UNIVERSIDADE DE SÃO PAULO

Instituto de Física e Química de São Carlos

Departamento de Física e Ciências dos Materiais

GENERALIZAÇÃO ÓTIMA EM PERCEPTRONS

Osame Kinouchi Filho

Dissertação apresentada no Instituto de Física e Química de São Carlos, Universidade de São Paulo, para obtenção do título de Mestre em Física Básica.

Orientador: Prof. Dr. Nestor Felipe Caticha Alfonso

Serviço de Biblioteca e Informação - IFQSC
FÍSICA

São Carlos

1992

OK

**IFSC UNIVERSIDADE
DE SÃO PAULO**
Instituto de Física e Química de São Carlos

Fone (0162) 72-6222
Fax (0162) 72-2218

MEMBROS DA COMISSÃO JULGADORA DA DISSERTAÇÃO DE MESTRADO DE OSAMU KIMOUCHI FILHO APRESENTADA AO
INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS, DA UNIVERSIDADE DE SÃO PAULO, EM 03.04.1992

Av. Dr. Carlos Botelho, 1465
Caixa Postal 369
CEP 13560 - São Carlos - SP
Brasil

COMISSÃO JULGADORA:

Prof. Dr. Nestor Felipe Caticha Alfonso

Prof. Dr. José Fernando Pontanari

Prof. Dr. Walter Felipe Wreszinski

AGRADECIMENTOS

À meus pais, pelo carinho e compreensão, e pela ajuda financeira.

À Nice, pela paciência durante a fase final de elaboração da dissertação, pelo auxílio com a bibliografia, e simplesmente por existir.

Ao Prof. Nestor Caticha, pela orientação e convívio intelectual estimulante que possibilitaram a realização deste trabalho (e pela amizade, paciência, ajuda financeira, almoços, pernoites, caronas, discussões aos sábados e domingos, revisões de última hora etc...).

Ao Prof. Michel André Aegerter, pela orientação durante a graduação e parte do mestrado, e pelas boas experiências e oportunidades que me foram oferecidas em seu grupo.

Ao Prof. José Fernando Fontanari, por algumas sugestões valiosas.

À tia Mari, pelo carinho e hospedagem em São Paulo nestes últimos meses.

À meu irmão Marcelo, pelo auxílio com a impressora.

Ao Fernando, pela amizade e incentivo durante a graduação e o mestrado.

Ao Lucas, pela amizade e pelo auxílio durante a entrega da dissertação.

À Irene e Vlã, e também ao Ítalo e pessoal de apoio do IFQSC, pela paciência.

À FAPESP e CAPES, pelo apoio financeiro.

"Willard Gibbs called this science statistical mechanics. It falls into two parts. The first investigates the conditions under which the externally noticeable properties of a complex of a very large number of mechanical individuals do not change, in spite of vigorous motions of these individuals; this part I shall call statistical statics. The second part calculates gradual changes in these externally visible properties if those conditions are not fulfilled, let it be called statistical dynamics. The wide perspectives opening up if we think of applying this science to the statistics of living beings, human society, sociology and so on, instead of only to mechanical bodies, can here only be hinted at in a few words."

Ludwig Boltzmann

On Statistical Mechanics (1904)

Índice

<u>CAPÍTULO 1 – INTRODUÇÃO</u>	05
1.1 O PERCEPTRON E AS REDES NEURAIS DE ATRADORES	06
1.2 APRENDIZAGEM E GENERALIZAÇÃO NO PERCEPTRON	12
1.2.1 Mapeamento aleatório, Generalização e "Extração de Regra"	13
1.2.1 Aprendizagem de Funções Linearmente Separáveis	14
1.3 DINÂMICAS DE APRENDIZAGEM	15
1.4 SITUAÇÕES DE APRENDIZAGEM	17
1.4.1 Tipos de Perceptrons	18
1.4.2 Condições de "Caixa Negra", "Localidade" e de "Linearidade"	18
1.4.3 Controle Sobre o Conjunto de Aprendizagem	20
1.4.4 Aprendizagem com Apresentação Única dos Exemplos	23
1.5 REVISÃO DO PROBLEMA E PLANO DE TRABALHO	24
<u>CAPÍTULO 2 – APRENDIZAGEM E GENERALIZAÇÃO NO PERCEPTRON</u>	26
2.1 CURVAS DE APRENDIZAGEM E GENERALIZAÇÃO	26
2.2 RECORDAÇÃO SERIADA DE LISTAS	30
2.3 EXEMPLO: A REGRA DE HEBB	31
<u>CAPÍTULO 3 – PROBLEMA DE GENERALIZAÇÃO COM APRESENTAÇÃO ÚNICA DOS EXEMPLOS</u>	36
3.1 SOLUÇÃO DO PROBLEMA DE GENERALIZAÇÃO SEM RESTRIÇÕES	37

3.2	APRENDIZAGEM PASSIVA	41
3.2.1	Aprendizagem com Perceptron Professor Linear	42
3.2.2	Perceptron Professor Booleano: Algoritmo com Peso Simétrico	46
3.2.3	Solução de Ótima Generalização com Professor Booleano	49
3.3	APRENDIZAGEM ATIVA OU SELEÇÃO DE EXEMPLOS	53
3.3.1	Perceptron Professor Linear	53
3.3.2	Perceptron Professor Booleano	54
3.4	APRENDIZAGEM AUTO-REGULADA PELO ERRO DE GENERALIZAÇÃO	56
3.5	COMPARAÇÃO COM OUTROS ALGORÍTMOS	58
3.6	ANALOGIAS BIOLÓGICAS	64
 <u>CAPÍTULO 4 – PROBLEMA DE GENERALIZAÇÃO COM APRESENTAÇÃO</u>		
	<u>REPETIDA DOS EXEMPLOS</u>	68
4.1	APRENDIZAGEM COM PROFESSOR LINEAR	69
4.1.1	Tratamento Estatístico do Processo de Aprendizagem	69
4.1.2	Método de Réplicas	73
4.1.3	Solução de Réplicas Simétricas	77
4.2	APRENDIZAGEM COM PROFESSOR BOOLEANO	79
4.2.1	Comparação com o Algoritmo de Bayes	81
4.2.2	Aprendizagem Usando Aproximação de Bayes	85
4.2.3	Seleção de Exemplos	88
 <u>CAPÍTULO 5 – CONCLUSÕES E PERSPECTIVAS</u>		90
 <u>REFERÊNCIAS</u>		94
 <u>APÊNDICES</u>		98

"Neste ponto pode ser levantada a questão sobre onde as capacidades do perceptron realmente acabam (...) O sistema descrito é suficiente para reconhecimento de padrões, aprendizagem associativa e capacidades cognitivas como as que são necessárias para atenção seletiva e lembrança seletiva. O sistema parece ser potencialmente capaz de reconhecimento temporal de padrões (...), aprendizagem por tentativa e erro, emissão de sequências ordenadas de respostas..."

Frank Rosenblatt (1958)

"Princípio de Rosenblatt: quanto maior o perceptron, mais facilmente ele aprende; logo, um perceptron infinito não precisa aprender nada..."

Stanislaw Lem (1973)

"Acredito agora que o livro (Perceptrons) foi demasiadamente letal (...) assim, depois de ter ficado irritado com Rosenblatt por exagerar e desviar todas aquelas pessoas para um falso caminho, eu comecei a perceber que, para aquilo que você pode obter dele — o tipo de reconhecimento que ele pode fazer — (o perceptron) é uma máquina tão simples que seria surpreendente se a natureza não fizesse uso dele em algum lugar".

Marvin Minsky (1987)

ABSTRACT

The perceptron has been studied in the context of statistical physics since the seminal work of Gardner and Derrida on the coupling space of this simple neural network. Recently, Oppen and Haussler calculated, with the replica method, the theoretical optimal performance of the perceptron for learning a rule (generalization).

In this work we found the optimal performance curve after the first presentation of the examples (first step of learning dynamics). In the limit of large number of examples the generalization error is only two times the error found by Oppen and Haussler.

We also calculated the optimal performance for the first step in the learning situation with "selection of examples". We show that optimal selection occurs when the new example is chosen orthogonal to the perceptron coupling vector. The generalization error in this case decay exponentially with the number of examples.

We also propose a new class of learning algorithms which approximates very well the optimal performance curves. We study analytically the first step of the learning dynamics and numerically its behaviour for long times.

We show that several known learning algorithms (Hebb, Perceptron, Adaline, Relaxation) can be seen as more or less reliable approximations of our algorithm.

RESUMO

O perceptron tem sido estudado no contexto da física estatística desde o trabalho seminal de Gardner e Derrida sobre o espaço de acoplamentos desta rede neural simples. Recentemente, Oppen e Haussler calcularam, via método de réplicas, o desempenho ótimo teórico do perceptron na aprendizagem de uma regra a partir de exemplos (generalização).

Neste trabalho encontramos a curva de desempenho ótimo após a primeira apresentação dos exemplos (primeiro passo da dinâmica de aprendizagem). No limite de grande número de exemplos encontramos que o erro de generalização é apenas duas vezes maior que o erro encontrado por Oppen e Haussler.

Calculamos também o desempenho ótimo para o primeiro passo da dinâmica de aprendizagem com "seleção de exemplos". Mostramos que a seleção ótima ocorre quando o novo exemplo é escolhido ortogonal ao vetor de acoplamentos do perceptron. O erro de generalização neste caso decai exponencialmente com o número de exemplos.

Propomos também uma nova classe de algoritmos de aprendizagem que aproxima muito bem as curvas de desempenho ótimo. Estudamos analiticamente o primeiro passo da dinâmica de aprendizagem e numericamente seu comportamento para tempos longos.

Mostramos que vários algoritmos conhecidos (Hebb, Perceptron, Adaline, Relaxação) podem ser interpretados como aproximações, de maior ou menor qualidade, de nosso algoritmo.

CAPÍTULO 1

INTRODUÇÃO

Modelos de redes neurais possuem uma longa história e têm sido estudados por neurofisiólogos, psicólogos, engenheiros, matemáticos e físicos a partir de perspectivas, métodos e objetivos diferentes. Três perspectivas se destacam:

- a abordagem biológica, cujo objetivo é entender o funcionamento das redes neurais e eventualmente do cérebro em termos de sua estrutura;
- a abordagem tecnológica e de inteligência artificial (IA), que se interessa pelas capacidades computacionais das redes de neurônios artificiais e a construção de dispositivos práticos;
- a abordagem física, que vê as redes neurais como exemplos de sistemas complexos a serem estudados com as técnicas da física estatística.

A abordagem dos físicos possui um viés particular: os modelos em geral são uniformes e possuem grande número de unidades, o que permite um tratamento estatístico; tenta-se capturar propriedades universais (no sentido que esta palavra possui em física estatística) através de modelos extremamente simplificados; há grande interesse por resultados analíticos; são usadas analogias com outros sistemas físicos, particularmente sistemas magnéticos desordenados.

O presente trabalho possui estas características. Estamos interessados em resultados analíticos para o problema de generalização mais simples possível na mais simples das redes neurais: o perceptron. Estes resultados são obtidos no limite "termodinâmico" em que o número de unidades N é muito grande ($N \rightarrow \infty$).

A dissertação é organizada do seguinte modo:

- O capítulo 1 contém um breve resumo histórico introduzindo o perceptron e situando o problema da dinâmica de aprendizagem em redes neurais. Algumas definições gerais são dadas e o problema que nos interessa é delimitado com cuidado. Uma pequena revisão da literatura mostra o estado atual dos estudos em problemas de generalização para esta rede neural simples.

— No capítulo 2 introduzimos as curvas de aprendizagem e generalização e estendemos os resultados de Vallet [Vl 89a, Vl 90] relativos à regra de Hebb calculando a curva de reconhecimento de versões com ruído de exemplos aprendidos.

— O capítulo 3 contém nossa derivação de um novo tipo de algoritmo de aprendizagem (que denominamos "algoritmo de estabilidade esperada") que produz generalização ótima para o problema de apresentação única dos exemplos. Encontramos as curvas analíticas de generalização ótima nas situações de aprendizagem com e sem "seleção de exemplos". Apresentamos simulações e fazemos a comparação com outros algoritmos.

— O capítulo 4 traz uma pequena revisão da chamada "mecânica estatística" do processo de aprendizagem através do cálculo da curva de generalização para o perceptron linear. Discute-se também o limite teórico de generalização para o perceptron booleano dado pelo "algoritmo de Bayes" de Oppen e Haussler [Op 91]. Propomos uma versão de nosso algoritmo com iteração e a estudamos através de simulações. Encontramos que o desempenho do algoritmo de estabilidade esperada é numericamente indistinguível do desempenho teórico ótimo.

— Nossas discussões finais e conclusões são apresentadas no capítulo 5.

1.1 O PERCEPTRON E AS REDES NEURAIS DE ATRADORES

Em termos matemáticos um modelo de rede neural é definido como um grafo orientado com as seguintes propriedades [Mu 90]:

- uma variável de estado S_i associada a cada nodo i ,
- um peso J_{ij} associado a cada conexão (i,j) entre os nodos,
- um parâmetro h_{i0} interno associado ao nodo i ,
- uma função de transferência $g(S_i, J_{ij}, h_{i0})$ que determina o valor da variável S_i , em geral na forma

$$S_i = g\left(\sum_{j=1}^N J_{ij} S_j - h_{i0}\right) \quad (1.1)$$

Usualmente g é uma função sigmoideal do tipo $\tanh(x)$ ou $\text{sgn}(x)$.

Na terminologia corrente os nodos ou unidades são chamados de neurônios artificiais e a variável S_i é chamada de estado de ativação do neurônio i . O peso J_{ij} mede a influência que o estado S_j do neurônio j terá na determinação do estado do neurônio i e é chamado de eficácia sináptica ou simplesmente sinapse. Uma sinapse pode ser positiva (ativadora) ou negativa (inibidora). Note que, em geral, podemos ter J_{ij} diferente de J_{ji} .

O parâmetro interno h_{i0} é o limiar de ativação do neurônio. A soma $\tilde{h}_i = \sum_{j=1}^N J_{ij} S_j$ é chamada de potencial de membrana (iremos reservar a notação h_i para representar a grandeza normalizada $\tilde{h}_i / J$, onde usamos a norma $J = \left(\sum_{j=1}^N J_{ij}^2 \right)^{-1/2}$).

Ou seja, pela equação (1.1) temos que o estado de ativação do neurônio i será uma função da diferença entre o potencial de membrana e o limiar de ativação. Se $g(x)$ é a função sinal $\text{sgn}(x)$ temos neurônios formais de McCulloch-Pitts [McC 43] enquanto que se $g(x)$ é contínua temos neurônios analógicos.

O perceptron é um sistema adaptativo proposto por Rosenblatt no final dos anos 50 [Ro 58] como modelo simplificado de uma rede neural biológica. Sua arquitetura, conforme sugerida historicamente [Ro 62, Ar 87], consiste em duas camadas de neurônios artificiais e uma camada de unidades sensoras (figura 1.1). As unidades sensoras enviam sinais para a primeira camada de neurônios através de conexões fixas C_{jk} . O estado S_j ($j = 1, \dots, N$) desses neurônios é passado para n neurônios de saída com um certo peso dado por conexões ajustáveis (sinapses J_{ij}). Tanto os estados dos neurônios de saída (que chamaremos de σ_i , $i = 1, \dots, n$) como os da primeira camada são determinados por uma equação do tipo (1.1).

Entretanto não existe processo de ajuste nas conexões C_{jk} entre as unidades sensoras e os neurônios da primeira camada, de modo que sinais externos que produzem o mesmo conjunto de estados $\{S_j\}$ não poderão ser distinguidos pelo perceptron (uma das funções dessa camada de conexões fixa é justamente reduzir a variabilidade dos sinais externos). Assim, podemos abstrair o processo de formação dos estados S_j desses neurônios chamando-os de neurônios de entrada e o conjunto de estados $\{S_j\}$ de padrão de entrada. O perceptron faz um mapeamento entre padrões de entrada $\{S_j\}$ e classes $\{\sigma_i\}$. O processo de ajuste dos pesos sinápticos a fim de que um certo mapeamento seja realizado é chamado de processo de aprendizado e será melhor definido nas próximas seções.

No perceptron simples [Ro 62, Bl 62a] não existem conexões entre as unidades de uma mesma camada¹. Neste caso temos efetivamente n estruturas desacopladas, constituídas de N unidades de entrada e apenas uma unidade de saída. Basta então estudar o processo de aprendizagem na arquitetura básica esquematizada na figura 1.3, a qual tem sido referida universalmente na literatura simplesmente como "perceptron".

No início da década de 60 o perceptron simples e suas extensões (como por exemplo, perceptrons de várias camadas, ver figura (1.4)), foram intensamente estudados como dispositivos tecnológicos para reconhecimento de padrões [Ro 62, Bl 62a, Bl 62b]. Mostrou-se como estes sistemas apresentavam propriedades de "aprendizagem", "memória", "generalização" e "auto-organização". Outros modelos de redes neurais também foram propostos e estudados nesse período [Wi 60].

Em 1969 Minsk e Papert publicaram seu livro, agora clássico, "Perceptrons" [Mi 69], contendo uma análise rigorosa sobre as limitações intrínsecas destes sistemas, particularmente do perceptron simples que possui uma única camada de conexões adaptativas. Por outro lado, não existiam na época algoritmos eficazes para aprendizagem em perceptrons de multicamadas. Enquanto isso, o paradigma concorrente em inteligência artificial (escola simbolista) baseado em softwares especialistas para computadores digitais mostrava realizações significativas. Assim, o livro "Perceptrons" pode ser considerado um marco que sinaliza o encerramento da primeira fase de interesse não só nos perceptrons como nas redes neurais artificiais em geral.

