunto de nós, ou neurônios, conectados por linhas de comunicação. Atualmente tem havido um crescente interesse no uso de modelos de redes neurais ou connexionistas. Modelos connexionistas são aplicáveis a vários problemas de ciência cognitiva, incluindo processamento de linguagem natural, processamento de fala e visão.

Num nível mais simples, pode-se conceber que o cérebro funciona da seguinte forma: neurônios ativam ou inibem o disparo de outros neurônios. Se um determinado neurônio dispara ou não depende das entradas inibitórias ou excitatórias de todos os neurônios conectados a ele.

A.3.1 Paralelismo

¹⁶ Despolarização é uma redução do potencial da membrana celular para zero mV, sendo que o interior do neurônio torna-se mais positivo. A despolarização para um nível de potencial crítico, o limiar, causa o início de um impulso. No seu pico, o interior da célula torna-se positivo em relação ao seu exterior. Na maioria das sinapses, o terminal pré-sináptico libera uma substância química, o transmissor, em resposta a uma despolarização. Numa sinapse excitatória, o transmissor liberado pelo terminal pré-sináptico despolariza a célula pós-sináptica, fazendo com que o potencial de sua membrana atinja o limiar. Numa sinapse inibitória, o transmissor tende a manter o potencial da membrana da célula pós-sináptica abaixo do limiar.

Uma outra razão para se estudar modelos parecidos com o cérebro é seu paralelismo. Os *circuitos* do cérebro são mais lentos do que os de um computador. Para que o cérebro trabalhe o mais rápido possível – os psicólogos mostraram que podemos reconhecer objetos num segundo ou menos – muitos neurônios devem trabalhar em paralelo. Em contraste, muitos programas de Inteligência Artificial conexionistas rodam muito lentamente, pois são simulados em um sistema uniprocessador, isto é, os cálculos relativos a cada “neurônio artificial” devem ser feitos um a cada vez.

A computação paralela tem sido bastante explorada em ciência da computação nos últimos anos. As redes neurais representam apenas uma linha de pesquisa em computação paralela. Basicamente, deve-se responder duas questões fundamentais no projeto de um sistema de computador paralelo: como conectar os processadores para propósito de comunicação e quanto de potência computacional e memória cada processador deve ter.

Os pesquisadores de redes neurais acreditam que seus modelos, por serem os mais fiéis sobre o cérebro conhecido, terão sucesso. Infelizmente, as redes neurais raramente têm sido construídas em hardware; normalmente elas são simuladas por software. Como já foi dito, estas simulações são geralmente muito lentas, pois um processador tem que fazer o trabalho de muitos. Até que se construa hardware de processamento paralelo efetivo, os modelos conexionistas alcançarão soluções não muito eficientes para problemas de Inteligência Artificial.

A.3.2 Variedades de Redes Neurais

Muitos modelos de redes neurais devem alguma coisa aos perceptrons (veja item A.4.1.1), mas são mais gerais. O modelo típico de rede neural consiste de um conjunto de nós, ou neurônios, e conexões. Cada nó tem a sua ativação, que geralmente é um número binário (1 significa presença do impulso de entrada e 0 a ausência). Cada conexão contém um número real, seu peso. Algumas unidades são conectadas à entrada e saída. Os pesos representam a força de conexão entre dois neurônios (força sináptica).

Geralmente, a rede neural é um sistema dinâmico, movendo de um estado para o próximo. Como tal, ela tem uma regra matemática que rege esse movimento. Um número muito grande de tais regras é possível. Entretanto, usualmente quer-se limitar os modelos a influenciar a ativação de um dado nó baseado apenas nas ativações dos nós conectados a ele e nos pesos das conexões a esses nós. Muitas críticas ao modelo conexionista vêm do fato de que esta abordagem é pobre biologicamente. Para viabilizar a implementação computacional, simplifica-se o modelo. Mas modelos alternativos enfatizando as características do neurônio biológico estão sendo estudados (Rocha, 1992; Rosa e França, 1998).

As redes neurais não são explicitamente programadas como um computador convencional. Por melhor dizer, elas obedecem leis, ou regras, como um sistema físico. Deve-se programar um computador convencional, mas uma rede neural simplesmente se conduz. Os projetistas de redes neurais vêem isto como uma vantagem, pois isto provê um mecanismo por meio do qual a inteligência pode surgir da lei física.

Uma das mais simples dessas regras é a regra linear. Computa-se a ativação de um dado nó como a soma dos produtos do peso de cada nó ao qual está conectado e a força dessa conexão. Essa regra é freqüentemente limitada: valores que passam de um certo limiar são cortados, para evitar os valores de ativação grandes. Existem muitas variantes das regras lineares.

Uma outra regra, sugerida por D. O. Hebb (1949), reforça a conexão entre dois nós que são altamente ativados ao mesmo tempo. Este tipo de regra é uma formalização da psicologia associacionista, que assegura que associações são acumuladas entre coisas que ocorrem juntas.

A.3.3 Aprendizado Competitivo

O aprendizado é, talvez, o fenômeno mais importante em psicologia. Os primeiros pesquisadores em redes neurais eram ansiosos para mostrar como as redes podiam aprender padrões de entrada apresentados a elas – ou seja, como elas podiam vir a perceber esses padrões, por elas mesmas.

Um dos métodos que vários pesquisadores têm planejado através dos anos é o aprendizado competitivo. Este método tem um primeiro nível, de unidades de entrada que contêm o padrão a ser inserido no sistema. O nível acima das unidades de entrada consiste de **clusters** de unidades. Cada unidade num cluster compete com as outras unidades no cluster pelo direito de reconhecer um padrão de entrada. Depois de um período de aprendizado, cada unidade num cluster reconhece um subconjunto dos padrões apresentados a ela. Portanto, cada cluster representa uma classificação, ou grupo, de padrões de entrada.

No aprendizado competitivo, cada unidade em cada cluster é conectado a todas as unidades de entrada. Os pesos das conexões são inicialmente colocados em valores aleatórios. Os pesos aleatórios fazem com que certas unidades nos clusters comecem a responder mais a determinados padrões de entrada, pois os pesos das conexões a essas unidades de entrada são mais fortes para alguns do que para outros.