Com o desenvolvimento de algoritmos de aprendizagem (backpropagation [Ru 85], máquina de Boltzmann [Se 86] etc.) para redes de multicamadas (que não possuem certas limitações do perceptron) o interesse retornou em meados da década de 80, principalmente entre alguns pesquisadores de IA e psicólogos cognitivistas insatisfeitos com o paradigma simbolista. Um paradigma alternativo se estabeleceu, chamado por alguns de "conexionismo", que concebe os processos cognitivos como sendo o resultado de um processamento paralelo distribuído [PDP 86].

Entre os físicos, o interesse foi despertado algum tempo antes pelo trabalho seminal de J. J. Hopfield [Ho 82] sobre o uso de redes neurais em problemas de

¹ É interessante notar que o esquema geral de Perceptron proposto historicamente com conexões entre neurônios de uma mesma camada faz com que haja retroalimentação e a possibilidade de uma dinâmica complexa, fato reconhecido nos anos 60 mas cujo estudo não teve continuidade. Em termos atuais, a figura 1.2 representa uma rede de atrator (ver mais adiante) cujo estado é passado unidirecionalmente para uma rede (perceptron simples) de saída.

FIGURA 1.1 — Modelo geral de perceptron conforme proposto historicamente por Rosenblatt [B1 62a, Ar 87]. As conexões C_{jk} são fixas e as J_{ji} são ajustáveis. Note que no modelo geral do perceptron (ao contrário do perceptron simples) é possível haver conexões entre unidades de uma mesma camada.

FIGURA 1.2 — Rede com retroalimentação (também chamada de rede de atratores). No modelo de Hopfield existe a restrição $J_{ij} = J_{ji}$.

FIGURA 1.3 — Arquitetura do perceptron simples. Basicamente, reduz-se a um neurônio de MacCulloch-Pitts já que os neurônios de entrada possuem uma função passiva.

FIGURA 1.4 — Exemplo de rede em multicamadas. Note que cada camada afeta apenas o estado da camada seguinte.

O problema de aprendizagem também têm sido estudado por métodos de mecânica estatística, principalmente a partir do trabalho pioneiro de Elizabeth Gardner [Ga 87, Ga88a] sobre o espaço de acoplamentos das redes neurais. A procura de algoritmos de aprendizagem eficazes e o estudo das respectivas dinâmicas também produziu numerosos resultados (ver artigos de revisão [Ab 90] e [Ki 89]). Nosso trabalho situa-se neste campo de estudo do processo de aprendizagem, em particular para o problema de "generalização" ou "extração de regra" a partir de exemplos e que será descrito na próxima seção.

Uma observação final sobre a terminologia usada neste trabalho: introduzimos nesta seção os três tipos de "linguagens" (ou semânticas) correntes na área de redes neurais: uma terminologia mais neutra ou matemática que poderia ser usada para descrever outros modelos conexionistas [Fa 90], uma terminologia de origem "neurobiológica" e uma terminologia "física" provinda da analogia com os sistemas magnéticos. Além disso, têm-se usado na literatura (por exemplo, [Wa 69, Op 91]) termos provindos da Epistemologia e Lógica indutiva. Isso é natural, pois como observou o filósofo da ciência Hilary Putnam [Pu 73]"(...) pode-se pensar num sistema de Lógica indutiva como se fosse um projeto de máquina de aprendizado. Isto é, como um projeto de computador que seja capaz de extrapolações, relativas a certas espécies de regularidades empíricas constatadas nos dados que lhe são oferecidos. O objetivo da Lógica indutiva é reconstruir a máquina universal de aprendizado".

Poderíamos ter feito um esforço neste trabalho para usar apenas a terminologia neutra, ou apenas uma das outras três. Optamos por uma certa fluidez, corrente nos trabalhos da área, usando termos provindos da Epistemologia, Psicologia, Biologia e Física, porque cremos ganhar com isso uma maior intuição e algumas analogias sugestivas (além do fato de que nem todos os termos possuem similares nas outras "linguagens"). Obviamente cada termo será adequadamente definido e delimitado. Um pequeno "dicionário" é fornecido na tabela 1.1.

Por outro lado, a terminologia neutra nos lembra de que a interpretação do perceptron como rede de neurônios é apenas isso, uma interpretação: se mecanismos do tipo do perceptron são usados em algum lugar na natureza (e concordamos com a citação de Marvin Minsky de que uma máquina tão simples deve ter sido usada em algum lugar), as unidades que denominamos "neurônios" tanto poderiam ser elementos que fazem algum tipo de computação dentro de um organismo unicelular como até sistemas cognitivos dentro do cérebro humano que fazem algum tipo de discriminação linear.

TABELA 1.1

Terminologia Usada em Modelos Conexionistas

<u>Matemática</u>	<u>Biologia</u>	<u>Física</u>	<u>Epistemologia</u>	<u>Símbolo</u>
Nodo	Neurônio	Spin	—	
Estado	Atividade	Orientação	—	S_i
Conexão	Sinapse	Acoplamento	—	
Peso da Co- nexão	Eficácia Sináptica	Força do acoplamento	—	J_{ij}
Parâmetro interno	Limiar de ativação	Campo externo	—	h_{0i}
—	Potencial de membrana	Campo local ou molecular	—	$\tilde{h}_i$
—	Ruído	Temperatura	—	T
Ajuste das conexões	Aprendizagem	—	Mudança nas hipóteses	
Rede aproximada	Aluno	—	Hipótese, Modelo ou Teoria	$ J\rangle$
Rede alvo	Professor	—	Lei, regra	$ B\rangle$
Elemento do Mapa	Exemplos	—	Dados de observação ou experiência	$(IS^{\mu}, \sigma_B^{\mu})$
Qualidade do ajuste	Habilidade de memorização	—	Poder explicativo da teoria	$L(\alpha)$
Interpolação/ Extrapolação	Habilidade de Generalização	—	Poder preditivo da teoria	$G(\alpha)$

1.2 APRENDIZAGEM E GENERALIZAÇÃO NO PERCEPTRON

As tarefas a serem desempenhadas por "máquinas de aprender" seriam aquelas de difícil definição e "algoritmização" (por exemplo, reconhecimento de caracteres caligráficos, reconhecimento de voz, controle de movimentos etc.). A implementação de uma função ou tarefa na máquina seria alcançado não por uma definição rigorosa e programação da mesma, mas sim por um processo adaptativo de ajuste de parâmetros internos do sistema durante uma "fase de aprendizagem", até se obter um desempenho satisfatório.

No campo de redes neurais e sistemas correlatos este conceito tem sido usado em três situações distintas (um artigo de revisão introdutório é [Gu 91]):

— **Aprendizagem supervisionada:** é fornecido ao sistema pares (entrada, saída) que exemplificam a tarefa a ser realizada. A aprendizagem consiste em uma minimização de alguma função custo que mede a diferença entre a saída do sistema e a saída desejada. É a situação de aprendizagem apropriada para o perceptron e que será melhor definida nas próximas seções.

— **Aprendizagem não supervisionada:** o sistema recebe apenas um conjunto de entradas que deverão ser agrupadas ou classificadas segundo algum critério de similaridade. Não são fornecidas, ou mesmo definidas, quais as saídas adequadas, número de classes etc. Um exemplo desse tipo de tarefa é dado pelos problemas de formação de agrupamentos ("clustering") de pontos dados sobre um plano.

— **Aprendizagem semi-supervisionada:** problema intermediário onde as saídas "corretas" para as entradas não são especificadas porém o efeito "macroscópico" dessas saídas em uma tarefa pré-determinada é fornecido. Por exemplo, uma rede neural deve determinar os movimentos adequados de um braço mecânico que sustenta um pêndulo invertido a fim de mantê-lo em equilíbrio. Não é fornecida à rede a sequência de respostas adequadas, mas é dada a informação sobre seu sucesso ou não em manter o pêndulo equilibrado.

As duas últimas situações de aprendizagem foram descritas de forma esquemática e superficial, e mais detalhes podem ser obtidos a partir da bibliografia citada. Nosso trabalho se refere à **aprendizagem supervisionada** de regras pelo perceptron, e que é descrita com detalhes a seguir.

1.2.1 MAPEAMENTO ALEATORIO, GENERALIZAÇÃO E "EXTRAÇÃO DE REGRA"

Grande parte dos trabalhos modernos sobre o perceptron [Ga 87, Ga 88a, Ga 88b]) preocupa-se com o problema de aprendizagem (supervisionada) de um mapeamento M entre αN padrões de entrada $|S^\mu\rangle$ e estados de saída σ^μ ,

$$M: |S^\mu\rangle \rightarrow \sigma^\mu, \quad \mu=1,\dots,\alpha N;$$

$$|S^\mu\rangle \equiv (S_1^\mu, \dots, S_N^\mu); \quad \sigma^\mu \equiv (\sigma_1^\mu, \dots, \sigma_n^\mu) \quad (S_i^\mu, \sigma_i^\mu \text{ reais ou binários})$$

onde σ_i^μ é escolhido de forma aleatória. Note que o domínio do mapa é o conjunto de αN "pontos" $|S^\mu\rangle$ e não o espaço que contém todos os $|S\rangle$ ($\mathbb{R}^N$ ou $(\pm 1)^N$).

O caso particular em que o número de neurônios de saída é igual ao de entrada, $n=N$, e onde desejamos que a saída seja igual a entrada, $\sigma^\mu = |S^\mu\rangle$, define o problema de autoassociação ou memória associativa para o perceptron. Este problema é equivalente ao de atração direta [Mu 90] (chegada ao atrator em apenas um passo) para o modelo de Little.

Em geral estes trabalhos com mapeamento aleatório se preocupam com o problema de cálculo da capacidade da rede, ou seja, da carga crítica α_c para a qual não é possível, através do algoritmo de aprendizagem em estudo, memorizar todos os αN padrões. O interesse nesses estudos se deve ao fato de que algoritmos de aprendizagem aplicáveis ao perceptron também são úteis para as ANN em seu uso como memórias associativas.

É possível construir os acoplamentos do perceptron de tal forma que não apenas o estado $|S^\mu\rangle$ mas toda uma "vizinhança" V_μ contendo estados "próximos" deste padrão seja mapeada em σ^μ . Esta vizinhança, no caso das ANN, é chamada de bacia de atração direta.

Às vezes descreve-se esta situação dizendo-se que o perceptron apresenta "generalização": o mapeamento, cujo domínio original se restringia aos estados $|S^\mu\rangle$, é generalizado para os estados "próximos" de $|S^\mu\rangle$, se torna robusto a ruído nos estados de entrada etc. Lembremos que, uma vez que σ^μ é escolhido de forma aleatória, o mapeamento foi definido apenas para os αN pontos, e uma infinidade de "generalizações" são possíveis, cada uma correspondendo a um conjunto definido de vizinhanças $\{V_\mu\}$. Note que neste caso não existem "generalizações" melhores ou piores, todas são igualmente aceitáveis em princípio.

Estamos interessados em um sentido mais forte do termo "generalização", algumas vezes chamado de "extração de uma regra" [De 87]. Neste caso, o mapeamento $\mathcal{M}$ é definido para qualquer $|S\rangle$ pertencente ao espaço de entradas, ou seja, para cada vetor $|S\rangle$ existe uma classificação correta σ . Os pares $(|S^\mu\rangle, \sigma^\mu)$ agora são chamados de "exemplos", pois apenas exemplificam a aplicação do mapa. O conjunto $L(\alpha)$ contendo $P = \alpha N$ exemplos é chamado "conjunto de treinamento".

A tarefa do perceptron agora é aproximar $\mathcal{M}$ usando a informação contida apenas em $L(\alpha)$. Obviamente isto só será possível se σ tiver uma dependência funcional de $|S\rangle$ em vez de ser definido aleatoriamente. E agora, o mapeamento $\mathcal{M}$ serve de padrão para se avaliar a qualidade da generalização produzida pela rede. Neste trabalho o termo **generalização** será usado neste sentido mais forte, o que já é usual na literatura.

1.2.2 - APRENDIZAGEM DE FUNÇÕES LINEARMENTE SEPARÁVEIS

Aprendizagem de uma regra ou função a partir de exemplos foi formalizada no contexto de teoria de aprendizagem de máquinas por Valiant [Va 84], que mostrou como certas classes de funções com N parâmetros podem ser "aprendidas" através de "exemplos" de modo polinomial em N . No caso do perceptron, iremos considerar a classe das funções Booleanas linearmente separáveis

$$\mathfrak{B}(|S\rangle) = \text{sgn}(\langle B | S \rangle - B_0) \quad (1.2)$$

onde

$\langle a | b \rangle$ é o produto escalar de dois vetores $\sum_{j=1}^N a_j b_j$

$|S\rangle$ é um vetor de entrada com N componentes binárias ($S_i = \pm 1$)

$|B\rangle$ é um vetor peso ($B_j \in \mathbb{R}$) que, sem perda de generalidade, está normalizado,

$$\langle B | B \rangle = 1 \quad (1.3)$$

Um perceptron com sinapses (acoplamentos) $|J\rangle$ e limiar de ativação (campo externo) h_0 pode reproduzir perfeitamente esta função uma vez que o estado do neurônio (spin) de saída é dado pela lei de ativação

$$\sigma_J(|S\rangle) = \text{sgn}(\langle J|S\rangle - h_0) \quad (1.4)$$

Geometricamente, os vetores $|S\rangle$ definem os 2^N vértices de um hipercubo de N dimensões e $|J\rangle$ é o vetor normal de um hiperplano que define uma superfície de separação entre os vértices classificados com $\sigma_J = 1$ e os vértices com $\sigma_J = -1$. A distância entre o hiperplano e a origem é dada por h_0 .

Neste trabalho usaremos sempre $h_0 = 0$. O caso com h_0 não nulo pode ser entendido como um problema em um espaço com $N+1$ dimensões onde a $(N+1)$ -ésima componente dos vetores $|S\rangle$ é sempre $+1$ e a $(N+1)$ -ésima sinapse tem valor $-h_0$.

Para implementar a função $\mathcal{B}$, no limite $N \rightarrow \infty$, basta que o perceptron tenha seus pesos sinápticos ajustados de modo que não haja diferença macroscópica entre os vetores $|J\rangle$ e $|B\rangle$. A este processo de ajuste chamamos de aprendizagem da função ou "regra" $\mathcal{B}$. Aprendizagem através de "exemplos" significa que o processo de ajuste usa apenas a informação contida em exemplos da aplicação da função, ou seja, pares $(|S^\mu\rangle, \sigma_B^\mu)$, onde $\sigma_B^\mu \equiv \mathcal{B}(|S^\mu\rangle)$.

Considerando apenas o caso de funções linearmente separáveis estamos limitando nosso estudo a funções que podem ser perfeitamente reproduzidas pelo perceptron. Outros problemas de generalização têm sido considerados: pode-se estudar o problema definido acima incluindo diversas fontes de ruído nos exemplos [Gy 90a, Gy 90b], ou então o problema de se separar os exemplos em classes definidas por vetores protótipos [Ha 90, dG 89], ou ainda pode-se analisar o desempenho do perceptron na aproximação de vários tipos de funções que não podem ser totalmente implementadas nesta arquitetura [Wt 92, So 91]. O problema de generalização de funções linearmente separáveis serve de ponto de partida e de comparação para o estudo destas situações mais complexas.

1.3 DINÂMICAS DE APRENDIZAGEM

Neste trabalho estamos considerando algoritmos de aprendizagem do tipo

$$|J(t, \mu)\rangle = |J(t, \mu-1)\rangle + \frac{1}{N} |B^\mu(t)\rangle \quad (1.5)$$

onde

$\mu = 1, \dots, \alpha N$ é o índice do exemplo $(|S^\mu\rangle, \sigma_B^\mu)$ no conjunto $\mathcal{L}$.

$t = 0, \dots, \infty$ refere-se ao ciclo de iteração (ou seja, uma apresentação completa de todos os exemplos de L),

$|D^\mu(t)\rangle$ é um vetor de correção que pode depender de t e μ diretamente ou de grandezas como $|J(t, \mu-1)\rangle$, σ_E^μ , $|S^\mu\rangle$ etc. Note que, devido ao fator $1/N$ na equação de evolução de $|J^\mu\rangle$, a mudança que o vetor de correção traz ao vetor de acoplamentos é de natureza infinitesimal no limite $N \rightarrow \infty$.

No caso $\mu = 1$ temos

$$|J(t, 1)\rangle = |J(t-1, \alpha N)\rangle + \frac{1}{N} |D^1(t)\rangle \quad (1.6)$$

e a condição inicial ("tabula rasa") é $|J(0, 0)\rangle = 0$.

De acordo com o modo como o vetor $|D^\mu(t)\rangle$ é construído podemos distinguir as seguintes dinâmicas de aprendizagem:

– Dinâmica assíncrona ou sequencial: o vetor de correção $|D^\mu(t)\rangle$ depende do estado do vetor de acoplamentos $|J(t, \mu-1)\rangle$ corrigido pelos $\mu-1$ exemplos usados no ciclo t , de modo que a correção é necessariamente sequencial.

A dinâmica descrita acima é uma dinâmica sequencial de ordem fixa, ou seja, em que cada exemplo aparece uma única vez no ciclo t e a ordem dos exemplos é sempre a mesma em cada ciclo. Podemos usar também uma dinâmica de ordem aleatória onde a ordem dos exemplos é escolhida de forma aleatória a cada início de ciclo, ou ainda uma dinâmica (totalmente) aleatória onde cada incremento é dado por um exemplo escolhido ao acaso de forma que apenas em média cada exemplo é usado uma vez por ciclo.

– Dinâmica síncrona ou em paralelo: o vetor de correção $|D^\mu(t)\rangle$ depende apenas do estado da rede dado pelo ciclo de iteração anterior, ou seja, de $|J(t-1)\rangle \equiv |J(t-1, \alpha N)\rangle$. Os exemplos são aprendidos em paralelo. Podemos reescrever a equação de aprendizagem na forma

$$|J(t)\rangle = |J(t-1)\rangle + |D(t)\rangle \quad (1.7)$$

com

$$|D(t)\rangle \equiv \frac{1}{N} \sum_{\mu=1}^{\alpha N} |D^\mu(t)\rangle \quad (1.8)$$

- Dinâmica sequencial em blocos: Definimos uma partição do conjunto de aprendizagem $\mathcal{P}(\mathcal{L}) = \{\mathcal{L}_1, \dots, \mathcal{L}_p\}$ em p blocos, cada um deles possuindo n_k exemplos, com $\sum_{k=1}^p n_k = \alpha N$. Os exemplos do bloco k são aprendidos em paralelo e o vetor de correção para um certo μ dentro do bloco depende apenas do estado da rede dado pelo aprendizado no bloco anterior, ou seja, de $|J(t, \mu_k)\rangle$ onde μ_k é o último exemplo do bloco k . Assim, o aprendizado se dá sequencialmente em blocos, mas em paralelo dentro dos blocos. O caso especial $p = 1$ corresponde à dinâmica síncrona e o caso $p = \alpha N$ à dinâmica assíncrona citados anteriormente.

Existem outras variações menos importantes no modo como $|D^\mu(t)\rangle$ pode depender dos estados anteriores do vetor sináptico. As citadas bastam para o nosso estudo. Veremos a seguir como diversas restrições no modo de se construir o vetor correção $|D^\mu(t)\rangle$ definem situações de aprendizagem diferentes.

1.4 SITUAÇÕES DE APRENDIZAGEM

Em geral os algoritmos considerados na literatura podem ser colocados na forma dada pela equação (1.5). Por exemplo, para a regra de Hebb, muito estudada no campo das redes de atratores, o vetor de correção toma a forma (note que não há iteração, dispensando-se o índice t)

$$|D^\mu\rangle = \sigma_B^\mu |S^\mu\rangle \quad (1.9)$$

O famoso algoritmo Perceptron de Roseblatt [Ro 62] pode ser escrito como

$$|D^\mu(t)\rangle = \eta \Theta(-\sigma_B^\mu \sigma_J^\mu(t)) \sigma_B^\mu |S^\mu\rangle \quad (1.10)$$

onde $\Theta(x)$ é a função de Heaviside e η é um fator de passo pequeno. Mesmo uma regra não local como a de "pseudoinversa" [Pe 85, Ka 87] (mais abaixo descrevemos o que se entende por algoritmo local) pode ser colocada na forma (1.5) [Di 87] (todos esses algoritmos serão devidamente apresentados mais tarde).

Partindo então do caso bastante geral da equação de aprendizagem (1.5) vamos encontrar uma solução para $|D^\mu\rangle$ com generalização ótima sem impor nenhuma restrição sobre o modo de se construir este vetor. Esta solução é relativamente trivial e a chamaremos de $|D^\mu\rangle^*$. A seguir vamos impor algumas restrições sobre o modo de se construir $|D^\mu\rangle$, que serão discutidas a seguir (localidade, condição de caixa preta, apresentação única dos exemplos etc.).

Um conjunto dessas restrições irá definir uma "situação de aprendizagem" S . Para cada situação S , haverá então um desempenho ótimo em generalização (usaremos como medida de desempenho o modo assintótico ($\alpha \rightarrow \infty$) com que $\|J\|$ se aproxima de $\|B\|$, ver capítulo 2). Nossa proposta será obter esse desempenho ótimo através da melhor aproximação de $\|D^u\|$ * que satisfaça as restrições de S .

1.4.1 TIPOS DE PERCEPTRONS

A arquitetura simples esquematizada na figura (1.3) não define completamente o perceptron. Devemos especificar ainda o tipo de função de transferência obedecida pelo neurônio de saída e as restrições sobre as sinapses.

Quanto à função de transferência podemos distinguir três tipos de perceptrons:

- Perceptron linear: a função de transferência é linear, $g(x) = x$.
- Perceptron analógico: a função de transferência é uma função contínua não linear sigmoideal do tipo $\tanh(x)$.

- Perceptron booleano: este é o perceptron originalmente estudado por Roseblatt, formado por neurônios de MacCulloch-Pitts, ou seja, onde $g(x) = \text{sgn}(x)$.

Quanto as restrições sobre as os valores das sinapses existe uma variedade muito maior, mas os tipos mais estudados na literatura são:

- Perceptron com pesos contínuos: possui sinapses que podem assumir qualquer valor real.

- Perceptron esférico: possui sinapses reais porém sujeitas à condição $\langle J | J \rangle = \text{constante}$.

- Perceptron binário: as sinapses podem assumir apenas os valores ± 1 . É um caso particular do perceptron de sinapses discretas cujas sinapses podem assumir os valores $\pm 1, \pm 2, \dots, \pm J$.

1.4.2 CONDIÇÕES DE "CAIXA NEGRA", DE "LOCALIDADE" E DE "LINEARIDADE"

Existem certas restrições básicas sobre quais grandezas podem ser usadas para se construir o vetor de correção $|D^\mu\rangle$. Na literatura estas restrições são tidas como as "regras do jogo" às quais os algoritmos deveriam se submeter. Fizemos um esforço em explicitar as restrições para o nosso problema, mesmo aquelas que normalmente são assumidas de forma tácita. Estas restrições podem ser agrupadas em duas classes que chamaremos de:

a) Condição de "caixa negra":

— D_j^μ não pode depender diretamente de $|B\rangle$ nem do campo local no perceptron professor $b_\mu = \langle B | S^\mu \rangle$, mas apenas da saída deste, $\sigma_B^\mu = g(b_\mu)$. Note que esta restrição é essencial para a definição de aprendizagem "através de exemplos". Também não pode haver dependência de uma função de $|B\rangle$ do tipo $\langle J | B \rangle$. Esta última restrição será muito importante em nosso trabalho.

b) Condição de "localidade":

— D_j^μ não deve depender diretamente dos valores das componentes de $|J(\mu-1)\rangle$, a não ser talvez do valor da componente J_j que está sendo atualizada em (1.5), do campo local $\tilde{h}_\mu = \langle J | S^\mu \rangle$ sobre o neurônio de saída ou do estado deste, $\sigma_j^\mu = g(\tilde{h}_\mu)$. Assim, um algoritmo local não deve depender diretamente de, por exemplo, $J = \sqrt{\langle J | J \rangle}$ ou de $h_\mu = \tilde{h}_\mu / J$.

— D_j^μ pode depender diretamente apenas do estado do neurônio de entrada S_j^μ mas não das outras componentes de $|S^\mu\rangle$.

Existem algoritmos conhecidos (por exemplo, o do projetor ou pseudoinversa [Pe 85, Ka 87]) que não satisfazem esta última restrição, que não é essencial mas possui motivação biológica. Em todo caso, há sempre o interesse em encontrar algoritmos locais (por exemplo, demonstra-se que a solução de pseudoinversa corresponde ao estado assintótico do algoritmo local Adaline [Di 87]).

Finalmente, podemos permitir sem problemas que $|D^\mu\rangle$ dependa do estágio de aprendizagem, ou seja, do número $\mu-1$ de exemplos já incluídos em $|J(t, \mu-1)\rangle$ e do índice t do ciclo de iteração. Desse modo, em termos de variáveis independentes, o vetor de correção satisfazendo as condições de localidade e caixa negra deverá ter a forma

$$D_j^\mu(t) \equiv D(\mu, t, J_j, \tilde{h}_\mu, S_j^\mu, \sigma_B^\mu) \quad (1.11)$$

Note também que estamos assumindo, por simetria, que a função D não depende explicitamente da posição da sinapse na rede, ou seja, o modo como D depende das variáveis listadas em (1.11) é o mesmo para todas as sinapses. Isso pode parecer óbvio dada a arquitetura do perceptron, mas é uma condição que não precisaria ser usada para outras arquiteturas (multicamadas).

Faremos ainda mais uma restrição na forma da função D que é comum nos trabalhos desta área [Ab 90], ou seja, que a dependência explícita em J_j e S_j^μ do vetor de correção seja linear

$$|D^\mu(t)\rangle = F(\mu, t, \tilde{h}_\mu, \sigma_B^\mu) |S^\mu\rangle + \Omega(\mu, t, \tilde{h}_\mu, \sigma_B^\mu) |J(t, \mu-1)\rangle \quad (1.12)$$

Chamaremos esta restrição de "**condição de linearidade**".

Em geral o último termo é usado para limitar o valor absoluto das sinapses [He 91] ou satisfazer condições de normalização [Gr 91].

Embora a maior parte dos algoritmos de aprendizagem considere apenas sinapses com esta condição de linearidade, sinapses não lineares também têm sido propostas e estudadas [van He 86].

1.4.3 CONTROLE SOBRE O CONJUNTO DE APRENDIZAGEM

Quanto ao caráter do conjunto de aprendizagem L podemos distinguir duas situações básicas [Va 84]. O conjunto de exemplos pode ser gerado a partir de uma distribuição de probabilidade dada a priori, ou seja, o desempenho ou estado do perceptron não afeta a formação do conjunto L . Chamaremos esta situação de "**aprendizagem passiva**".

Por outro lado, se não houver esta restrição, podemos assumir que o estado ou desempenho do perceptron lhe permite (ou ao seu treinador) ir escolhendo os exemplos mais adequados a cada etapa do aprendizado, ou seja, a distribuição de probabilidade dos exemplos não é fixa mas pode ser modulada por algum critério de seleção. Neste caso teremos uma "**aprendizagem ativa**" ou "**seleção de exemplos**" [Ki 90]. Especificando melhor, estudaremos as seguintes situações:

a) **Aprendizagem passiva:** Neste caso o estudante tem acesso apenas a um dado conjunto de aprendizagem L contendo exemplos gerados por uma fonte EXEMPLOS a partir de uma distribuição dada e ele precisa obter o melhor desempenho em generalização usando apenas a informação contida nesses exemplos.

Seguindo a maior parte da literatura, estudaremos a aprendizagem passiva no caso particular do conjunto de aprendizagem L ser construído a partir de nN vetores exemplos escolhidos de forma independente e aleatória com distribuição de probabilidade $P(\mathbf{I} | \mathbf{S}^\mu)$ uniforme. Isto significa que a quantidade

$$b_\mu = \langle \mathbf{B} | \mathbf{S}^\mu \rangle \quad (1.14)$$

e o campo local normalizado no perceptron

$$h_\mu = \frac{\tilde{h}_\mu}{\sqrt{\langle \mathbf{J} | \mathbf{J} \rangle}} = \frac{\langle \mathbf{J} | \mathbf{S}^\mu \rangle}{\sqrt{\langle \mathbf{J} | \mathbf{J} \rangle}} \quad (1.16)$$

são variáveis gaussianas reduzidas (lembramos que $\langle \mathbf{B} | \mathbf{B} \rangle = 1$) com distribuição de probabilidade conjunta

$$\begin{aligned} P(b_\mu, h_\mu) &= P(h_\mu) P(b_\mu | h_\mu) \\ &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{h_\mu^2}{2}\right) \times \frac{1}{\sqrt{2\pi(1-\rho^2)}} \exp\left(-\frac{(b_\mu - \rho h_\mu)^2}{2(1-\rho^2)}\right) \end{aligned} \quad (1.17)$$

onde ρ é a correlação entre h_μ e b_μ

$$\rho = \langle h_\mu b_\mu \rangle = \frac{\langle \mathbf{J} | \mathbf{B} \rangle}{\sqrt{\langle \mathbf{J} | \mathbf{J} \rangle}} \quad (1.18)$$

Vetores com b_μ grande são fáceis de classificar por um perceptron aproximado $|\mathbf{J}\rangle = |\mathbf{B}\rangle + |\delta\rangle$, (onde $\langle \delta | \delta \rangle$ é pequeno) uma vez que neste caso as saídas dos perceptrons $\sigma_B^\mu = \text{sgn}(b_\mu)$ e $\sigma_J^\mu = \text{sgn}(h_\mu)$ possuem alta correlação.

Embora durante a aprendizagem não temos acesso a b_μ , a dificuldade de classificação de um exemplo pode ser avaliada pelo campo local h_μ de modo que os exemplos não são equivalentes para o estudante e portanto não há motivo para que sejam incluídos com igual peso em seu vetor $|\mathbf{J}\rangle$ como acontece na regra de Hebb

simples. Regras de Hebb com pesos foram usadas com sucesso para se acelerar o aprendizado (memorização) no caso de mapeamento aleatório [Ab 89a].

Iremos explorar essa idéia tentando encontrar uma função peso F ótima a ser dada aos exemplos de modo a se obter máxima probabilidade de generalização $G(\alpha)$ (que será definida mais tarde). Isso é possível porque b_μ (e portanto, em certa medida h_μ) mede a proximidade do vértice do hipercubo (dado por $|S^\mu\rangle$) em relação ao hiperplano de separação definido pelo perceptron alvo. Os pontos com b_μ próximo de zero estão perto da fronteira de separação e portanto a sua classificação σ_B^μ contém maior informação sobre a localização exata dessa fronteira.

b) Aprendizagem ativa: Na segunda situação de aprendizagem definida por Valiant, a função σ_B^μ é acessível, em princípio, a partir de uma fonte ORÁCULO para qualquer vetor $|S^\mu\rangle$. Neste caso, o perceptron estudante (ou seu treinador) pode gerar "questões" $|S^\mu\rangle$ selecionadas com características específicas e assim controlar a distribuição de probabilidade $P(|S^\mu\rangle)$. Ou seja, além da possibilidade de se dar um peso aos exemplos de L como no caso anterior, existe outro grau de liberdade a ser ajustado: a própria composição de L . Podemos incluir em L apenas exemplos selecionados de modo a otimizar $G(\alpha)$.

Esta situação não é particularmente artificial como pode parecer: é análoga à uma atitude científica experimental que não se contenta apenas em recolher os exemplos espontâneos de um fenômeno mas pede dados específicos à Natureza (que aqui faz o papel da fonte ORÁCULO) através de experimentos (questões) bem escolhidos. Corresponde ainda a uma situação onde a máquina ou organismo tem capacidade de realizar certas ações "testes" e observar os resultados dessas ações na realização de uma certa tarefa. Watanabe [Wa 69] chama a primeira situação de "observação passiva" e a segunda situação de "experimentação ativa".

Pictóricamente poderíamos dizer que a otimização destes graus de liberdade correspondem ao problema de "didática" para redes neurais: devemos escolher bem os exemplos a serem ensinados ao perceptron (ou então deixar o "aluno" elaborar "questões" $|S^\mu\rangle$) e formar assim o conjunto L . Além disso devemos encontrar qual o peso adequado a ser dado a cada exemplo durante o processo de aprendizagem.

1.4.4 APRESENTAÇÃO ÚNICA DOS EXEMPLOS

"As circunstâncias em que a experiência foi feita não se reproduzirão jamais. O fato observado jamais se repetirá. (...) O que é então, uma boa experiência? É a que nos desvenda algo além de um fato isolado; é a que nos permite prever, isto é, a que nos permite generalizar (...).

Henri Poincaré

A apresentação única dos exemplos é uma restrição que define uma situação de aprendizagem com interesse próprio: uma máquina ou organismo frente a um ambiente natural experimenta o exemplo de um dado fenômeno ("regra") em um curto espaço de tempo e geralmente em uma única vez (as próximas experiências serão novos exemplos da mesma regra, mas não repetições idênticas do mesmo exemplo). Isto equivale a exigir que os algoritmos de aprendizagem para esta situação estejam restritos a apenas um ciclo de apresentação dos exemplos, $t = 1$.

Em outras palavras, o vínculo consiste em considerar o "exemplo" como um acontecimento efêmero: não existe um mecanismo de memória externa ou interna ao perceptron que guarde os exemplos e repita aqueles que necessitam de melhor aprendizado. As modificações que o exemplo produz no vetor sináptico devem ter um efeito imediato, serem feitas em um único passo e o problema consiste em se obter máximo ganho em generalização nesta única apresentação do exemplo.

Tecnicamente, a mudança em $|J\rangle$ nesta situação é equivalente à produzida no primeiro passo de um processo de aprendizagem iterativo onde os exemplos sejam usados um número não limitado de vezes. Mas enfatizamos que o problema de aprendizagem não pode ser reduzido a isso. Se assim fosse a busca de generalização ótima deveria considerar apenas os algoritmos iterados, os quais são necessariamente mais eficientes. Mas não existe uma capacidade de generalização ótima per se: cada situação de aprendizado, com seus vínculos e restrições específicas (algoritmo local ou não local, possibilidade de seleção de exemplos ou não, possibilidade de repetição dos exemplos ou não etc.) determinará para o perceptron um desempenho ótimo em generalização naquela situação. Devemos então encontrar, para cada situação de aprendizagem que nos interesse, o

algoritmo que produzirá esta habilidade de generalização máxima. E cremos que a apresentação única de exemplos é uma situação interessante a se estudar.