No decorrer do aprendizado, os pesos mudam (via regra de aprendizado). Como determinadas unidades no cluster se tornam sensíveis a determinadas unidades no padrão de entrada, os pesos conectando os pares associados de unidades aumentam, à custa de pares não associados de unidades. Unidades diferentes no mesmo cluster se inibem, de tal forma que apenas uma unidade num cluster “ganha” o direito de reconhecer um dado padrão.

Assim, com o tempo, unidades diferentes num cluster “reconhecem” propriedades diferentes de padrões de entrada. Por exemplo, um cluster de duas unidades pode separar todos os padrões de entrada naqueles que têm a maioria das suas unidades altamente ativadas e aqueles que estão na maioria desligadas. Os clusters maiores fariam mais classificações discriminatórias.

A.3.4 Representações Distribuídas

Uma importante característica de muitos modelos de redes neurais é sua natureza distribuída. Uma rede semântica padrão, como aquelas usadas nos primeiros esquemas de representação do conhecimento, consiste de um conjunto de nós conectados de alguma forma. Cada nó representa uma única palavra ou conceito. Se a rede estiver “pensando” na palavra *gato*, o nó para *gato* é ativado, e todos os outros nós não. Esta é uma representação local.

Em contraste, numa rede distribuída, os nós não têm um único significado; ou seja, um conceito individual é representado por um padrão por todos os nós. Por exemplo (Zeidenberg, 1987), se há dez nós, ativando-se os nós 1, 3, 4 e 7, pode-se representar o conceito *gorila* enquanto que ativando-se os nós 2, 4, 5 e 7, pode-se representar o conceito próximo *chimpanzé*. Conceitos que são próximos têm representações similares.

Uma rede de processamento paralelo distribuído, uma rede neural que usa representação distribuída, oferece a vantagem de generalização automática. Se se quer representar o conceito “gorilas são cabeludos”, reforça-se a conexão entre todos os nós que compõem o conceito *gorila* e todos os nós que compõem o conceito *cabeludo*. Como resultado, desde que a maioria dos nós em *gorila* são também usados em *chimpanzé*, uma associação é também feita entre *chimpanzé* e *cabeludo*. É assim que a generalização automática trabalha. Numa representação local, onde *gorila* e *chimpanzé* são representados por nós separados, uma conexão entre *gorila* e *cabeludo* não implicaria numa conexão entre *chimpanzé* e *cabeludo*.

Uma outra vantagem de uma representação distribuída é sua insensibilidade a danos. Numa representação local, se o sistema perde o nó que representa *avó*, ele perde seu conceito de avó.

Em uma representação distribuída, para perder um conceito, deve-se perder todos os nós que o representam. Se se perde apenas um ou dois nós, o conceito pode se degradar, mas ainda está lá. Isto é mais próximo ao tipo de memória perdida observada em adultos idosos.

A.3.5 Máquinas de Boltzmann

Uma importante classe de redes neurais simulam o comportamento de sistemas físicos. Os sistemas físicos têm uma tendência a se moverem para estados de energia potencial mínima. Um exemplo simples disto é uma bola rolando num vale entre duas colinas. No alto da colina, a energia potencial é alta; no vale, é baixa.

Este processo é chamado de relaxação. John Hopfield (1982) mostrou que uma certa regra evolucionária simples para uma rede neural levará à relaxação. Sistemas como os de Hopfield, que remontam aos sistemas termodinâmicos, são chamados de máquinas de Boltzmann. As máquinas de Boltzmann são muito usadas em várias aplicações de redes neurais.

A.3.6 Processamento de Sentenças

Um importante aspecto do entendimento de sentença envolve determinar os vários casos que as partes diferentes de uma sentença têm. Por exemplo, considere as seguintes sentenças isoladas de um contexto:

O macaco morreu.

O macaco quebrou.

Na primeira sentença, *macaco* é um animal, pois *morrer* é uma característica dos seres vivos; na segunda, *macaco* é uma ferramenta de trocar pneus, pois um animal não pode “quebrar”. De alguma forma, o modelo deve discernir seus casos diferentes.

McClelland e Kawamoto (1986) desenvolveram um sistema conexionista para fazer esta atribuição de casos. Palavras são descritas por *microcaracterísticas semânticas* – dimensões básicas que

descrevem muitos objetos e ações. Por exemplo, duas das microcaracterísticas que descrevem substantivos são “humano” e “leveza”, que têm os valores “humano, não-humano”, e “leve, pesado”, respectivamente. As palavras não são representadas diretamente nas redes do sistema, mas em termos das ativações de unidades representando microcaracterísticas.

Versões deste sistema para a língua portuguesa foram desenvolvidas por Rosa (1993), Rosa e Netto (1994) e Rosa (1997). O modelo tem um grupo de unidades para cada um dos casos principais que substantivos diferentes podem ter em uma ação. Estes casos são Agente, Paciente, Instrumento e Modificador. Por exemplo, a sentença “O homem comeu o sanduíche”, ativaria as microcaracterísticas de “comeu” e “homem” no conjunto das unidades que correspondem ao Agente; isto representa o fato de que o Agente para o verbo “comeu” é “homem”.

O sistema é treinado em uma série de sentenças. As atribuições do caso correto para as sentenças de treinamento são mostradas ao sistema. Estas atribuições correspondem às ativações de nós particulares. O sistema ajusta as conexões entre esses nós de tal forma que eles se reforcem mutuamente.

Depois de ser treinado com um número suficiente de sentenças, o sistema pode fazer atribuições de caso correto para novas sentenças. Ele ainda pode fazer atribuições de caso correto para sentenças com alguma ambigüidade sintática. Por exemplo, na sentença “O homem abateu o garoto com a maleta”, o sistema considera que “maleta” é o Instrumento de “abateu” ao invés de pertencer ao “garoto”, desde que “maleta” tenha microcaracterística que indique que ela é um instrumento.