Resumindo, neste trabalho queremos encontrar algoritmos de aprendizagem para o perceptron que produzam habilidade de generalização ótima para quatro situações de aprendizagem distintas:

- Aprendizagem passiva sem repetição: Cada exemplo é usado uma única vez e não existe seleção dos exemplos a serem apresentados. O algoritmo é do tipo assíncrono não iterado.

- Aprendizagem ativa sem repetição: Existe controle sobre a distribuição de probabilidade dos exemplos, porém estes são apresentados uma única vez. O algoritmo também é do tipo assíncrono não iterado.

- Aprendizagem passiva iterada: Cada exemplo do conjunto de treinamento L dado a priori pode ser usado um número não limitado de vezes durante o processo de aprendizagem. O algoritmo pode ser síncrono ou assíncrono.

- Aprendizagem ativa iterada: Existe seleção dos exemplos e estes são usados um número não limitado de vezes.

1.5 REVISÃO DO PROBLEMA E PLANO DE TRABALHO

Nos estudos modernos sobre o perceptron, o problema de aprendizagem de funções Booleanas linearmente separáveis foi inicialmente proposto por Gardner para o perceptron binário [Ga 89]. Para o perceptron booleano de sinapses contínuas os primeiros cálculos analíticos foram feitos por Vallet para a regra de Hebb [Vl 89a, Vl 90]. Algumas interpolações entre a solução de Hebb e a de Pseudo-inversa foram estudadas numericamente [Vl 89b, Re 89].

As curvas de generalização para a solução do projetor ou de "pseudo-inversa" e para o "perceptron de ótima estabilidade" foram obtidas por Oppen et al. [Op 90] usando-se o método de réplicas, seguindo a abordagem feita por Gardner e Derrida [Ga 88b] para o caso de mapeamento aleatório. A capacidade de generalização para perceptrons satisfazendo a função custo de Gardner e Derrida foi obtida por Gyorgyi e Tishby [Gy 90a]. O mesmo foi feito recentemente para o algoritmo Perceptron de Roseblatt e para o algoritmo de relaxação por Meir e Fontanari [Me 91].

O limite teórico de desempenho para o perceptron booleano foi dado pelo "algoritmo de Bayes" [Op 91] (mas este algoritmo não pode ser implementado em um perceptron simples).

Outros trabalhos consideraram o problema de generalização para o perceptron binário [Gy 90b, So 90, Se 91].

A situação de aprendizagem em que é possível a seleção de exemplos foi inicialmente proposta para o perceptron booleano por Kinzel e Ruján [Ki 90], que estudaram analiticamente a regra de Hebb e numericamente o perceptron de ótima estabilidade. A curva analítica de generalização para este último foi calculada recentemente [Wt 92], tanto para o perceptron booleano como para o binário. Não se conhece ainda o limite teórico de desempenho para esta situação de aprendizagem.

Todos estes trabalhos se referem ao desempenho dos algoritmos após longo tempo de aprendizagem, ou seja, as propriedades estacionárias ou de equilíbrio. Quanto aos aspectos propriamente dinâmicos pouco se tem feito. Conhecemos apenas resultados para o caso de mapeamento aleatório [Op 88, Op 89, An 89]. Para o problema de generalização no perceptron linear, Hertz estudou a dinâmica de um algoritmo tipo Adaline [He 91].

Assim, a tendência geral seguida na literatura foi tomar um algoritmo conhecido (muitas vezes desenvolvido no contexto de memórias associativas e mapeamento aleatório), aplicá-lo ao problema de generalização e calcular seu desempenho (eventualmente mostrando que este é melhor que o dos anteriores).

Neste trabalho tentaremos seguir um caminho inverso, de espírito dedutivo: partindo do requisito de ótima generalização no primeiro passo da dinâmica, iremos obter uma nova família de algoritmos de aprendizagem (que chamaremos de algoritmos de "estabilidade esperada") e mostrar que, fazendo-se certas restrições ou aproximações, obtém-se algoritmos já conhecidos.

Estudaremos o primeiro passo da dinâmica desse novo algoritmo analiticamente e seu comportamento para longos tempos numericamente.

CAPITULO 2

APRENDIZAGEM E GENERALIZAÇÃO NO PERCEPTRON

Neste capítulo introduzimos as medidas de desempenho do perceptron: a habilidade de aprendizagem $L(\alpha)$ e a habilidade de generalização $G(\alpha)$. Tais grandezas são melhor entendidas como casos particulares de uma probabilidade de reconhecimento $P_R(\alpha, m)$, onde o parâmetro m mede a correlação entre um padrão incluído no conjunto de treinamento e um padrão a ser classificado que é uma versão com ruído do primeiro. Como exemplo, calculamos $P_R(\alpha, m)$ para a regra de Hebb. Introduzimos também uma nova medida de aprendizagem para o caso da apresentação única dos exemplos: a habilidade de recordação de lista $R(\alpha', \alpha)$.

Embora a maior parte dos conceitos definidos a seguir aplicam-se em um perceptron com número de entradas N finito, estaremos sempre pensando no limite termodinâmico $N \rightarrow \infty$, que será usado como uma aproximação para redes com grande número de neurônios. Nossos cálculos para as curvas de desempenho do perceptron serão feitos para este caso, seguindo a linha de trabalho majoritária na literatura. Veremos que em geral uma simulação usando $N \sim 50$ ou 100 neurônios de entrada será bem descrita por esta aproximação.

2.1 CURVAS DE APRENDIZAGEM E GENERALIZAÇÃO

Considere um "conjunto de aprendizagem" $\mathcal{L} = \{\mathcal{E}_\mu\}$ com αN "exemplos" de uma função Booleana $\mathcal{B}$, ou seja, pares $\mathcal{E}_\mu = \{\mathbf{I} \mathbf{S}^\mu\}, \sigma_B^\mu\}$ de vetores de entrada ("padrões") binários $\mathbf{I} \mathbf{S}^\mu$ e respectivas imagens $\sigma_B^\mu \equiv \mathcal{B}(\mathbf{I} \mathbf{S}^\mu)$. Os padrões de entrada possuem uma dada distribuição $P(\mathbf{I} \mathbf{S}^\mu)$ arbitrária.

Estaremos considerando apenas funções Booleanas linearmente separáveis, que têm a forma (ver seção 1.2.2)

$$\sigma_B^\mu = \text{sgn}(\langle \mathbf{B} | \mathbf{I} \mathbf{S}^\mu \rangle) = \text{sgn}(b_\mu) \quad (2.1)$$

de modo que podemos interpretar σ_B^μ como a saída gerada por um perceptron de referência com vetor sináptico $\mathbf{I} \mathbf{B}$. A variável $b_\mu \equiv \langle \mathbf{B} | \mathbf{I} \mathbf{S}^\mu \rangle$ é o campo local no neurônio de saída do perceptron de referência (assumimos $\langle \mathbf{B} | \mathbf{I} \mathbf{B} \rangle = 1$).

Uma regra ou algoritmo de aprendizagem é um procedimento, direto ou iterativo, para se construir um perceptron $|J\rangle$ tal que $|B\rangle$ (que chamaremos de "professor") seja aproximado por $|J\rangle$ (que chamaremos de "aluno" ou "estudante"²).

Uma medida dessa aproximação é a "probabilidade de reconhecimento" $P_R(\alpha)$, definida como a probabilidade, medida sobre os possíveis conjuntos de aprendizagem $\mathcal{L}(\alpha)$, de que a saída do perceptron estudante para um certo vetor teste $|S\rangle$,

$$\sigma_J = \text{sgn}(h), \quad h \equiv \frac{\langle J | S \rangle}{\sqrt{\langle J | J \rangle}} \quad (2.2)$$

seja a mesma do professor, isto é,

$$P_R(\alpha) = \langle P_R^{\mathcal{L}}(\alpha) \rangle_{\{\mathcal{L}\}}$$

onde

$$P_R^{\mathcal{L}}(\alpha) = \text{PROB}(\sigma_J \cdot \sigma_B = 1)$$

Se denotarmos por $P(b,h)$ a probabilidade conjunta das variáveis b e h , podemos escrever

$$P_R^{\mathcal{L}}(\alpha) = \int_{-\infty}^{\infty} dh \int_{-\infty}^{\infty} db P(b,h) \Theta(h \cdot b) = 2 \int_0^{\infty} dh \int_0^{\infty} db P(b,h) \quad (2.4)$$

Note que nas expressões acima a saída do aluno σ_J depende de $\mathcal{L}$ porque $|J\rangle$ é construído a partir de um determinado conjunto de treinamento.

Se o conjunto de teste é o próprio $\mathcal{L}$, (ou seja, o vetor $|S\rangle$ pertence a $\mathcal{L}$ e foi usado na construção de $|J\rangle$) então $P_R(\alpha)$ é chamada de "probabilidade de aprendizagem" $L(\alpha)$. A quantidade $e_1(\alpha) = 1 - L(\alpha)$ será chamada de erro de aprendizagem. Outro nome para $L(\alpha)$ é habilidade de aprendizagem.

Se o conjunto de teste é escolhido de forma estatisticamente independente de $\mathcal{L}$, então $P_R(\alpha)$ é chamado de "probabilidade de generalização" $G(\alpha)$ e a quantidade $e_2(\alpha) = 1 - G(\alpha)$ será o erro de generalização. $G(\alpha)$ também é chamada de habilidade de generalização.

² Alguns autores denominam $|B\rangle$ de "regra" e $|J\rangle$ de "hipótese" [Op 91].

De forma mais geral, nós podemos definir a probabilidade de reconhecimento $P_R(\alpha, m)$, onde os vetores de teste $|S\rangle$ são versões com ruído dos padrões incluídos em L . O parâmetro m , $-1 \leq m \leq 1$, mede a correlação entre o vetor de teste $|S\rangle$ e o vetor não corrompido $|S^\mu\rangle$ do qual foi gerado

$$m = \frac{\langle S | S^\mu \rangle}{N} \quad (2.5)$$

Desse modo, temos que $G(\alpha) \equiv P_R(\alpha, 0)$ e $L(\alpha) \equiv P_R(\alpha, 1)$. Estas definições ficarão claras mais tarde quando calcularmos $P_R(\alpha, m)$ para a regra de Hebb.

Definimos agora a correlação ("overlap") entre o perceptron aluno e o perceptron professor

$$\rho_J = \frac{\langle B | J \rangle}{B J} \quad (2.6)$$

onde

$$B = \sqrt{\langle B | B \rangle}, \quad J = \sqrt{\langle J | J \rangle}$$

que é justamente o cosseno do ângulo entre os vetores $|B\rangle$ e $|J\rangle$. O overlap típico para um conjunto de treinamento de tamanho αN será denotado por $\rho(\alpha)$.

Como foi colocado anteriormente, o nosso interesse principal é a habilidade média de generalização $G(\alpha)$. Mostra-se [Op 90, Gy 90a] a partir de (2.4) que essa grandeza é uma função monotônica do overlap típico $\rho(\alpha)$

$$G(\alpha) = 1 - \frac{1}{\pi} \arccos(\rho(\alpha)) \quad (2.7)$$

Uma maneira simples de visualizar este resultado é mostrada na figura (2.1): P_B e P_J são os hiperplanos definidos pelos vetores $|B\rangle$ e $|J\rangle$ respectivamente. As regiões achuradas representam o volume do hiperespaço onde um vetor teste $|S^\mu\rangle$ recebe classificação σ_J diferente de σ_B . Portanto, a probabilidade de erro de classificação é igual a razão entre este volume e o volume total, que por sua vez são proporcionais aos arcos Θ e π , ou seja,

$$e_g = \frac{\Theta}{\pi} \quad (2.8)$$

de onde segue (2.7). No caso em que $e_g \rightarrow 0$ para $\alpha \rightarrow \infty$ temos uma relação simples que será muito usada no futuro

$$e_g \simeq \pi^{-1} \sin \Theta = \pi^{-1} \sqrt{1 - \rho^2} \simeq \pi^{-1} \sqrt{2(1 - \rho)} \quad (2.9)$$

Assim, se o nosso objetivo é otimizar a habilidade de generalização, deveremos maximizar $\rho(\alpha)$.

Foram dadas duas definições de habilidade de generalização $G(\alpha)$:

a) probabilidade de classificação correta de um padrão $|S\rangle$ escolhido de forma estatisticamente independente do conjunto de treinamento L ;

b) probabilidade de classificação correta de um padrão $|S\rangle$ com correlação m nula com os vetores do conjunto de aprendizagem.

Devemos notar que as duas definições só serão equivalentes no limite $N \rightarrow \infty$, dado que um vetor escolhido aleatoriamente terá correlação da ordem de $1/\sqrt{N}$ com os vetores de L . Este detalhe é importante porque existem duas maneiras de se medir $G(\alpha)$ em uma simulação no caso do perceptron:

a) podemos escolher de forma estatisticamente independente de L um conjunto de teste contendo um número grande de exemplos $|S\rangle$ e testá-los após a fase de aprendizagem, aproximando $G(\alpha)$ pela frequência média de acertos, ou

b) podemos medir o overlap médio $\rho(\alpha)$ e calcular $G(\alpha)$ através de (2.7).

O método a) apresenta efeitos produzidos pelo tamanho finito da rede maiores que o método b). Isto acontece porque no método a), para N finito, a probabilidade de um vetor de teste possuir m macroscópico com algum vetor de L não pode ser desprezada e sua probabilidade de reconhecimento poderá ser da ordem de $P_R(N^{-1/2}, \alpha)$, maior que o $G(\alpha)$ dado por (2.7)³. Como regra geral esperamos que o valor de $G(\alpha)$ estimado pelo método a) seja sempre ligeiramente superior aos obtidos pelo método b) ou pelo cálculo teórico. Isto será observado nas simulações apresentadas no capítulo 3.

Neste ponto gostaríamos de enfatizar a diferença entre os dois sentidos em que estamos usando a palavra "aprendizagem":

— Aprendizagem dos exemplos refere-se à capacidade do perceptron de reproduzir os exemplos e está relacionada com $L(\alpha)$.

— Aprendizagem da regra através dos exemplos refere-se à capacidade do perceptron de reproduzir a função Booleana B e está relacionada com $G(\alpha)$.

³ Na verdade, a probabilidade de reconhecimento será menor, uma vez que o padrão terá correlação com vários vetores exemplos com classificações σ diferentes. Haverá interferência entre esses padrões, de modo que podemos esperar que, quanto maior α , menor será este efeito de tamanho finito da rede.

Poderíamos usar "memorização" para o primeiro sentido de "aprendizagem", porém na literatura o uso desta última é corrente. Assim, os dois sentidos deverão ser distinguidos pelo contexto quando necessário.

2.2 RECORDAÇÃO SERIADA DE LISTAS

Iremos introduzir uma outra medida de aprendizado que, no nosso conhecimento, nunca foi usada nos trabalhos com o perceptron. Esta medida é usada em experiências de recordação de listas de pares associados feitas em psicologia, e nos será útil na situação de aprendizagem com apresentação única dos exemplos. Com efeito, nesta situação o conjunto de aprendizagem $L = \{(\mathbf{s}^\mu, \mathbf{t}^\mu)\}$ pode ser considerado uma lista ordenada onde o índice μ denota a posição do exemplo (par associado) na lista.

Assim, chamaremos de "habilidade de recordação de lista" $R(\alpha', \alpha)$ a probabilidade de que, após a apresentação total da lista (com $P = \alpha N$ exemplos), o vetor exemplo $\mu = \alpha' N$ seja corretamente classificado. É importante notar a diferença entre α e α' , que será bastante usada nos próximos capítulos.

Evidentemente estamos considerando uma grandeza média, tomada sobre muitas listas L , mas que não deve ser confundida com a habilidade de aprendizagem $L(\alpha)$, onde também é feita a média sobre todos os exemplos de L , ou seja

$$L(\alpha) = \frac{1}{\alpha} \int_0^\alpha d\alpha' R(\alpha', \alpha) \quad (2.10)$$

Usaremos $R(\alpha', \alpha)$ para averiguar se a memorização de um certo exemplo depende de sua posição na lista L . Uma medida similar foi usada no contexto memorização de padrões aleatórios em redes de atratores [Bu 90].

2.3 EXEMPLO: REGRA DE HEBB

A regra de Hebb, extensamente considerada na área de redes de atratores [Ho 82, Am 85a], é um procedimento extremamente simples para se construir o vetor de acoplamentos $|J\rangle$ de modo a se correlacionar a saída do perceptron aluno com a saída dada pelo exemplo. A curva de habilidade de generalização $G(\alpha)$ para esta regra tem sido usada universalmente como base de comparação para se avaliar o desempenho de algoritmos de aprendizagem mais sofisticados. As curvas $G(\alpha)$ e $L(\alpha)$ foram calculadas inicialmente por Vallet [Vl 89a, Vl 90]. Iremos estender seus resultados para curvas $P_R(\alpha, m)$ com valores intermediários de m .

A regra de Hebb para o perceptron toma a forma

$$|J(\mu)\rangle = |J(\mu-1)\rangle + \frac{1}{N} \sigma_B^\mu |S^\mu\rangle \quad (2.11)$$

onde cada exemplo é apresentado uma única vez já que a função peso é uma constante. O padrão $|S\rangle$ será classificado corretamente se

$$\tilde{h} \cdot \text{sgn}(b) \equiv \langle J(\alpha N) | S \rangle \text{sgn}(\langle B | S \rangle) \geq 0 \quad (2.12)$$

Usando a forma de $|J(\alpha N)\rangle$ obtida a partir de (2.11) esta condição pode ser escrita na forma

$$\sum_{\mu=1}^{\alpha N} Z_\mu \geq 0 \quad (2.13)$$

onde

$$Z_\mu = \frac{1}{N} \text{sgn}(b) \text{sgn}(b_\mu) \langle S | S^\mu \rangle \quad (2.14)$$

Vamos supor que $|S\rangle$ seja uma versão com ruído de algum vetor do conjunto de treinamento, por exemplo $|S^1\rangle$, possuindo com este uma correlação $m = \langle S | S^1 \rangle / N$ (a escolha de $\mu = 1$ é irrelevante, qualquer dos exemplos poderia ser considerado como protótipo). Iremos calcular a probabilidade de reconhecimento para este vetor $|S\rangle$, $P_R^L(\alpha, m, |S\rangle)$, e depois fazer a média sobre os possíveis $|S\rangle$ e os possíveis conjuntos de treinamento L .

Temos que

$$Z_1 = m \operatorname{sgn}(b) \operatorname{sgn}(b_1) \quad (2.15)$$

e as restantes $\alpha N - 1$ variáveis Z_u serão aleatórias e estatisticamente independentes, com valor médio (Apêndice 2.A)

$$\langle Z_u \rangle = \frac{1}{N} \sqrt{\frac{2}{\pi}} |b| \quad (2.16)$$

e variância

$$\sigma^2 \simeq 1/N \quad (2.17)$$

Usamos agora um resultado do Teorema do Limite Central [Fe 68]: a probabilidade de que a soma de p variáveis aleatórias z_i independentes com valor médio $\langle z \rangle$ seja maior que um certo valor M é dada por

$$\operatorname{Prob} \left(\sum_{i=1}^p z_i \geq M \right) = \frac{1}{2} \operatorname{erfc} \left(\frac{M - p \langle z \rangle}{\sigma \sqrt{2p}} \right) \quad (2.18)$$

onde a função erro complementar é definida por

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty dt e^{-t^2} \quad (2.19)$$

Aplicando na condição (2.13) obtemos

$$\begin{aligned} P_R^L(\alpha, m, |S\rangle) &= \operatorname{Prob} \left(\sum_{u=2}^{\alpha N} Z_u \geq -Z_1 \right) \\ &= \frac{1}{2} \operatorname{erfc} \left(\frac{-mw}{\sqrt{2\alpha}} - \sqrt{\frac{2\alpha}{\pi}} \nu \right) \end{aligned} \quad (2.20)$$

onde

$$w = \operatorname{sgn}(b) \operatorname{sgn}(b_1), \quad \nu = \frac{1}{\sqrt{2}} |b| \quad (2.21)$$

A dependência no conjunto de treinamento L se dá apenas através da variável w . Note que se $m = 1$ teremos obrigatoriamente $w = 1$, enquanto que se $m = 0$ a parcela contendo w irá desaparecer de (2.20). Em ambos os casos não será necessário fazer a média sobre os possíveis conjuntos de treinamento L (é o que faz Vallet [V1 90] e outros [Op 90]). Porém, para m com valores intermediários será necessário fazer essa média. A variável w tem distribuição de probabilidade

$$P(w) = P(b.b_1 \geq 0) \delta(w-1) + P(b.b_1 < 0) \delta(w+1) \quad (2.22)$$

Temos que

$$P(b.b_1 \geq 0) = 2 \int_0^\infty \int_0^\infty db db_1 P(b,b_1) \quad (2.23)$$

onde a distribuição de probabilidade conjunta das variáveis gaussianas b e b_1 é dada por

$$\begin{aligned} P(b,b_1) &= P(b) P(b_1|b) \\ &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{b^2}{2}\right) \times \frac{1}{\sqrt{2\pi(1-m^2)}} \exp\left(-\frac{(b_1-mb)^2}{2(1-m^2)}\right) \end{aligned} \quad (2.24)$$

Um cálculo direto fornece

$$P(b.b_1 \geq 0) = 1 - \frac{1}{\pi} \arccos(m) \quad (2.25)$$

Agora podemos fazer a média sobre a variável w , obtendo

$$\begin{aligned} P_R(\alpha, m, |S\rangle) &= \left\langle P_R^L(\alpha, m, |S\rangle) \right\rangle_w \\ &= \left(\frac{1}{2} - \frac{1}{2\pi} \arccos(m) \right) \operatorname{erfc} \left(\frac{-m}{\sqrt{2\alpha}} - \sqrt{\frac{2\alpha}{\pi}} \nu \right) \\ &\quad + \frac{1}{2\pi} \arccos(m) \operatorname{erfc} \left(\frac{m}{\sqrt{2\alpha}} - \sqrt{\frac{2\alpha}{\pi}} \nu \right) \end{aligned} \quad (2.26)$$

Assim, a probabilidade de reconhecimento de um certo vetor $|S\rangle$ depende apenas do módulo de sua projeção no vetor $|B\rangle$ através da variável v , que possui distribuição de probabilidade

$$P(v) = \frac{2}{\sqrt{\pi}} e^{-v^2} \Theta(v) \quad (2.27)$$

Fazendo a média sobre os possíveis $|S\rangle$ temos finalmente

$$\begin{aligned} P_R(\alpha, m) &= \left(1 - \frac{1}{\pi} \arccos(m)\right) \int_0^\infty \frac{dv}{\sqrt{\pi}} e^{-v^2} \operatorname{erfc}\left(\frac{-m}{\sqrt{2\alpha}} - \sqrt{\frac{2\alpha}{\pi}} v\right) \\ &\quad + \frac{1}{\pi} \arccos(m) \int_0^\infty \frac{dv}{\sqrt{\pi}} e^{-v^2} \operatorname{erfc}\left(\frac{m}{\sqrt{2\alpha}} - \sqrt{\frac{2\alpha}{\pi}} v\right) \end{aligned} \quad (2.28)$$

Como esperado, $P_R(\alpha, -m) = P_R(\alpha, m)$, ou seja, o antipadrão $-|S^\mu\rangle$ é memorizado junto com o padrão $|S^\mu\rangle$.

Os casos em que $m = 0$ e $m = 1$ se reduzem às expressões encontradas por Vallet

$$\begin{aligned} G(\alpha) \equiv P_R(\alpha, 0) &= \int_0^\infty \frac{dv}{\sqrt{\pi}} e^{-v^2} \operatorname{erfc}\left(-\sqrt{\frac{2\alpha}{\pi}} v\right) \\ &= 1 - \frac{1}{\pi} \arccos\left(\left(1 + \frac{\pi}{2\alpha}\right)^{-1/2}\right) \end{aligned} \quad (2.29)$$

$$L(\alpha) \equiv P_R(\alpha, 1) = \int_0^\infty \frac{dv}{\sqrt{\pi}} e^{-v^2} \operatorname{erfc}\left(\frac{-1}{\sqrt{2\alpha}} - \sqrt{\frac{2\alpha}{\pi}} v\right) \quad (2.30)$$

Comparando (2.29) com (2.7) encontramos que

$$\rho(\alpha) = \left(1 + \frac{\pi}{2\alpha}\right)^{-1/2} \quad (2.31)$$

A figura (2.2) mostra algumas curvas teóricas para $0 \leq m \leq 1$ junto com resultados de simulações. Esta figura ilustra os dois tipos de "generalização" discutidos no capítulo 1. Para $m = 0$ temos apenas um processo de "extração de regra" puro, ou "generalização forte". Para $m > 0$, o perceptron apresenta também uma tendência de generalização no sentido fraco, do tipo "padrões semelhantes

devem ser classificados do mesmo modo". Como esta propriedade é parcialmente válida para o nosso problema (um vetor exemplo situado fora do hiperplano de separação, ou seja, com $b \neq 0$, sempre possui uma vizinhança de outros vetores com a mesma classificação), esse tipo de generalização fraca (ou "memorização generalizada") tem um efeito positivo no desempenho do perceptron, especialmente para valores pequenos de α . No caso do problema de memória associativa em redes de atratores (que está relacionado com o problema de mapeamento aleatório no perceptron) esta propriedade de generalização fraca estará ligada à existência das bacias de atração das memórias.

No limite $\alpha \rightarrow \infty$ os efeitos de memorização desaparecem e a probabilidade de reconhecimento independe de m ,

$$P_R(\alpha, m) \simeq 1 - \frac{1}{\sqrt{2\pi\alpha}} \quad (2.32)$$

Isso quer dizer que $G(\alpha) \simeq L(\alpha)$ assintoticamente. Este comportamento parece ser uma característica da situação com apresentação única dos exemplos e ocorre para outros algoritmos que iremos estudar. Com a apresentação repetida dos exemplos isto não ocorre necessariamente pois, para vários algoritmos (por exemplo, Perceptron ou Relaxação), podemos obter $L(\alpha) = 1$ para qualquer α .

Por (2.32) temos que o erro de generalização e_g decai assintoticamente com $\alpha^{-1/2}$. Veremos no próximo capítulo que algoritmos mais sofisticados (porém ainda com única apresentação dos exemplos) podem produzir um erro de generalização que decai com α^{-1} .

Finalmente, para este algoritmo temos $R(\alpha', \alpha) = L(\alpha)$, uma vez que a regra de Hebb dá o mesmo peso para cada exemplo não importando sua ordem de apresentação. Isto não será verdade para outros algoritmos.

Nota a respeito das simulações: Em todos os resultados apresentados nas figuras estaremos indicando o número de neurônios da rede por N e o número de redes simuladas por n (de modo que os resultados sempre são os valores médios calculados a partir dessas n medidas). Procuramos usar um número suficiente de neurônios para que houvesse pouca flutuação nos resultados. Em geral, o desvio padrão dos dados será da ordem do tamanho do símbolos usados nos figuras e as barras de erro foram evitadas para maior legibilidade. Assim, só apresentamos indicações de erro quando necessárias para efetuar alguma comparação de precisão.

FIGURA 2.1 — Representação bidimensional dos hiperplanos cujos vetores normais são $|B\rangle$ e $|J\rangle$. Vetores $|S^k\rangle$ situados no volume achurado possuem projeção sobre $|J\rangle$ com sinal diferente da projeção sobre $|B\rangle$, e portanto são mal classificados.

PROBABILIDADE DE RECONHECIMENTO X NUMERO DE EXEMPLOS

FIGURA 2.2 — Curvas de reconhecimento $P_R(\alpha, m)$. As linhas sólidas são obtidas pela integração numérica da equação (2.27) para os valores $m = 1.0, 0.98, 0.90, 0.80$ e 0.0 . Resultados de simulação ($N = 149, n = 10$): (o) $m = 0.98$, (o) $m = 0.90$ e (o) $m = 0.80$.

CAPITULO 3

PROBLEMA DE GENERALIZAÇÃO

COM APRESENTAÇÃO ÚNICA DOS EXEMPLOS

Neste capítulo iremos estudar algoritmos de aprendizagem que não usam o processo de iteração, ou seja, correção incremental de $|J\rangle$ baseada na apresentação repetida dos exemplos. Tecnicamente, isto é equivalente a estudar apenas o primeiro passo da dinâmica (sequencial) de algoritmos com iteração. Porém, como foi discutido na seção 1.4.4, as "situações de aprendizagem" não são equivalentes e estamos assumindo a apresentação única dos exemplos como um vínculo ou restrição que define uma situação de aprendizagem válida e interessante de se estudar.

Consideremos um algoritmo assíncrono onde cada novo exemplo é adicionado com um peso F_μ e um termo de normalização Ω_μ (cuja forma não precisa ser especificada agora) atua sobre as sinapses

$$|J(\mu)\rangle = |J(\mu-1)\rangle + \frac{F_\mu}{N} |S^\mu\rangle - \frac{\Omega_\mu}{N} |J(\mu-1)\rangle \quad (3.1)$$

Os exemplos são gerados com uma distribuição de probabilidade $P(|S^\mu\rangle)$ que, por enquanto, também não é necessário especificar.

O overlap típico $\rho(\mu)$ entre o perceptron aluno e o perceptron professor até o estágio μ será agora um funcional de F_μ (veremos que ρ não irá depender de Ω_μ). Lembremos que a probabilidade de generalização $G(\alpha)$ cresce monotonicamente com ρ . Iremos calcular esse funcional e encontrar a função F_μ que maximiza $\rho(\mu)$ em relação a $\rho(\mu-1)$, ou seja, iremos maximizar a "velocidade de aprendizagem da regra" $d\rho/d\alpha$, e portanto o ganho na probabilidade de generalização gerado pelo novo exemplo.

Mais tarde iremos exigir que o algoritmo satisfaça a condição de localidade e de caixa negra, ou seja, F_μ deverá ter a forma (ver seção 1.4.2)

$$F_{\mu} \equiv F(\sigma_B^{\mu}, \tilde{h}_{\mu}, \mu) \quad (3.2)$$

onde σ_B^{μ} é a saída do perceptron professor e $\tilde{h}_{\mu}$ é o campo local não normalizado definidos anteriormente.

Inicialmente, porém, vamos obter a solução do nosso problema variacional sem levar em conta esta restrição. Iremos encontrar uma solução na forma

$$F_{\mu}^* \equiv F(|S^{\mu}\rangle, |B\rangle, |J(\mu-1)\rangle) \quad (3.3)$$

e assim obter o valor teórico máximo ρ^* que o procedimento dado por (3.1) pode alcançar. Caso a solução não satisfaça a condição (3.2), vamos tentar aproximá-la levando em conta as condições de localidade e caixa negra.

3.1. SOLUÇÃO DO PROBLEMA DE GENERALIZAÇÃO SEM RESTRIÇÕES

O overlap ρ obtido até a adição do μ -ésimo exemplo é

$$\rho(\mu) = \frac{R(\mu)}{B J(\mu)} \quad (3.4)$$

onde

$$R(\mu) = \langle B | J(\mu) \rangle \quad (3.5)$$

$$J(\mu) = \sqrt{\langle J(\mu) | J(\mu) \rangle}, \quad B = \sqrt{\langle B | B \rangle} = 1 \quad (3.6)$$

Guardando termos até ordem N^{-1} temos

$$R(\mu) = \left(1 - \frac{\Omega_{\mu}}{N}\right) R(\mu-1) + \frac{1}{N} F_{\mu} b_{\mu} \quad (3.7)$$

$$J(\mu) = J(\mu-1) \left(1 - \frac{\Omega_{\mu}}{N} + \frac{1}{N} \left(\frac{F_{\mu} h_{\mu}}{J(\mu-1)} + \frac{F_{\mu}^2}{2 J^2(\mu-1)} \right) \right) \quad (3.8)$$

onde

$$h_{\mu} = \frac{\langle J(\mu-1) | S^{\mu} \rangle}{J(\mu-1)}, \quad b_{\mu} = \frac{\langle B | S^{\mu} \rangle}{B} \quad (3)$$

Desprezando termos de ordem N^{-2} temos que o overlap $\rho(\mu)$ será dado por

$$\rho(\mu) = \rho(\mu-1) + \frac{1}{N J(\mu-1)} \left((b_\mu - \rho(\mu-1) h_\mu) F_\mu - \frac{\rho(\mu-1) F_\mu^2}{2 J(\mu-1)} \right) \quad (3.10)$$

Note que os termos contendo Ω_μ se cancelaram. Isso reflete o fato de que este termo afeta apenas o módulo de $|J\rangle$ mas não sua direção, e portanto não pode ser usado para otimizar ρ mas apenas para limitar J .

Esta é uma equação estocástica de diferenças finitas, que chamaremos de "equação de aprendizagem". A partir desta equação iremos encontrar uma "equação de aprendizagem média" mais simples e que servirá bem (na verdade, surpreendentemente bem) para nossos propósitos.

Considerando que para N grande a grandeza $\rho(\mu-1)$ deverá depender apenas do número de exemplos já aprendidos (mas não de quais particulares exemplos foram vistos) iremos fazer a média apenas sobre as possíveis escolhas de $|S^\mu\rangle$, o que equivale a fazer a média sobre as variáveis b_μ e h_μ cuja probabilidade conjunta chamaremos de $P(b_\mu, h_\mu)$. Ficamos com

$$\rho(\mu) = \rho(\mu-1) + \frac{\rho(\mu-1)}{N J(\mu-1)} \int_{-\infty}^{\infty} dh_\mu db_\mu P(b_\mu, h_\mu) F_\mu \left(\frac{b_\mu}{\rho(\mu-1)} - h_\mu - \frac{F_\mu}{2 J(\mu-1)} \right) \quad (3.11)$$

No limite $N \rightarrow \infty$ obtemos uma equação diferencial para a evolução de ρ

$$\frac{d\rho}{d\alpha'} = \frac{1}{J} \int_{-\infty}^{\infty} dh_\mu db_\mu P(b_\mu, h_\mu) F_\mu \left(b_\mu - \rho h_\mu - \frac{\rho F_\mu}{2 J} \right) \quad (3.12)$$

Esta é nossa equação de aprendizagem média, válida para qualquer distribuição de probabilidades dos exemplos e para qualquer algoritmo (definido pela função F_μ).