O sistema também manipula bem vários outros problemas, e geralmente faz um bom trabalho em atribuição de casos.

A.3.7 O Futuro

As redes neurais são adequadas para várias tarefas de processamento de linguagem natural, incluindo reconhecimento de letra, leitura, e entendimento de sentença. Elas também são úteis em recu-

perar itens da memória. Elas não são milagrosas, mas trazem uma direção para a Inteligência Artificial e Psicologia Cognitiva, forte e biologicamente plausível, para muitos problemas importantes.

Eventualmente, um modelo conexionista do processo de entendimento de linguagem natural será provavelmente construído, desde que envolva conhecimento integrado de muitos domínios, incluindo fonética, morfologia, sintaxe e semântica. Modelos conexionistas são particularmente adequados à integração desses tipos de conhecimento.

A.4 Algoritmos Conexionistas

Uma vez identificado o problema que se queira solucionar através da abordagem conexionista, deve-se construir a rede neural. Ou seja, montar a arquitetura da rede: para uma rede de três camadas, quantos neurônios deve-se ter na entrada da rede (que corresponde, normalmente, ao número de bits que representa o padrão), quantos deve-se ter na saída (que corresponde, normalmente, à quantidade de bits do padrão de saída) e, o mais difícil, o número de neurônios na camada escondida. Os neurônios da camada escondida normalmente não são “calculados” (apesar de haver alguns algoritmos para isso) e seu número deve ser descoberto por tentativa e erro.

Depois de construída a rede neural artificial, deve-se escolher um *algoritmo conexionista* para “treinar” a rede (fase de aprendizado). O treinamento da rede normalmente é demorado, pois requer muitos “ciclos”, ou seja, deve-se mostrar várias vezes à rede, tudo que se deseja que ela aprenda. Depois do treinamento, a rede neural deve ser capaz de, numa única propagação (único ciclo) reconhecer o padrão com o qual ela foi treinada (fase de reconhecimento).

Os algoritmos de redes neurais em geral se dividem em dois tipos básicos: os algoritmos *supervisionados*, ou seja, quando a saída desejada da rede durante o treinamento é fornecida para comparação, e os *não-supervisionados*, quando a rede se conduz por si só, ou seja, não há um supervisor que verifique as suas saídas.

Entre os algoritmos supervisionados mais conhecidos está o algoritmo *backpropagation* (Rumelhart *et al.*, 1986a). Neste algoritmo, a cada ciclo, o padrão de entrada é propagado pela rede e na saída ele é comparado com a saída desejada (supervisor). Caso haja erros, os mesmos são corrigidos gradativamente, através das mudanças de pesos dos neurônios que se conectam ao valor de ativação errado (retropropagação dos erros).

Entre os algoritmos não-supervisionados mais representativos está o *competitive learning*, ou aprendizado competitivo (Grossberg, 1987), já discutido anteriormente.

A.4.1 Redes Perceptron Multicamadas

A.4.1.1 O Perceptron

O primeiro modelo matemático do neurônio foi o modelo proposto por McCulloch e Pitts (1943). Mais tarde, Rosenblatt (1957) criou o modelo do *perceptron*. Um *perceptron* modela um neurônio tomando uma soma ponderada de suas entradas e enviando a saída 1 se esta soma é maior que um determinado limiar (senão, envia 0). Veja a figura A.4.

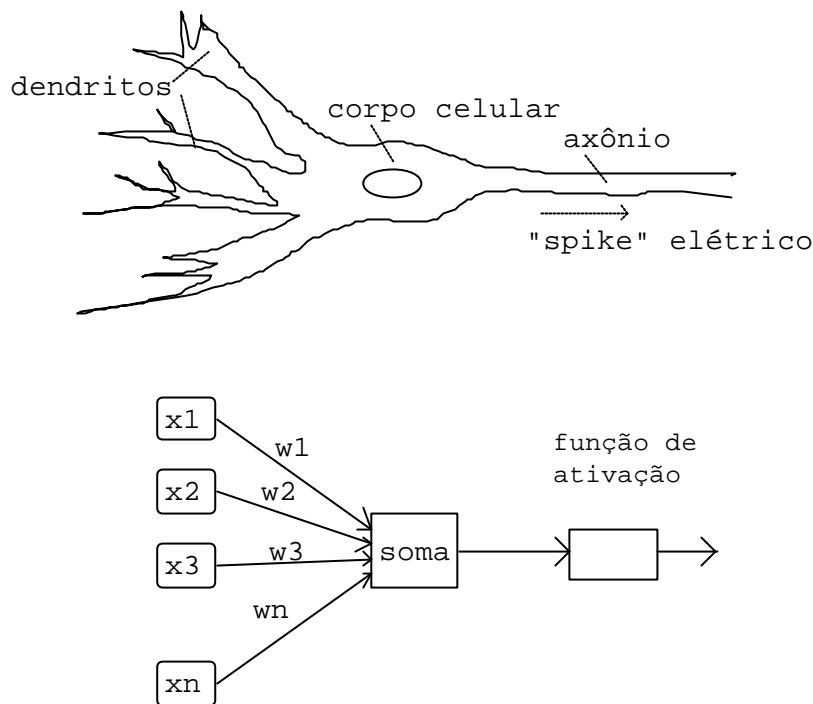


Figura A.4. Um neurônio e um perceptron (Rich e Knight, 1994).

A habilidade para treinar redes com várias camadas é um passo importante na direção da construção de máquinas inteligentes a partir de componentes inspirados nos neurônios. A meta é agrupar uma massa de elementos processadores, que simulam a célula nervosa, e ensiná-la a realizar tarefas úteis. É desejável que ela seja rápida e resistente a danos. É desejável que generalize a partir das entradas que vê.

O que uma rede multicamadas pode calcular? A resposta é: qualquer coisa. Dado um conjunto de entradas, pode-se usar unidades de limiar como simples portas lógicas *and* (conjunção), *or* (disjunção) e *not* (complemento), arranjando apropriadamente o limiar e os pesos de conexão. Sabe-se que é possível construir qualquer circuito combinatório a partir destas unidades lógicas básicas.