Note que usamos a notação $\alpha' = \mu/N$, reservando a letra α apenas para designar o número total de exemplos de L , $P = \alpha N$. Pode parecer irrelevante fazer a distinção entre α e α' para a situação de apresentação única dos exemplos, mas a diferença será importante quando considerarmos o algoritmo iterado.

Queremos encontrar F_μ tal que o funcional em (3.12) seja maximizado. Usando considerações de cálculo variacional introduzimos funções variadas

$$F_{\mu} = F_{\mu}^* + \epsilon \eta(h_{\mu}, b_{\mu}) \quad (3.13)$$

onde F_{μ}^* é a solução desejada, ϵ é um parâmetro pequeno e $\eta(h_{\mu}, b_{\mu})$ é uma função arbitrária. Agora $\dot{\rho} \equiv d\rho/d\alpha'$ é uma função de ϵ e a condição para se obter um ponto estacionário é que a variação $\delta\dot{\rho}$ seja zero,

$$\delta\dot{\rho} = \left. \frac{d\dot{\rho}}{d\epsilon} \right|_{\epsilon=0} d\epsilon = 0 \quad (3.14)$$

Calculando a derivada em relação a ϵ no ponto $\epsilon = 0$ obtemos

$$\left. \frac{d\dot{\rho}}{d\epsilon} \right|_{\epsilon=0} = \frac{1}{J} \int_{-\infty}^{\infty} dh_{\mu} db_{\mu} P(b_{\mu}, h_{\mu}) \eta(h_{\mu}, b_{\mu}) \left(b_{\mu} - \rho h_{\mu} - \frac{\rho F_{\mu}^*}{J} \right) \quad (3.15)$$

e, dada a arbitrariedade em $\eta(h_{\mu}, b_{\mu})$, a condição de estacionaridade implica que

$$F_{\mu}^* = J \left(\frac{b_{\mu}}{\rho} - h_{\mu} \right) \quad (3.16)$$

Podemos escrever esta função peso na forma

$$F_{\mu}^* = \langle F^*(\mu) | S^{\mu} \rangle \quad (3.17)$$

com

$$| F^*(\mu) \rangle = \frac{J}{\rho} | B \rangle - | J(\mu-1) \rangle \quad (3.18)$$

em que o vetor $| F^*(\mu) \rangle$ é ortogonal a $| J(\mu-1) \rangle$

$$\langle J(\mu-1) | F^*(\mu) \rangle = 0 \quad (3.19)$$

A solução F_{μ}^* tem um sentido geométrico claro (figura 3.1): cada exemplo produz um incremento em $| J \rangle$ na direção de um vetor aleatório $| S^{\mu} \rangle$; se o peso do incremento for dado por F_{μ}^* com $| F^* \rangle$ perpendicular a $| J \rangle$ então cada incremento irá diminuir o ângulo entre $| J \rangle$ e $| B \rangle$. Se $| F^* \rangle$ não fosse ortogonal a $| J \rangle$, isto nem sempre iria acontecer.

Neste ponto não é visível que o algoritmo satisfaz o critério de localidade pois parece depender do módulo de $| J(\mu-1) \rangle$. Mas $J(\mu-1)$ e $\rho(\mu-1)$ podem ser aproximados por seus valores médios $J(\alpha')$ e $\rho(\alpha')$ para N grande.

No apêndice 3.A mostramos que se usarmos F_{μ}^* na equação de evolução para J podemos obter (para $\Omega_{\mu} = 0$, isto é, sem normalizar J)

$$J(\alpha') = C \rho(\alpha') \quad (3.20)$$

onde C é uma constante arbitrária. Sendo assim, F_{μ}^* toma a forma

$$F_{\mu}^* = C \langle B | S^{\mu} \rangle - \langle J(\mu-1) | S^{\mu} \rangle \quad (3.21)$$

Isto significa que podemos tomar C igual a 1 de forma autoconsistente, ou seja, se usarmos uma função peso F_{μ}^* assumindo $J/\rho = 1$, a evolução de $|J\rangle$ (que depende de F_{μ}^*) irá produzir realmente um vetor com módulo $J = \rho$.

Usando portanto $C = 1$, $|F^* \rangle$ se torna simplesmente o vetor diferença entre $|B\rangle$ e $|J\rangle$. Já que $|J\rangle$ é sempre perpendicular à $|F^* \rangle$, a trajetória de $|J\rangle$ se dará na superfície de uma hiperesfera (representada na figura 3.2), evoluindo com passos aleatórios (mas sempre progressivamente) do "pólo sul" até o "pólo norte" definido por $|B\rangle$.

FIGURA 3.1 — O vetor diferença $|F^* \rangle = |B \rangle - |J \rangle$ é perpendicular ao vetor $|J \rangle$. O ponto definido por $|J \rangle$ reside portanto em uma hipersfera. O incremento infinitesimal $1/N \langle F^* | S^{\mu} \rangle | S^{\mu} \rangle$, onde $| S^{\mu} \rangle$ é aleatório, terá sempre um sentido que diminui o ângulo entre o perceptron aluno e o professor.

FIGURA 3.2 — Ilustração de uma possível trajetória de $|J \rangle$ durante o processo de aprendizagem.

3.2 APRENDIZAGEM PASSIVA

Estudaremos a situação em que o perceptron recebe de uma fonte EXEMPLOS um conjunto de aprendizagem $\mathcal{L} = \{(|S^\mu\rangle, \sigma_B^\mu)\}$, $\mu = 1, \dots, \alpha N$, cujos vetores exemplos binários $|S^\mu\rangle \equiv \{S_1^\mu, \dots, S_N^\mu\}$ são gerados antes do início da aprendizagem de forma estatisticamente independente e com uma distribuição de probabilidade $P(|S^\mu\rangle)$ dada. Estamos chamando esta situação de aprendizagem passiva pois o estado ou desempenho do perceptron não influenciam a formação do conjunto $\mathcal{L}$ (ver [Va 84], [Wa 69] pg.255). Em particular, estudaremos o caso em que as variáveis S_j^μ são variáveis de Ising ± 1 escolhidas de forma independente com mesma probabilidade $p = 1/2$.

Se $P(|S^\mu\rangle)$ é uniforme, então b_μ e h_μ são variáveis gaussianas reduzidas (possuem média igual a zero e variância igual a 1) com probabilidade conjunta

$$P(b_\mu, h_\mu) = P(h_\mu) P(b_\mu | h_\mu) \\ = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{h_\mu^2}{2}\right) \times \frac{1}{\sqrt{2\pi(1-\rho^2)}} \exp\left(-\frac{(b_\mu - \rho h_\mu)^2}{2(1-\rho^2)}\right) \quad (3.22)$$

onde $\rho = \rho(\mu-1)$.

Usaremos nas próximas seções a notação

$$\langle \bullet \rangle_{b/h} = \frac{\int_{-\infty}^{\infty} db_\mu \bullet P(b_\mu, h_\mu)}{\int_{-\infty}^{\infty} db_\mu P(b_\mu, h_\mu)} = \int_{-\infty}^{\infty} \bullet \tilde{D}b_\mu \quad (3.23)$$

$$\langle \bullet \rangle_h = \int_{-\infty}^{\infty} \bullet Dh_\mu \quad (3.24)$$

onde

$$\tilde{D}b_\mu = \frac{db_\mu}{\sqrt{2\pi(1-\rho^2)}} \exp\left(-\frac{(b_\mu - \rho h_\mu)^2}{2(1-\rho^2)}\right) \quad (3.25)$$

$$Dh_\mu = \frac{dh_\mu}{\sqrt{2\pi}} \exp\left(-\frac{h_\mu^2}{2}\right) \quad (3.26)$$

3.2.1 — APRENDIZAGEM COM PERCEPTRON PROFESSOR LINEAR

Na seção 3.1 encontramos a solução ótima para a função peso F_μ .

$$F_\mu^* = \langle B | S^\mu \rangle - \langle J_{(\mu-1)} | S^\mu \rangle \quad (3.27)$$

Esta função tem a forma $F_\mu^* = F^*(b_\mu, \tilde{h}_\mu, \mu)$ e só satisfaz a condição de caixa negra no caso em que $\sigma_B^\mu = \langle B | S^\mu \rangle$, ou seja, para o problema em que o perceptron professor é linear. No caso do professor booleano onde $\sigma_B^\mu = \text{sgn}(\langle B | S^\mu \rangle)$ deveremos encontrar uma aproximação para F_μ que satisfaça essa condição, tendo a forma $F(\sigma_B^\mu, \tilde{h}_\mu, \mu)$.

Para maior clareza, listamos as quatro situações de aprendizagem possíveis:

- A. Perceptron professor linear e perceptron aluno linear.
- B. Perceptron professor linear e perceptron aluno booleano.
- C. Perceptron professor booleano e perceptron aluno linear.
- D. Perceptron professor booleano e perceptron aluno booleano.

Mais tarde veremos que diversas regras e algoritmos de aprendizagem (Hebb, Adaline, Perceptron, algoritmo de Relaxação etc.) usam aproximações de F^* não otimizadas para o problema de generalização no caso D. Iremos também propor a nossa aproximação, que será a solução ótima para o perceptron booleano.

Antes disso, porém, vamos estudar a evolução de ρ^* obtido por um algoritmo que usa F_μ^* (casos A e B). Introduzindo F_μ^* na equação de evolução para ρ (3.12) temos

$$\frac{d\rho}{d\alpha'} = \int_{-\infty}^{\infty} Dh_\mu \int_{-\infty}^{\infty} \tilde{D}b_\mu \left(\frac{b_\mu^2}{2\rho} - h_\mu b_\mu + \frac{\rho h_\mu^2}{2} \right) \quad (3.28)$$

Fazendo a integração obtemos a equação diferencial simples

$$\frac{d\rho}{d\alpha'} = \frac{1 - \rho^2}{2\rho} \quad (3.29)$$

cuja solução é

$$\rho = \sqrt{1 - e^{-\alpha'}} \quad (3.30)$$

A situação B é equivalente à situação D com violação da condição de caixa negra (ou seja, o aluno tem acesso ao campo interno b_μ do professor). Assim, iremos examinar o caso B pois ele nos fornece um limite superior (bound) teórico para a situação D. Como neste caso o erro de generalização é dado por $e_g = \frac{1}{\pi} \arccos(\rho)$, no limite $\alpha \rightarrow \infty$ teremos

$$\rho \simeq 1 - \frac{\pi^2 e_g^2}{2} \quad (3.31)$$

de onde obtemos o comportamento assintótico do erro de generalização

$$e_g \simeq \frac{1}{\pi} e^{-\alpha'/2} \quad (3.32)$$

Enfatizamos novamente que este é o erro de generalização de um perceptron aluno booleano que não obedece à condição de caixa negra. O erro de generalização do perceptron aluno linear (caso A) é definido de outra forma [He 91] [Se 92], usando-se a diferença entre as saídas (lineares) do perceptron aluno e professor, ou seja

$$\begin{aligned} e_g(\text{linear}) &= \int_{-\infty}^{\infty} \prod_{j=1}^N dS_j P(S_j) \frac{1}{2} (b-h)^2 = 1 - \rho \\ &= 1 - \sqrt{1 - e^{-\alpha'}} \end{aligned} \quad (3.33)$$

O algoritmo de aprendizagem pode ser obtido pela discretização da dinâmica

$$\frac{dJ_j}{d\alpha'} = - \frac{\partial E^*(\alpha')}{\partial J_j} \quad (3.34)$$

onde a função custo ou "energia de aprendizagem" $E^*(\mu)$, no estágio $\mu = \alpha'N$, é dada por

$$E^*(\mu) = \frac{1}{2} \left(\langle B | S^\mu \rangle - \langle J_{(\mu-1)} | S^\mu \rangle \right)^2 \quad (3.35)$$

Este é simplesmente o algoritmo Adaline sequencial para o perceptron linear [Wi 60]. O problema de generalização assim colocado consiste simplesmente em encontrar a solução $\{ J_j \}$ de αN equações lineares

$$\sum_{j=1}^N s_j^{\mu} J_j = b_{\mu} \quad (\mu = 1 \dots \alpha N) \quad (3.36)$$

onde os coeficientes são as componentes dos vetores exemplos.

A convergência exponencial em direção à solução ($\rho=1$) se deve ao fato de que cada exemplo é usado apenas uma vez. No algoritmo com iteração (capítulo 4) obteremos $\rho = 1$ (transição para perfeita generalização) para $\alpha \geq 1$, pois bastam N equações independentes para tornar o sistema (3.36) determinado [He90] [Kr92b]. Assim, concluímos que, **para a situação com única apresentação dos exemplos não existe transição de fase para perfeita generalização.**

Devemos notar que só foi possível definir a energia E^* porque, para o primeira apresentação dos exemplos, temos $\rho(\alpha) = J(\alpha)$. Se tivermos $J \neq \rho$ (como irá acontecer com o algoritmo iterado), então só será possível escrever uma equação do tipo (3.34) para a variável normalizada $\hat{J}_j = J_j/J$ (confira mais abaixo).

A fim de fazer a comparação com outros algoritmos de aprendizagem, vamos escrever o algoritmo definido por F_{μ}^* na forma de uma regra de Hebb com peso

$$|J(\mu)\rangle = |J(\mu-1)\rangle + J \frac{w_{\mu}^*}{N} \sigma_B^{\mu} |S^{\mu}\rangle \quad (3.37)$$

onde $\sigma_B^{\mu} = \text{sgn}(b_{\mu})$ é a saída do perceptron professor booleano (e não a do perceptron linear). A nova função peso w_{μ}^* será dada por (note que extraímos o fator J da função peso)

$$w_{\mu}^* = \frac{1}{J} \sigma_B^{\mu} F_{\mu}^* = (K_{\mu} - \Delta_{\mu}) \quad (3.38)$$

que escrevemos em termos da "estabilidade local"

$$\Delta_{\mu} = h_{\mu} \sigma_B^{\mu} = \frac{\langle J | S^{\mu} \rangle \sigma_B^{\mu}}{\sqrt{\langle J | J \rangle}} \quad (3.39)$$

e do "parâmetro de Gardner"

$$K_{\mu} = b_{\mu} \sigma_B^{\mu} / \rho \quad (3.40)$$

que é sempre positivo já que $\sigma_{\mu} = \text{sgn}(b_{\mu})$.

Aqui, ao contrário de um parâmetro K constante conforme introduzido por Gardner [Ga 87], o parâmetro K_μ depende de μ , já que no problema de generalização os exemplos não são simétricos em relação ao vetor $|B\rangle$. O parâmetro K_μ tem uma interpretação clara: está relacionado com a estabilidade desejada para o exemplo μ . Uma situação similar, onde K_μ é dado por uma distribuição arbitrária $P(K_\mu)$, foi estudada por Abbott e Kepler [Ab 89b] no problema de mapeamento aleatório.

Esta mudança de variáveis poderia ter sido evitada se tivéssemos desde o início partido da forma "Hebbiana" do algoritmo de aprendizado, eq. (3.37). Não fizemos isto visando mostrar que o aprendizado "Hebbiano" não necessita ser postulado como ponto de partida nem implica em alguma perda de generalidade. Mostramos também que é importante incluir o fator J no algoritmo, já que o que interessa não é a mudança no valor absoluto de J_j , mas sim da grandeza relativa J_j/J .

A equação de evolução para ρ para qualquer algoritmo do tipo Hebbiano, que é um dos resultados principais deste capítulo, assume a forma

$$\frac{d\rho}{d\alpha'} = \rho \int_{-\infty}^{\infty} dh_\mu db_\mu P(b_\mu, h_\mu) W_\mu \left(K_\mu - \Delta_\mu - \frac{W_\mu}{2} \right) \quad (3.41)$$

Aqui existe um ponto sutil. Uma mudança de variáveis do tipo (3.38) é escrita de forma um pouco mais rigorosa

$$W_\mu = \frac{\sigma_B^\mu F_\mu}{J(F_\mu)} \quad (3.42)$$

onde o módulo do vetor $|J\rangle$ é uma função do peso F_μ usado no algoritmo. Assim, a transformação

$$F_\mu' = C F_\mu \quad (3.43)$$

onde C é uma constante deixa W_μ invariante porque $J(F_\mu') = C J(F_\mu)$. Chamaremos W_μ de peso invariante do algoritmo. Agora fica claro porque a equação (3.41) é invariante pela transformação (3.43).

A equação (3.37) pode agora ser escrita, no limite $N \rightarrow \infty$, como uma equação diferencial para a variável normalizada $\hat{J}_j = J_j/J$

$$\frac{d\hat{J}_j}{d\alpha'} = - \frac{\partial \hat{E}}{\partial \hat{J}_j} \quad (3.44)$$

A energia de aprendizagem para o perceptron linear será agora

$$\hat{E}(\mu) = \frac{1}{2} (k_\mu - \Delta_\mu)^2 \quad (3.45)$$

3.2.2 PERCEPTRON PROFESSOR BOOLEANO: ALGORÍTMO DE PESO SIMÉTRICO

Na seção anterior encontramos como solução uma regra de Hebb com peso w_μ^* que produz generalização ótima mas que só pode ser usada no caso do professor linear pois exige o conhecimento de b_μ . Nosso problema então fica claramente definido: devemos encontrar a melhor aproximação de

$$w_\mu^* = \sigma_B^\mu \left(\frac{b_\mu}{\rho} - h_\mu \right) \quad (3.46)$$

que satisfaça a condição de caixa negra.

Uma aproximação possível [KC 91] (porém não ótima) é fazer a média sobre a variável b_μ dado um certo campo local h_μ .

$$\bar{w}(\mu, h_\mu) \equiv \langle w_\mu^* \rangle_{b/h} = \int_{-\infty}^{\infty} db_\mu \frac{w_\mu^*}{\sqrt{2\pi(1-\rho^2)}} \exp\left(-\frac{(b_\mu - \rho h_\mu)^2}{2(1-\rho^2)}\right) \quad (3.47)$$

onde usamos a notação

$$\langle \bullet \rangle_{b/h} \equiv \frac{\int_{-\infty}^{\infty} db_\mu \bullet P(b_\mu, h_\mu)}{\int_{-\infty}^{\infty} db_\mu P(b_\mu, h_\mu)} = \int_{-\infty}^{\infty} \tilde{D}b_\mu \bullet \quad (3.48)$$

Fazendo a integração em b_μ temos

$$\langle |b_\mu| \rangle_{b/h} = \sqrt{\frac{2}{\pi}} \rho \lambda \exp\left(-\frac{h_\mu^2}{2\lambda^2}\right) + \rho h_\mu \operatorname{erf}\left(\frac{h_\mu}{\sqrt{2} \lambda}\right) \quad (3.49)$$

$$\left\langle \operatorname{sgn}(b_\mu) \right\rangle_{b/h} = \operatorname{erf}\left(\frac{h_\mu}{\sqrt{2} \lambda}\right) \quad (3.50)$$

onde a função erro é definida por

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x dt e^{-t^2} \quad (3.51)$$

e usamos a notação simplificada

$$\rho \equiv \rho(\mu-1) \quad , \quad \lambda \equiv \lambda(\mu) = \frac{\sqrt{1-\rho^2}}{\rho} \quad (3.52)$$

Substituindo em (3.47) vem

$$\bar{W}(\mu, h_\mu) = \sqrt{\frac{2}{\pi}} \lambda \exp\left(-\frac{h_\mu^2}{2\lambda^2}\right) \quad (3.53)$$

Introduzindo essa solução na equação de evolução para ρ obtemos

$$\frac{d\rho}{d\alpha'} = -\frac{\rho \lambda^2}{\pi} \int_{-\infty}^{\infty} Dh_\mu e^{-h_\mu^2/\lambda^2} \quad (3.54)$$

ou, fazendo a integral,

$$\frac{d\rho}{d\alpha'} = -\frac{(1-\rho^2)^{3/2}}{\pi \rho \sqrt{1+\rho^2}} \quad (3.55)$$

Por separação de variáveis obtemos

$$\int \frac{x^2}{\sqrt{(2-x^2)^3}} dx = -\frac{x}{\pi} + C \quad (3.56)$$

onde fizemos a mudança de variáveis $x^2 = 1 + \rho^2$.

Integrando obtemos

$$\frac{x}{\sqrt{(2-x^2)}} - \operatorname{arcsen}\left(\frac{x}{\sqrt{2}}\right) = \frac{\alpha}{\pi} + 1 - \frac{\pi}{4} \quad (3.57)$$

onde a constante C foi achada usando-se a condição $\rho = 0$ para $\alpha = 0$. A solução para essa equação transcendente foi calculada numericamente e é mostrada na figura (3.3) junto com os resultados de simulação.

No limite $\rho \rightarrow 1$ (ou $\alpha \rightarrow \infty$) temos $\operatorname{arcsen}(x/\sqrt{2}) \simeq \pi/2$, e

$$\rho \simeq \left(1 + \frac{2}{(\pi/4 + 1 + \alpha/\pi)^2 - 1} \right)^{-1/2} \quad (3.58)$$

de onde obtemos o comportamento assintótico do erro de generalização

$$e_g \simeq \frac{\sqrt{2}}{\alpha} \quad (3.59)$$

Vemos então que, embora já não tenhamos o comportamento exponencial, este algoritmo (que não é ótimo) na forma de uma regra de Hebb com peso muda o comportamento assintótico $e_g \simeq \alpha^{-1/2}$ característico da solução de Hebb.

Precisamos ainda achar uma forma explícita para os fatores J e $\bar{W}(\mu, h_\mu)$, uma vez que pela condição de localidade o algoritmo não deve depender de J (a função $\bar{W}$ está dependendo de J através de h_μ). Podemos encontrar uma equação diferencial para J procedendo de forma análoga à que fizemos para ρ (Apêndice 3.A)

$$\frac{dJ}{d\alpha'} = J \left(2\rho + 1/\rho \right) \frac{d\rho}{d\alpha'} \quad (3.60)$$

Fazendo a integração obtemos

$$J = C e^{-(1-\rho^2)} \quad (3.61)$$

onde a constante de integração foi escolhida tal que $J = 1$ no limite $\rho \rightarrow 1$.

ALGORÍTIMO SIMÉTRICO: $G(\alpha)$ E $L(\alpha)$

FIGURA 3.3 — Algoritmo Simétrico: a curva sólida é o valor de $G(\alpha)$ obtido a partir da integração numérica da equação (3.56). Resultados de simulação ($N = 99$, $n = 50$): (o) $G(\alpha)$ e (●) $L(\alpha)$.

FUNÇÃO PESO $W/J \times \Delta$

FIGURA 3.4 — Função peso $W_S(\alpha, \Delta_\mu)$ do algoritmo Simétrico. De cima para baixo: $\alpha = 0.5, 1.0, 2.0$ e 4.0 .

Assim, o peso usado no algoritmo toma a forma local

$$\bar{w} = J(\alpha') \bar{w}(\alpha', \tilde{h}_\mu) = \sqrt{\frac{2}{\pi}} \mathbb{D} \exp\left(-\frac{\tilde{h}_\mu^2}{2 \mathbb{D}^2} \exp(-\mathbb{D}^2)\right) \exp(-\mathbb{D}^2) \quad (3.62)$$

onde $\rho(\alpha')$ e $\mathbb{D}(\alpha') = \sqrt{1 - \rho^2}$ são os valores teóricos obtidos através da equação (3.57). Assim, a condição de apresentação única dos exemplos permite encontrar um algoritmo que respeita totalmente a condição de localidade.

A forma de $\bar{w}(\alpha', \Delta)$ é mostrada na figura (3.4) e possui a característica interessante de que o mesmo peso é dado tanto a exemplos classificados corretamente ($\Delta > 0$) como os mal classificados ($\Delta < 0$). O desempenho do perceptron para aquele específico exemplo não influencia seu peso de aprendizagem. Chamaremos de "algoritmo de peso simétrico" este método de aprendizagem definido por $\bar{w}(\alpha', |\Delta|)$, que daqui por diante será referido como w_s .

3.2.3 SOLUÇÃO DE ÓTIMA GENERALIZAÇÃO COM PROFESSOR BOOLEANO

Na seção anterior discutimos a aproximação

$$w_s(\alpha', h_\mu) = \frac{\int_{-\infty}^{\infty} db_\mu P(b_\mu, h_\mu) w^*(\alpha', b_\mu, h_\mu)}{\int_{-\infty}^{\infty} db_\mu P(b_\mu, h_\mu)} \quad (3.63)$$

que produziu resultados analíticos bastante simples. Mas esta aproximação não é ótima pois não usa o conhecimento que temos de σ_B^μ .

A **aproximação ótima** de w_μ^* é o valor esperado dessa grandeza para h_μ e σ_B^μ fixos, ou seja, devemos considerar w_μ^* na forma

$$w_\mu^* = \frac{1}{\rho} (C_\mu - \rho \sigma_B^\mu h_\mu) \quad (3.64)$$

onde usamos $C_\mu \equiv \sigma_B^\mu b_\mu$, e calcular

$$\begin{aligned}\bar{W}(\rho, \sigma_B^\mu, h_\mu) &= \frac{\int_{-\infty}^{\infty} dC_\mu P(C_\mu, \sigma_B^\mu, h_\mu) W_\mu^*}{\int_{-\infty}^{\infty} dC_\mu P(C_\mu, \sigma_B^\mu, h_\mu)} \\ &= \langle k_\mu \rangle_{C|\Delta - \Delta_\mu}\end{aligned}\quad (3.65)$$

onde $\langle \bullet \rangle_{C|h, \sigma}$ denota a média em C dado h e σ_B^μ . A distribuição de probabilidade

$$\begin{aligned}P(C_\mu, \sigma_B^\mu, h_\mu) &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{h_\mu^2}{2}\right) \times \\ &\times \frac{1}{\sqrt{2\pi(1-\rho^2)}} \exp\left(-\frac{(C_\mu - \rho \sigma_B^\mu h_\mu)^2}{2(1-\rho^2)}\right) \Theta(C_\mu)\end{aligned}\quad (3.66)$$

está normalizada ($\Theta(x)$ é a função de Heaviside)

$$\sum_{\sigma_B^\mu = \pm 1} \int_{-\infty}^{\infty} dh_\mu \int_{-\infty}^{\infty} dC_\mu P(\sigma_B^\mu, h_\mu, C_\mu) = 1 \quad (3.67)$$

Temos que

$$\begin{aligned}P(\sigma_B^\mu, h_\mu) &= \int_{-\infty}^{\infty} dC_\mu P(\sigma_B^\mu, h_\mu, C_\mu) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{h_\mu^2}{2}\right) \times H\left(-\frac{\sigma_B^\mu h_\mu}{\lambda}\right) \\ &= P(h_\mu) \times P(\sigma_B^\mu | h_\mu)\end{aligned}\quad (3.68)$$

onde

$$H(x) \equiv \int_x^\infty \frac{du}{\sqrt{2\pi}} e^{-u^2/2} = \frac{1}{2} - \frac{1}{2} \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right) \quad (3.69)$$

Temos também

$$\int_{-\infty}^{\infty} dC_\mu C_\mu P(\sigma_B^\mu, h_\mu, C_\mu) = P(h_\mu) \left(\frac{\rho \lambda e^{-h_\mu^2/2\lambda^2}}{\sqrt{2\pi}} + \sigma_B^\mu \rho h_\mu H\left(-\frac{\sigma_B^\mu h_\mu}{\lambda}\right) \right) \quad (3.70)$$

de modo que

$$\bar{W}(\rho, \sigma_B^\mu, h_\mu) = \frac{1}{\sqrt{2\pi}} \lambda \exp\left(-\frac{(h_\mu/\lambda)^2}{2}\right) \left(H\left(-\sigma_B^\mu h_\mu/\lambda\right) \right)^{-1} \quad (3.71)$$

onde $\lambda(\rho)$ foi definido em (3.52).

Introduzindo a expressão para $\bar{W}(\rho, \sigma_B^\mu, h_\mu)$ na equação diferencial (3.41) para ρ temos

$$\begin{aligned} \frac{d\rho}{d\alpha'} &= \rho \sum_{\sigma_B^\mu = \pm 1} \int_{-\infty}^{\infty} dh_\mu \frac{\bar{W}^2}{2} P(h_\mu) H(-\sigma_B^\mu h_\mu / \lambda) \\ &= \frac{1-\rho^2}{4\pi \rho} \sum_{\sigma_B^\mu = \pm 1} \int_{-\infty}^{\infty} Dh_\mu e^{-h_\mu^2 / \lambda^2} \frac{1}{H(-\sigma_B^\mu h_\mu / \lambda)} \\ &= \frac{\rho \lambda^3}{2\pi} \int_{-\infty}^{\infty} Du e^{-(1+\lambda^2)u^2/2} \frac{1}{H(u)} \end{aligned} \quad (3.72)$$

O valor de $\rho(\alpha)$ calculado numericamente a partir dessa equação foi comparado com os resultados de simulação na figura (3.5).

No limite $\alpha \rightarrow \infty$ ($\lambda \rightarrow 0$) temos

$$\frac{d\rho}{d\alpha'} = \frac{\rho \lambda^3}{2\pi} I$$

onde

$$I = \int_{-\infty}^{\infty} Du \frac{e^{-u^2/2}}{H(u)} \quad (3.73)$$

O erro de generalização para $\alpha \rightarrow \infty$ será

$$e_g \simeq \frac{2}{I\alpha} \simeq \frac{0.882}{\alpha} \quad (3.74)$$

que é exatamente duas vezes o erro mínimo teórico para aprendizagem iterada (com repetição dos exemplos) dado pelo algoritmo de Bayes de Oppen e Haussler, ver capítulo 4, seção 4.2.1.

Notando que a função peso depende apenas da estabilidade local $\Delta_\mu \equiv \sigma_B^\mu h_\mu$ e não de σ_B^μ e h_μ separadamente, escrevemos

$$\bar{W}(\rho, \Delta_\mu) = \frac{1}{\sqrt{2\pi}} \lambda \exp\left(-\frac{\Delta_\mu^2}{2}\right) \left(H(-\Delta_\mu/\lambda)\right)^{-1} \quad (3.75)$$

A forma desta função peso para diversos valores de α' é mostrada na figura (3.6). Chamaremos de algoritmo de estabilidade esperada a regra de Hebb que usa este peso $\bar{W}(\rho, \Delta_\mu)$.

Nesta forma, o algoritmo não é local pois depende de $J(\mu)$ e também viola a condição de caixa negra pois requer o conhecimento de $\rho(\mu)$. Mas é importante reconhecer nesta altura que a curva teórica $\rho(\alpha)$ obtida representa um limite teórico para o desempenho do primeiro passo da dinâmica (sequencial ou paralela) de qualquer algoritmo de aprendizagem no caso do professor booleano.

Vamos a seguir mostrar como tornar o algoritmo local e evitar a dependência em $\rho(\mu)$. Neste caso de apresentação única dos exemplos pode-se mostrar que, para o nosso algoritmo, a evolução do módulo de $\langle J \rangle$ satisfaz $J(\alpha') = \rho(\alpha')$ como no caso do perceptron linear (Apêndice 3.A). Assim, a função peso a ser usada na prática usa duas aproximações:

a) Em vez de se medir o valor particular $\rho(\mu)$ usa-se o valor médio $\rho(\alpha')$.

b) Aproxima-se $J(\mu)$ por seu valor médio $J(\alpha')$ e usa-se a equivalência $J(\alpha') = \rho(\alpha')$ para se simplificar a expressão da função peso.

O resultado é

$$\bar{\varphi} = J(\alpha') \bar{W}(\alpha', \tilde{\Delta}_\mu) = \frac{\mathcal{D}}{\sqrt{2\pi}} \exp\left(-\frac{\tilde{\Delta}_\mu^2}{2\mathcal{D}^2}\right) \left(H\left(\frac{-\tilde{\Delta}_\mu}{\mathcal{D}}\right)\right)^{-1} \quad (3.76)$$

onde $\tilde{\Delta}_\mu = \sigma_B^\mu \langle J | S^\mu \rangle$ é a estabilidade local não-normalizada e

$$\mathcal{D}(\alpha') = \sqrt{1 - \rho^2(\alpha')} \quad (3.77)$$

Notemos ainda que

$$\mathcal{D} = \lim_{\rho \rightarrow 1} \sin(\pi e_g) \cong \pi e_g \quad (3.78)$$

de forma que, se tivéssemos acesso ao erro de generalização $e_g(\alpha')$, não haveria necessidade de usar a expressão teórica para $\rho(\alpha')$. O uso do erro de generalização como parâmetro de aprendizagem em vez de $\rho(\alpha')$ tem certo apelo biológico e será discutido mais tardes (seção 3.4).

ALGORÍTMO ESTAB. ESP.: $G(\alpha)$ E $L(\alpha)$

FIGURA 3.5 — Algoritmo de Estabilidade Esperada a curva sólida é o valor de $G(\alpha)$ obtido a partir da integração numérica da equação (3.71). Resultados de simulação ($N = 99, n = 50$) (○), $G(\alpha)$ e (●), $L(\alpha)$

FUNÇÃO PESO $\bar{W}/J \times \Delta$

FIGURA 3.6 — Função peso $\bar{W}(\alpha, \Delta_\mu)$ do algoritmo de Estabilidade Esperada De cima para baixo: $\alpha = 0.5, 1.0, 2.0, 4.0$ e $\alpha \rightarrow \infty$

3.3 APRENDIZAGEM ATIVA OU SELEÇÃO DE EXEMPLOS

Consideremos agora a segunda situação de aprendizagem definida por Valiant [Va 84] onde o estudante pode obter da fonte ORÁCULO a resposta correta para exemplos escolhidos durante a aprendizagem, ou seja, onde existe o controle sobre $P(h_\mu)$. Essa situação de aprendizagem no caso do perceptron foi estudada inicialmente por Kinzel e Ruján [Ki 90] para a regra de Hebb simples.

A equação de evolução de ρ pode ser escrita para uma densidade de probabilidade dos exemplos $P(h_\mu)$ arbitrária

$$\frac{d\rho}{d\alpha'} = \frac{1-\rho^2}{2\rho} \int_{-\infty}^{\infty} dh_\mu P(h_\mu) I(h_\mu/\lambda) \quad (3.79)$$

onde

$$I(h_\mu/\lambda) = \rho^2 \int_{-\infty}^{\infty} \tilde{D}b_\mu \frac{W_\mu}{1-\rho^2} (2W_\mu^* - W_\mu) \quad (3.80)$$

dependerá da função W_μ usada para aproximar a função peso do perceptron linear W_μ^* . A velocidade de aprendizagem da regra $d\rho/d\alpha'$ terá valor máximo quando a integral em h_μ também for maximizada. Isto pode ser obtido usando-se $P(h_\mu)$ igual a uma função delta de Dirac centrada no valor de h_μ que maximiza $I(h_\mu/\lambda)$.