O maior problema é o aprendizado. O sistema de representação de conhecimento empregado pelas redes neurais é um tanto obscuro: as redes devem aprender suas próprias representações porque

programá-las é impossível. Uma propriedade das redes neurais diz que tudo que elas podem calcular, elas podem aprender a calcular.

É útil tratar primeiro com uma subclasse de redes multicamadas, chamadas de redes *totalmente conectadas*, divididas em *camadas* e alimentadas *para frente*. Um exemplo de tal rede é mostrada na figura A.5. Nesta figura x_i , h_i e o_i representam os níveis de unidade de ativação das unidades de entrada, escondida e saída, respectivamente. Os pesos das conexões entre as camadas de entrada e escondida são denotados por $w1_{ij}$, enquanto que os pesos das conexões entre as camadas escondida e de saída são denotados por $w2_{ij}$. Esta rede tem três camadas, ainda que seja possível, e algumas vezes útil, ter mais de três. Cada unidade numa camada é conectada a toda unidade da próxima camada na direção para frente, ou seja, cada unidade da camada de entrada é conectada a todas as unidades da camada escondida, nesta direção. As ativações fluem a partir da camada de entrada através da camada escondida, para a camada de saída. O conhecimento da rede é codificado nos pesos das conexões entre as unidades. Os níveis de ativação das unidades da camada de saída determinam a saída da rede.

A existência da camada escondida permite que a rede desenvolva representações internas. O comportamento destas unidades escondidas é automaticamente aprendido, não é pré-programado.

A propriedade mais importante dos sistemas conexionistas é que a rede neural não aprende apenas a classificar as entradas nas quais ela é treinada, mas também a *generalizar* e ser capaz de classificar entradas nunca vistas.

Tudo que as redes neurais parecem capazes de fazer é classificar. Os graves problemas da Inteligência Artificial, como planejamento, análise de linguagem natural e prova de teorema, não são simplesmente tarefas de classificação, então como as redes neurais resolvem estes problemas? Resolver os problemas de classificação são, no presente, o que as redes neurais fazem melhor. Mas pesquisa-se a aplicação a outros problemas, como por exemplo, processamento de linguagem natural (o sistema HTRP desta Tese é um exemplo).

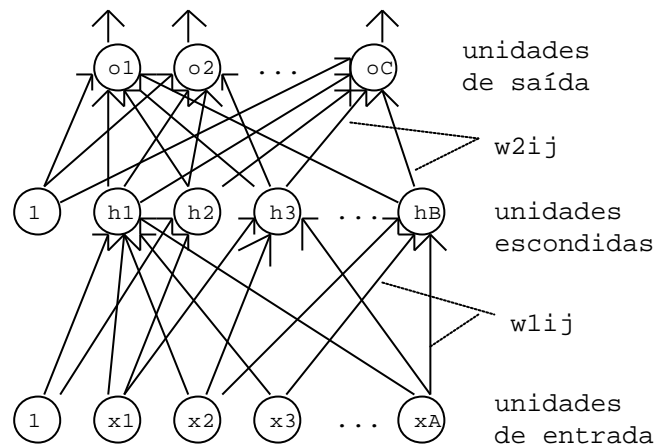


Figura A.5. Uma rede multicamada (Rich e Knight, 1994).

A.4.1.2 O Algoritmo Backpropagation e a Rede Perceptron Multicamadas

Por razões de simplificação, vai-se chamar a rede perceptron multicamadas de rede backpropagation. A unidade na rede *backpropagation* requer uma função de ativação baseada numa função denominada sigmóide (ou forma de *S*) que é contínua e diferenciável. Uma unidade soma suas entradas ponderadas e produz como saída um valor real entre 0 e 1. Seja *soma* a soma ponderada das entradas de uma unidade. A equação para a saída da unidade é dada por:

$$\text{saída} = 1 / (1 + e^{-\text{soma}})$$

Uma rede *backpropagation* tipicamente inicia com um conjunto de pesos aleatórios. A rede ajusta seus pesos cada vez que ela recebe um par entrada-saída. Cada par requer dois estágios: um passo para frente e um passo para trás. O passo para frente envolve a apresentação de uma amostra de entrada à rede e as ativações propagam-se até alcançarem a camada de saída. Durante o passo para trás, a saída real da rede (do passo para frente) é comparada com a saída desejada e as estimativas de erro são calculadas para as unidades de saída. Os pesos conectados às unidades de saída podem ser

ajustados a fim de reduzir estes erros. Pode-se usar as estimativas de erro das unidades de saída para derivar as estimativas de erro para as unidades das camadas escondidas. Finalmente, os erros são propagados de volta às conexões que tiveram origem nas unidades de entrada.

O algoritmo backpropagation geralmente atualiza seus pesos depois de ver cada par entrada-saída. Depois de vistos todos os pares entrada-saída (e ajustados seus pesos muitas vezes), diz-se que uma *época* completou-se. O treinamento de rede *backpropagation* usualmente requer muitas épocas.

A.4.1.3 O Algoritmo Backpropagation

O algoritmo seguinte é baseado na estrutura básica da figura A.5 (Rich e Knight, 1994).

Algoritmo *Backpropagation*

Dado: Um conjunto de pares de vetores de entrada-saída.

Calcular: Um conjunto de pesos para uma rede de três camadas que mapeia entradas nas saídas correspondentes.

1. Seja A o número de unidades na camada de entrada, como determinado pelo comprimento dos vetores de treinamento de entrada. Seja C o número de unidades na camada de saída. Agora escolher B , o número de unidades na camada escondida. Como mostrado na figura A.5, as camadas de entrada e escondida têm uma unidade extra usada para limiar; portanto, as unidades nestas camadas serão indexadas pela faixa $(0, \dots, A)$ e $(0, \dots, B)$. Denota-se os níveis de ativação das unidades na camada de entrada por x_j , na camada escondida por h_j e na camada de saída por o_j . Os pesos conectando a camada de entrada à camada escondida são denotados por wl_{ij} , onde o índice i indexa as unidades de entrada e o índice j indexa as unidades escondidas.

Da mesma forma, os pesos conectando a camada escondida à camada de saída são denotados por $w2_{ij}$, com i indexando as unidades escondidas e j indexando as unidades de saída.

2. Iniciar os pesos da rede. A cada peso deve ser atribuído um valor aleatório entre -0.1 e 0.1.

$$w1_{ij} = \text{random}(-0.1, 0.1) \quad \text{para todo } i = 0, \dots, A, j = 1, \dots, B$$

$$w2_{ij} = \text{random}(-0.1, 0.1) \quad \text{para todo } i = 0, \dots, B, j = 1, \dots, C$$

3. Iniciar as ativações para as unidades de limiar. Os valores destas unidades nunca devem mudar.

$$x_0 = 1.0$$

$$h_0 = 1.0$$

4. Escolher um par entrada-saída. Suponha que o vetor de entrada seja x_i , e o vetor de saída desejada seja y_i . Atribuir níveis de ativação às unidades de entrada.

5. Propagar as ativações a partir das unidades na camada de entrada para as unidades na camada escondida usando a função de ativação sigmóide:

$$h_j = 1 / (1 + e^{-soma1}) \quad \text{para todo } j = 1, \dots, B$$

em que

$$soma1 = \sum_{i=0}^A w1_{ij} \cdot x_i$$

Note que i varia de 0 a A . $w1_{0j}$ é o peso do limiar para a unidade escondida j (sua propensão a *disparar*¹⁷, a despeito de suas entradas). x_0 é sempre 1.0.

¹⁷ Disparar é tornar-se igual a 1.0.

6. Propagar as ativações a partir das unidades na camada escondida para as unidades na camada de saída

$$o_j = 1 / (1 + e^{-soma2}) \quad \text{para todo } j = 1, \dots, C$$

em que

$$soma2 = \sum_{i=0}^B w2_{ij} \cdot h_i$$

Novamente, o peso de limiar $w2_{0j}$ para a unidade de saída j traz uma contribuição à soma ponderada. h_0 é sempre 1.0.

7. Calcular os erros¹⁸ das unidades na camada de saída, denotado por $\delta 2_j$. Os erros são baseados na saída real da rede (o_j) e na saída desejada (y_j).

$$\delta 2_j = o_j(1 - o_j)(y_j - o_j) \quad \text{para todo } j = 1, \dots, C$$

8. Calcular os erros das unidades na camada escondida, denotado por $\delta 1_j$.

$$\delta 1_j = h_j(1 - h_j) \cdot soma3 \quad \text{para todo } j = 1, \dots, B$$

em que

$$soma3 = \sum_{i=1}^C d2_i \cdot w2_{ji}$$

9. Ajustar os pesos entre a camada escondida e a camada de saída. A taxa de aprendizado é denotada por η . Um valor razoável para η é 0.35.

¹⁸ A fórmula do erro é relacionada à derivada da função de ativação (sigmóide). Trata-se do erro quadrático médio.

$$\Delta w_{2ij} = \eta \cdot \delta 2_j \cdot h_i \text{ para todo } i = 0, \dots, B, j = 1, \dots, C$$

10. Ajustar os pesos entre a camada de entrada e a camada escondida.

$$\Delta w_{1ij} = \eta \cdot \delta 1_j \cdot x_i \text{ para todo } i = 0, \dots, A, j = 1, \dots, B$$

11. Ir para o passo 4 e repetir. Quando todas os pares de entrada-saída estiverem sido apresentados à rede, uma época completou-se. Repetir os passos 4 a 10 para quantas épocas desejar.

O algoritmo pode ser generalizado para redes com mais de três camadas¹⁹. A velocidade do aprendizado pode ser aumentada alterando os passos de modificação de pesos 9 e 10, com a inclusão de um termo α . As fórmulas de atualização de pesos ficam:

$$\Delta w_{2ij}(t+1) = \eta \cdot \delta 2_j \cdot h_i + \alpha \Delta w_{2ij}(t)$$

$$\Delta w_{1ij}(t+1) = \eta \cdot \delta 1_j \cdot x_i + \alpha \Delta w_{1ij}(t)$$

onde h_i , x_i , $\delta 1_j$ e $\delta 2_j$ são medidos no tempo $t + 1$. $\Delta w_{ij}(t)$ é a mudança que o peso experimenta durante o passo para frente-para trás anterior. Se α é colocado em 0.9, a velocidade de aprendizado aumenta²⁰.

Como a função de ativação tem a forma sigmóide, com assíntotas tendendo a 0 e 1, pesos de valor muito elevados seriam necessários para as saídas reais da rede alcançarem 0.0 e 1.0, com uma pequena margem de erro, portanto, as saídas desejadas (os y_j 's dos passos 4 e 7 acima) são usual-

¹⁹ Uma rede com três camadas (uma única camada escondida) pode calcular qualquer função que uma rede com muitas camadas escondidas pode calcular. Entretanto, o aprendizado é, às vezes, mais rápido, com múltiplas camadas escondidas (Rich e Knight, 1994).

²⁰ Empiricamente, os melhores resultados acontecem quando α é zero para os primeiros passos de treinamento, aumentando seu valor gradativamente até 0.9 durante o treinamento, segundo Rich e Knight (1994).

mente dadas como 0.1 e 0.9. A sigmóide é útil para a rede *backpropagation*, pois a derivação da regra de atualização do peso requer que a função de ativação seja contínua e diferenciável.

A.4.1.4 Generalização

Se todas as entradas e saídas possíveis são mostradas a uma rede *backpropagation*, ela terá seus pesos adaptados para que haja minimização do erro quadrático médio entre a saída desejada e a propagada. Para muitos problemas de Inteligência Artificial, entretanto, é impossível fornecer todas as entradas possíveis. Para resolver este problema, a rede *backpropagation* é adequada ao mecanismo de generalização. Se se trabalha num domínio onde entradas similares são mapeadas em saídas similares, a rede *backpropagation* irá interpolar quando forem fornecidas entradas que a rede nunca viu antes.

A.4.2 Redes Recorrentes

Uma deficiência clara nos modelos de redes neurais comparados aos modelos simbólicos é a dificuldade que eles têm em tratar com tarefas temporais em Inteligência Artificial tais como planejamento e análise de linguagem natural. As redes recorrentes, ou redes com loops, são uma tentativa de corrigir esta situação.

Considere a tentativa de ensinar uma rede como arremessar uma bola de basquete à cesta (Rich e Knight, 1994). Pode-se apresentar à rede uma situação inicial à ser entrada (distância e altura da cesta, posição inicial dos músculos), mas necessita-se mais que um simples vetor de saída. Necessita-se de uma série de vetores de saída: primeiro mova os músculos desta forma, depois desta forma, etc. A *rede de Jordan* (Jordan, 1986) faz algo parecido com isto. É mostrada na figura A.6. As *unidades de plano* da rede permanecem constantes. Elas correspondem a uma instrução como “arremessar uma bola a cesta”. As *unidades de estado* codificam o estado corrente da rede. As *unidades de saída* simultaneamente dá comandos (por exemplo, movimento o braço x para a posição y) e atualiza as unida-

des de estado. A rede nunca se estabiliza, ou seja, nunca alcança um estado estável; ao invés disto, ela muda a cada passo de tempo.

As redes recorrentes podem ser treinadas com o algoritmo *backpropagation*. A cada passo, compara-se as ativações das unidades de saída com as ativações desejadas e os erros são propagados de volta através da rede. Quando o treinamento está completo, a rede ainda é capaz de realizar uma sequência de ações. Características de *backpropagation*, tal como a generalização automática, também ocorrem nas redes recorrentes. Entretanto, há a necessidade de algumas modificações. Primeiro, deseja-se que as unidades de estados mudem suavemente. A suavidade pode ser implementada como uma mudança na regra de atualização de peso; essencialmente, o *erro* de uma saída torna-se uma combinação do erro real e da magnitude da mudança nas unidades de estado. O reforço da restrição da suavidade torna-se muito importante no aprendizado rápido, já que ele remove muitas das opções de manipulação de peso disponíveis no *backpropagation*.

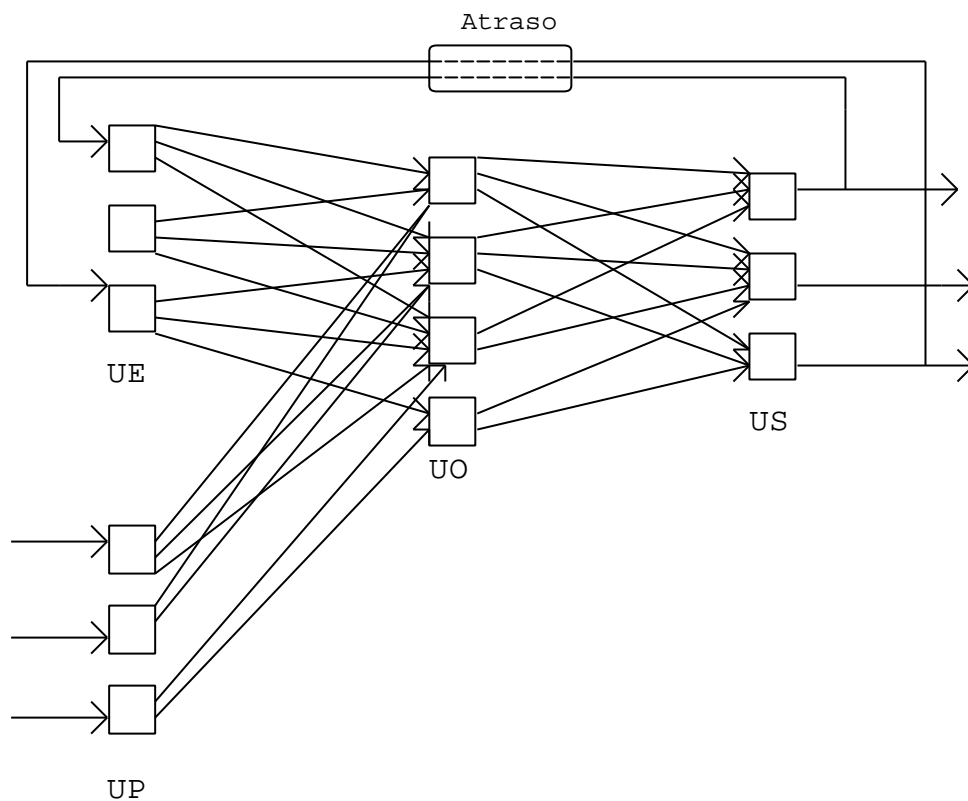


Figura A.6. Uma rede de Jordan, onde UE = unidades de estado, UP = unidades de plano, UO = unidades ocultas (escondidas) e US = unidades de saída.

Um problema maior nos sistemas de aprendizado supervisionado ocorre na correção do comportamento da rede. Se dados de treinamento suficientes podem ser coletados, então saídas alvo podem ser providas para muitos vetores de entrada. Entretanto, as redes recorrentes têm problemas de treinamento especiais, por causa da dificuldade de especificar completamente uma série de saídas alvo. No arremesso de bolas de basquete, por exemplo, a retroalimentação vem do mundo externo (isto é, onde a bola cai), não de um professor mostrando como mover cada músculo. Para contornar esta dificuldade, pode-se aprender um *modelo mental*, um mapeamento que relaciona as saídas da rede aos eventos no mundo. Com tal modelo, uma vez conhecido, o sistema proposto pode aprender tarefas sequenciais pela propagação de volta (*backpropagation*) dos erros que ele vê no mundo real. Então isto é necessário para aprender duas coisas diferentes: o relacionamento entre o plano e a saída da rede e entre a saída da rede e o mundo real.

As redes deste tipo são essencialmente iguais a da figura A.6, exceto pela adição de mais duas camadas: uma outra camada escondida e uma camada representando os resultados como visto no mundo. Primeiro, a última porção mencionada da rede é treinada (usando *backpropagation*) em vários pares de saídas e alvos até que a rede consiga saber como suas saídas afetam o mundo real. Depois disto os pesos brutos são estabelecidos, a rede inteira é treinada usando retroalimentação do mundo real até que ela seja capaz de funcionar bem.

Um outro tipo de rede recorrente é descrita por Elman (1990). Neste modelo, os níveis de ativação são explicitamente copiados das unidades escondidas para as unidades de estado. As redes deste tipo têm sido usadas em várias aplicações, incluindo análise de linguagem natural.

A.4.2.1 A Representação do Tempo

A questão de como representar o tempo em modelos conexionistas é muito importante. Uma abordagem é representar o tempo implicitamente pelos seus efeitos no processamento ao invés de explicitamente (como numa representação espacial).

O tempo é muito importante em cognição. Está intrinsecamente ligado a muitos comportamentos que se expressam como seqüências temporais (como na linguagem). A questão de como representar o tempo parece ser um problema unicamente dos modelos de processamento paralelo, mas mesmo em sistemas tradicionais (seriais) a representação da ordem serial e a interação de uma entrada serial ou saída com outros níveis de representação apresentam desafios. Os lingüistas normalmente não se preocupam com a representação dos aspectos temporais no processamento de discursos (assumindo, por exemplo, que todas as informações num discurso estão disponíveis simultaneamente numa árvore sintática); mas a pesquisa na análise de linguagem natural mostra que o problema não é trivial. Portanto, uma das características mais elementares da atividade humana – a extensão temporal – é algumas vezes ignorada e é freqüentemente problemática.

Nos modelos de processamento paralelo distribuído, o processamento de entradas sequenciais é completado de muitas formas. A solução mais comum é “paralelizar o tempo”, dando a ele uma representação espacial. Entretanto, existem problemas com esta abordagem e ela não é mais considerada uma boa solução. Uma abordagem mais interessante seria representar o tempo implicitamente, isto é, representa-se o tempo pelo efeito que ele tem no processamento e não como uma dimensão adicional da entrada (Elman, 1990). Isto significa dar ao sistema de processamento propriedades dinâmicas que são respostas às sequências temporais. Em resumo, à rede deve ser dada memória.

A abordagem de Elman (1990) pode ser modificada da seguinte forma. Suponha uma rede (mostrada na figura A.7) aumentada no nível de entrada por unidades adicionais chamadas de Unidades de Contexto. Estas unidades também estão “escondidas” no sentido em que elas interagem exclusivamente com outros nós internos da rede e não com o mundo externo.

Imagine que exista uma entrada sequencial a ser processada e algum relógio que controle a apresentação da entrada à rede. O processamento então consistiria da seguinte sequência de eventos. No tempo t , as unidades de entrada recebem a primeira entrada da sequência. Cada unidade deve ter um valor escalar simples ou um vetor, dependendo da natureza do problema. As unidades de contexto são inicialmente colocadas em 0.5²¹. As unidades de entrada e de contexto, ambas, ativam as unidades escondidas; as unidades escondidas, então, alimentam para frente para ativar as unidades de saída. As unidades de saída também retroalimentam para ativar as unidades de contexto. Isto constitui a ativação para frente. Dependendo da tarefa, pode existir ou não uma fase de aprendizado neste ciclo de tempo. Se existir, a saída é comparada à entrada mestre e a propagação para trás do erro é usada para ajustar os pesos de conexão. As conexões recorrentes são fixas em 1.0 e não são sujeitas ao ajuste²². No próximo passo de tempo, $t+1$, a sequência acima é repetida. Desta vez, as unidades de contexto contêm valores que são exatamente os valores das unidades de saída no tempo t , então estas unidades de con-

²¹ No caso da função de ativação usada ter valores entre 0.0 e 1.0.

²² Na maioria das redes usadas, existem conexões um-para-um entre cada unidade de saída e cada unidade de contexto. Isto implica que existe um número igual de unidades de contexto e unidades de saída. As conexões para cima entre as unidades de contexto e as unidades escondidas são totalmente distribuídas, de tal forma que cada unidade de contexto ativa todas as unidades escondidas.

texto provêm a rede de memória. Note que tanto a rede de Jordan como a de Elman contêm a camada adicional (unidades de estado ou de contexto). A diferença básica é que na rede de Elman, a realimentação para a camada de entrada se dá a partir da camada escondida e não da camada de saída, como em Jordan.

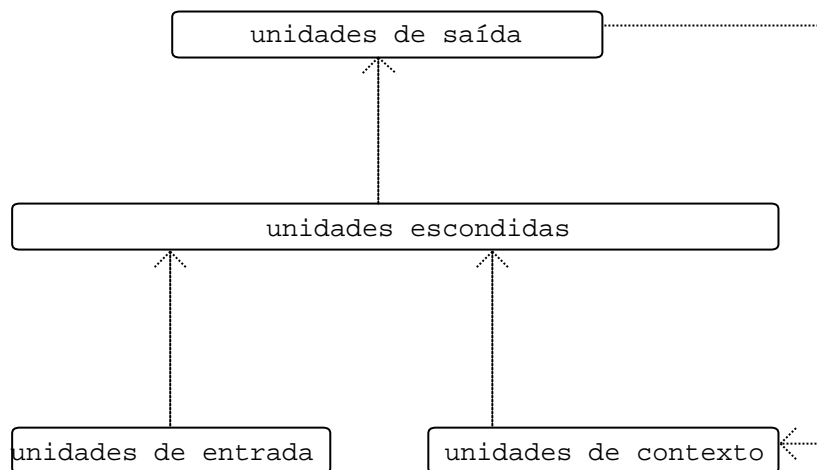


Figura A.7. Uma rede recorrente simples na qual as ativações são copiadas da camada de saída para a camada de contexto na base um-por-um, com pesos fixos em 1.0. As linhas pontilhadas representam conexões de treinamento (propagação do padrão de entrada).

A.4.2.2 Conclusões sobre Tarefas Temporais

Muitos comportamentos humanos desenvolvem-se através do tempo. Seria tolice tentar entender estes comportamentos sem levar em consideração sua natureza temporal. O conjunto corrente de simulações explora as consequências de desenvolver representações do tempo que são distribuídas, dependentes de tarefas e nas quais o tempo é representado implicitamente na dinâmica da rede.

A.5 Abordagem Híbrida: As Redes Neurais Baseadas em Conhecimento

A rede neural trabalha muito bem ao resolver certos tipos de problemas. A maior crítica que se faz aos sistemas conexionistas é o fato de que sabe-se que funciona mas não se sabe como. A representação interna dos pesos de uma rede neural é uma incógnita para os pesquisadores. Mas este tipo de crítica está sendo respondida gradativamente. Já há alguns anos começaram as pesquisas em Redes Neurais Baseadas em Conhecimento (RNBC), ou seja, redes neurais nas quais um conhecimento inicial simbólico é representado. A rede não começa mais com pesos sinápticos aleatórios e sim com um conjunto de pesos que refletem regras de produção. Na verdade trata-se de uma mistura das abordagens simbólica e conexionista (a chamada abordagem *híbrida*).

Numa rede baseada em conhecimento, dada uma gramática simbólica, cria-se a partir das regras desta gramática, uma arquitetura de rede neural. Ou seja, de uma regra $A \rightarrow B$, cria-se uma conexão entre um neurônio que representa o conceito A e outro neurônio que representa o conceito B. Para uma regra $(A \wedge B) \rightarrow C$, tem-se dois neurônios de entrada A e B, um neurônio na camada escondida faz o papel da conjunção e um neurônio na camada de saída representa o conceito C. Basta ligá-los através de pesos de conexão suficientemente grandes e tem-se uma rede que representa esta regra. A função da rede neural é revisar esta teoria. A teoria inicial (regras) está representada na rede. A rede começa a aprender. No final do aprendizado pode-se extrair de volta as regras da rede e, nesse caso, a teoria representada pelas regras foi revisada pelo aprendizado. Fu (1991a, 1991b e 1993), Towell e Shavlik (1993), Setiono e Liu (1996) e outros apresentam sistemas neurais baseados em conhecimento.

Numa rede neural baseada em conhecimento, através do conhecimento inicial simbólico baseado em regras de produção, constróem-se conexões fortes entre neurônios que representam estes conceitos. No restante da rede, atribuem-se pesos pequenos, mas não nulos, de tal forma que uma regra ainda inexistente possa eventualmente se estabelecer. Através do treinamento ao que a rede neural é exposta, vários padrões diferentes são apresentados à rede. Se um determinado padrão ocorre várias vezes, supondo que o treinamento é coerente com o que acontece no mundo, significa que deveria haver uma regra relacionando os conceitos envolvidos por este padrão. Isto é, a teoria simbólica inicial

deveria prever este tipo de comportamento. Se a teoria realmente já contava com este acontecimento, sua regra já foi implementada, e o treinamento se encarregará de fortalecer ainda mais os pesos sinápticos que relacionam estes conceitos. Caso contrário, é desejável que a teoria seja revista, ou seja, que esta “nova” regra seja adicionada ao conjunto de regras da teoria simbólica. Isto é conseguido através da extração de regras da rede neural.

Segundo Fu (1993), o procedimento completo é o seguinte: primeiro atribuem-se os pesos altos, relativos às regras da teoria inicial, à rede. Aos demais pesos são atribuídos valores pequenos. A seguir a rede passa pelo processo de treinamento, através do algoritmo backpropagation ou de um outro algoritmo conexionista qualquer. Depois, a rede sofre um processo de anulação de seus pesos pequenos, pois estima-se que pesos muito pequenos não vão contribuir para o conhecimento da rede. Com a anulação, a rede é simplificada. Uma outra simplificação ocorre quando a rede é “clusterizada”, ou seja, formam-se grupos de neurônios com vetores de pesos próximos entre si. Depois, novamente a rede é submetida a um treinamento, sendo que no final deste processo, ocorre a extração das regras. Fu propõe um algoritmo de extração de regras chamado KT, que trabalha com subconjuntos de pesos das entradas de um determinado neurônio, que, se alcançado o limiar de ativação, forma uma regra com o neurônio alvo representando o conceito de saída. O grande problema do algoritmo KT é a grande quantidade de regras geradas.

Já Towell e Shavlik (1993) propõem um outro algoritmo de extração de regras um pouco diferente do algoritmo KT, chamado MofN. Este algoritmo reduz bastante o número de regras obtidas. Basicamente, o algoritmo MofN consiste em seis passos:

1. *Agrupar*: criação de classes equivalentes, para agrupar as conexões da rede em *clusters*;
2. *Tirar a média*: depois de agrupados, faz com que os pesos de todas as conexões dentro de um *cluster* tenham o valor médio de todas as conexões deste *cluster*;
3. *Eliminar*: eliminação dos *clusters* com valores insignificantes, que não contribuem para o cálculo;
4. *Otimizar*: com os *clusters* sem importância eliminados no passo 3, otimiza-se os limiares da

unidade;

5. *Extrair*: formam-se regras que expressam a rede, tal que uma regra é verdadeira se a soma dos seus antecedentes ponderados exceder o limiar;
6. *Simplificar*: as regras são simplificadas quando possível para eliminar pesos e limiares.

A.6 Conclusão

O cérebro humano é o modelo natural para a construção de máquinas inteligentes. Uma idéia óbvia para a Inteligência Artificial é simular o cérebro em um computador (Rich e Knight, 1994). O aprendizado em representações de redes complexas é um dos tópicos mais atuais em ciência (Russell e Norvig, 1995), que promete grandes aplicações em ciência da computação, neurobiologia, psicologia e física. Apresentou-se nesse Anexo algumas das idéias e técnicas básicas das redes neurais artificiais. Uma rede neural é um modelo computacional que compartilha algumas das propriedades dos cérebros: consiste de muitas unidades simples trabalhando em paralelo sem nenhum controle central. As conexões entre as unidades têm pesos numéricos que podem ser modificados pelo aprendizado.