Iremos assumir que o perceptron ou seu treinador possuem a capacidade de selecionar os exemplos segundo essa função delta. Discutiremos como fazer isso mais tarde quando estivermos apresentando as simulações realizadas.

3.3.1 PERCEPTRON PROFESSOR LINEAR

Se $W_\mu = W_\mu^*$ então substituindo em (3.80) vem

$$I(h_\mu/\lambda) = \rho^2 \int_{-\infty}^{\infty} \tilde{D}b_\mu \frac{(W_\mu^*)^2}{1-\rho^2} = \int_{-\infty}^{\infty} \tilde{D}b_\mu \frac{(b_\mu - \rho h_\mu)^2}{1-\rho^2} = 1 \quad (3.81)$$

Isso significa que não existe vantagem alguma em selecionar exemplos para o problema com professor linear de modo que sua curva de generalização fica inalterada. Isto é óbvio quando lembramos que o problema de generalização com professor linear é equivalente a um simples sistema de equações lineares. Cada exemplo possui tanta informação quanto qualquer outro para aumentar o grau de determinação do sistema.

3.3.2 PERCEPTRON PROFESSOR BOOLEANO

No caso do algoritmo de estabilidade esperada para o professor booleano temos

$$I(h_\mu/\lambda) = \frac{1}{2\pi} e^{-(h_\mu/\lambda)^2} \left(\frac{1}{H(h_\mu/\lambda)} + \frac{1}{H(-h_\mu/\lambda)} \right) \quad (3.82)$$

A função $I(h_\mu/\lambda)$ é uma função par com valor máximo $2/\pi$ em $h_\mu = 0$ (ver figura (3.7)), e portanto a melhor estratégia de seleção consiste em escolher cada novo exemplo ortogonal ao vetor $|J\rangle$ construído com os exemplos anteriores. Com efeito esta é a escolha efetuada, mas justificada apenas de forma heurística, por Kinzel e Ruján.

Usando a distribuição de probabilidade dos exemplos $P(h_\mu) = \delta(h_\mu)$, temos

$$\frac{d\rho}{d\alpha'} = \frac{1-\rho^2}{\pi\rho} \quad (3.83)$$

cujas solução é

$$\rho(\alpha) = \sqrt{1 - e^{-2\alpha/\pi}} \quad (3.84)$$

O comportamento assintótico do erro de generalização neste caso é exponencial

$$e_g \simeq \frac{1}{\pi} e^{-\alpha/\pi} \quad (3.85)$$

Kinzel e Ruján usam o peso $\psi_\mu = 1$ em seu algoritmo (regra de Hebb clássica). No nosso caso a função peso depende da etapa do processo de aprendizagem (parametrizada por α')

$$\bar{\psi}(\alpha') = J \bar{W} = \sqrt{\frac{2}{\pi}} J \lambda = \sqrt{\frac{2}{\pi}} e^{-\alpha'/\pi} \quad (3.86)$$

onde usamos o fato de que também neste caso vale $J(\alpha') = \rho(\alpha')$ (apêndice 3.A).

Na figura (3.8) comparamos as curvas de generalização da regra de Hebb e do nosso algoritmo nesta situação com seleção de exemplos.

Para fazer a simulação, partimos do vetor inicial $|J(1)\rangle = |S^1\rangle$ escolhido aleatoriamente. Nas etapas seguintes, determinamos um vetor binário $|S^\mu\rangle$ que produz um campo local $\tilde{h}_\mu$ próximo de zero. Isso foi feito por um método de Monte Carlo simples: mudamos sequencialmente as componentes de um vetor binário aleatório sempre que isso diminui a "energia" $E = |\tilde{h}_\mu|$, ou seja, usamos um processo à "temperatura" zero. Não foi feita nenhuma tentativa de otimizar esse procedimento.

Os resultados da simulação, apresentados na figura (3.9), foram bastante satisfatórios. Entretanto, podemos observar efeitos de tamanho finito da rede (ou seja, que desaparecem para $N \rightarrow \infty$) mas que não se referem ao modo como medimos $G(\alpha)$ (conforme discutido na seção 2.1) já que tendem a aumentar com α . Estes efeitos merecem um maior estudo.

PERCEPTRON LINEAR, EST.ESP. E HEBB: SELEÇÃO DE EXEMPLOS

FIGURA 3.8 — Curvas teóricas para situação de aprendizagem com seleção de exemplos (sólido) Perceptron "linear" eq (3.29), (traços) Estabilidade Esperada eq (3.81) e (pontos) Hebb, segundo [K1 90]. As duas curvas inferiores foram colocadas para comparação e se referem a situação sem seleção de exemplos Estabilidade Esperada (sólido) e Hebb (traços longos).

GENERALIZAÇÃO X NÚMERO DE EXEMPLOS

FIGURA 3.9 — Aprendizagem com seleção de exemplos curvas teóricas para algoritmos de (sólido) Estabilidade Esperada e (traços) Hebb. Resultados de simulação ($n = 20$). ($\circ$) $N = 99$ e ($\square$) $N = 199$.

FIGURA 3.7 — Função $I(h/\lambda)$ definida pela equação (3.79) com $\rho = 0.9$.

3.4 — APRENDIZAGEM AUTO-REGULADA PELO ERRO DE GENERALIZAÇÃO

A segunda maneira de se eliminar a dependência em $\rho(\alpha')$ consiste em fazer o algoritmo depender do erro de generalização, usando a equação (3.78). Isto pode ser feito aproximando-se o erro de generalização e_g pelo desempenho médio $\hat{e}_g$ nos últimos $\Delta\alpha$ exemplos aprendidos. A largura $\Delta\alpha$ é um parâmetro a ser otimizado. A função peso a ser usada no algoritmo assume a forma

$$\bar{w}(\hat{e}_g, \Delta\mu) = \frac{\sin(\pi\hat{e}_g)}{\sqrt{2\pi}} \exp\left(-\frac{\tilde{\Delta}_\mu^2}{2\sin^2(\pi\hat{e}_g)}\right) \left(H\left(\frac{-\tilde{\Delta}_\mu}{\sin(\pi\hat{e}_g)}\right)\right)^{-1} \quad (3.87)$$

Em nossa simulação procedemos do seguinte modo:

— Começamos com $|J\rangle = \sigma_B^1 |S^1\rangle$. Para $\nu = 2, \dots, \Delta\alpha$, de forma sequencial, a variável de erro

$$\epsilon_\nu = \Theta(-\sigma_B^\nu \sigma_J^\nu) \quad (3.88)$$

onde Θ é a função de Heaviside, é guardada e o exemplo ν é aprendido usando-se um valor estimado $\hat{e}_g = 1/2$ na expressão para $\bar{w}$. Note que a variável ϵ_ν é medida antes que o exemplo seja usado para modificar as sinapses.

— Quando foram aprendidos $\Delta\alpha$ exemplos, faz-se uma nova estimativa do erro de generalização através de

$$\hat{e}_g \simeq \frac{1}{\Delta\alpha N} \sum_{\nu=1}^{\Delta\alpha N} \epsilon_\nu \quad (3.89)$$

— Os próximos $\Delta\alpha$ exemplos serão aprendidos usando-se esta nova estimativa de e_g na função peso, quando então uma nova estimativa de e_g será feita, e assim por diante. Note que a avaliação de e_g deverá ser feita usando-se apenas as últimas $\Delta\alpha$ variáveis ϵ_ν .

No algoritmo acima a avaliação de e_g se faz a cada bloco de $\Delta\alpha N$ exemplos. Podemos pensar em um método mais exato (e mais plausível biologicamente) que usaria, no estágio μ , um erro estimado $\hat{e}_g(\mu)$ obtido por uma média ponderada sobre as últimas p variáveis ϵ_ν . Ou seja, o perceptron deveria "lembrar" dos últimos $p = \Delta\alpha N$ acertos ou erros cometidos com um peso $\Omega(\nu)$ ($\nu = \mu - p, \dots, \mu$),

$$\hat{e}_g(\mu) = \frac{1}{p} \sum_{\nu=\mu-p}^{\mu} \Omega(\nu) \epsilon_\nu \quad (3.90)$$

A forma da função Ω também é um parâmetro a ser otimizado. Fizemos a simulação usando-se a avaliação de e_g em blocos apenas por economia de esforço computacional. Não esperamos grande diferença entre os dois métodos se $\Delta\alpha$ for pequeno.

Os resultados da simulação mostram que o método simples da avaliação em blocos é eficiente, aproximando o desempenho teórico previsto (figura (3.10)), porém sem reproduzi-lo exatamente. Foram testadas "janelas" com $\Delta\alpha$ entre 0.1 e 2.0 para avaliar e_g .

Na figura (3.11) mostramos a avaliação $\hat{e}_g$ do erro de generalização feita pelas diversas janelas. Como é de se esperar, uma janela pequena apresenta muitas flutuações na avaliação de e_g , enquanto que uma janela maior, embora apresente menos flutuações, fornece uma estimativa conservadora de e_g . Para o método usado nas simulações, a janela ótima é $\Delta\alpha \sim 0.25 \alpha$.

O princípio novo (em teoria de perceptron) que estamos tentando enfatizar é o reconhecimento de que, embora a rede tenha acesso apenas à sua saída σ_j^H , seu erro de generalização e_g também é um observável que pode ser usado no processo de aprendizagem. Para isto basta imaginar um mecanismo simples de memória. O princípio de medição de e_g é similar ao de outras máquinas cibernéticas [Ar 87] que, por exemplo, tendo acesso apenas à variável X que mede uma posição, obtém informação sobre a velocidade através de medidas repetidas de X a curto intervalo de tempo usando algum sistema de memória⁴.

O que o algoritmo de estabilidade esperada fornece é a forma funcional com que e_g entra no processo de aprendizagem.

⁴ Um curioso exemplo biológico desse tipo de procedimento ocorre em bactérias capazes de quimiotaxia [McN72]. Segundo parece, a minúscula célula bacteriana, em vez de simplesmente determinar se há mais moléculas em uma extremidade do que em outra, mede o gradiente de concentração de uma substância comparando a concentração em dois momentos distintos, enquanto se move numa certa direção. Como sublinharam estes autores, essa comparação deve envolver uma memória a curto prazo.

APRENDIZAGEM AUTO-REGULADA

FIGURA 3.10 — Aprendizagem auto-regulada: efeito do tamanho da janela de memória $\Delta\alpha$ no desempenho do algoritmo. Resultados de simulação ($N = 149$, $n = 30$): (Δ) $\Delta\alpha = 2.0$, ($\circ$) $\Delta\alpha = 0.5$ e ($\square$) $\Delta\alpha = 0.25$, (sólido) teórico estabilidade esperada.

GENERALIZAÇÃO ESTIMADA

FIGURA 3.11 — Aprendizagem auto-regulada: efeito do tamanho de janela de memória na avaliação do parâmetro $\langle G(\alpha) \rangle = 1 - \langle e_g \rangle$. Resultados de simulação ($N = 149$, $n = 30$): ($\square$) $\Delta\alpha = 1.0$, ($\circ$) $\Delta\alpha = 0.25$, (sólido) teórico estabilidade esperada.

3.5 COMPARAÇÃO COM OUTROS ALGORÍTMOS

Podemos agora fazer a comparação do algoritmo com professor linear

$$w^* = J W^*(K_\mu \Delta_\mu) = J (K_\mu - \Delta_\mu) \quad (3.91)$$

onde $K_\mu = |b_\mu|/0$, e do algoritmo de estabilidade esperada

$$\begin{aligned} \bar{w} &= J \bar{W}(\alpha', \Delta_\mu) = J \left(\langle K_\mu \rangle_{\sigma|h\sigma} - \Delta_\mu \right) \\ &= J \frac{1}{\sqrt{2\pi}} N(\alpha') \exp\left(-\frac{(\Delta_\mu/\lambda(\alpha'))^2}{2}\right) \left(H\left(-\Delta_\mu/\lambda(\alpha')\right) \right)^{-1} \end{aligned} \quad (3.92)$$

com outros métodos de aprendizagem. Uma revisão feita por Abbott [Ab 90] descreve os mais conhecidos. Estudaremos apenas o caso sem seleção de exemplos. Lembramos que nosso algoritmo apresenta erro assintótico $e_g \simeq 0.88/\alpha = 2 e_g(\text{Bayes})$.

Partindo da forma geral

$$|J(\mu)\rangle = |J(\mu-1)\rangle + J \frac{W_\mu}{N} \sigma_S^\mu |S^\mu\rangle \quad (3.93)$$

definimos as seguintes classes de algoritmos (lembremos que o desempenho do algoritmo depende apenas de W e não da função peso completa $\mathcal{W} = J(W)$):

$$\text{Hebb: } \mathcal{W}_H(J, \alpha') = J W_H(\alpha') \quad (3.94)$$

Nesta classe o peso W depende apenas de α' , e todos os exemplos produzem modificação em $|J\rangle$.

Se $W_H(\alpha') = 1/J(\alpha')$ (ou seja, $\mathcal{W} = 1$) temos a regra de Hebb clássica, inspirada na proposta de Hebb [Hb 48], e por vezes chamada de prescrição de Cooper [Co 73]. Muito investigada no campo das redes de atratores (corresponde ao modelo de Hopfield).

Verificamos que (apêndice 3B) se $W_H = \text{constante}$ o algoritmo estaciona para algum $\rho < 1$. Encontramos que a escolha $W_H = 1/J$ (regra de Hebb clássica) é ótima, isto é, produz assintoticamente o menor erro de generalização: $e_g \simeq 1/\sqrt{2\pi\alpha}$.

$$\text{Adaline: } \mathcal{W}_A(J, \alpha', \Delta_\mu) = J W_A(\alpha', \Delta_\mu) = J \eta(\alpha') (K(\alpha') - \Delta_\mu) \quad (3.95)$$

Nesta classe o peso W depende de α' e Δ_μ , e todos os exemplos produzem modificação em $|J\rangle$.

Com $K = C/J$ é um algoritmo local, isto é, com função peso independente de J , e foi proposto originalmente por Widrow e Hoff (com $C = 1$, $\eta = \text{const.}$) [Wi 60]. Mostra-se [Di 87] que o vetor de acoplamentos relaxa exponencialmente para a solução do projetor ou "pseudoinversa" [Pe 85]. A dinâmica deste algoritmo foi estudada recentemente para o professor linear [He 89] e professor booleano [Ki 89].

Estudamos apenas o caso em que $\eta = \text{const.}$ (apêndice 3C). Neste caso existe um $\kappa(\alpha')$ ótimo dado por $K = C/\rho(\alpha')$. Uma vez que neste caso $J(\alpha') = \rho(\alpha')$, concluímos que o algoritmo Adaline clássico é ótimo neste tipo de algoritmo. Seu desempenho em generalização, porém, é pobre: o algoritmo estaciona em $\rho = \sqrt{2/\pi}$.

$$\text{Perceptron: } \mathcal{W}_p(J, \alpha', \Delta_\mu) = J \mathcal{W}_p(\alpha', \Delta_\mu) = J \eta(\alpha') \Theta(\kappa(\alpha') - \Delta_\mu) \quad (3.96)$$

Nesta classe apenas os exemplos com estabilidade menor que certo limiar K produzem modificação em $|J|$. A contribuição η dessas exemplos, porém, não depende da estabilidade do exemplo.

Se $K = C/J(\alpha')$ e $\eta = C'/J(\alpha')$, o fator J presente em Δ_μ pode ser eliminado e o peso toma a forma local $\mathcal{W}_p(\tilde{\Delta}_\mu) = \Theta(C - \tilde{\Delta}_\mu)$. Com $C = 0$, foi proposto por Rosenblatt (algoritmo perceptron clássico) [Ro 62], que demonstrou seu teorema de convergência. Com $C \neq 0$, o algoritmo converge para um vetor de acoplamentos satisfazendo $\langle J | S^\mu \rangle > C$ [Di 87].

Se a correção é feita usando apenas o exemplo com menor estabilidade (algoritmo Minover), mostra-se que no limite $C \rightarrow \infty$ é possível alcançar o perceptron de ótima estabilidade no mapeamento aleatório. A versão com $K = \text{const.} > 0$ foi estudada por Gardner [Ga 88a] e não é local na medida em que depende de J através de Δ_μ .⁵

Estudamos apenas o caso com $K = \text{constante}$ (apêndice 3D). Encontramos que, se K for diferente de zero, o algoritmo estaciona em $\rho < 1$. E mesmo que K seja nulo, se η for constante o algoritmo também estaciona em $\rho < 1$. Porém se $\eta(\alpha') = \left(\frac{\pi}{2}(1 - \rho^2(\alpha'))\right)^{1/2}$ então o algoritmo atinge $e_3 \simeq 1.27/\alpha$ para α grande.

⁵ Uma observação de interesse histórico (que no nosso conhecimento ainda não foi feita na literatura): este algoritmo com $K > 0$, chamado às vezes de algoritmo Edinburgh, foi proposto pela primeira vez em 1966 por Lambert e Watanabe [Wa 69 pg.442] sendo chamado por seus autores de método DZM (dead zone maximization).

$$\begin{aligned}
 \text{Relaxação: } W_R(J, \alpha', \Delta_u) &= J W_P(\alpha', \Delta_u) \\
 &= J \eta(\alpha') (K(\alpha') - \Delta_u) \Theta(K(\alpha') - \Delta_u)
 \end{aligned} \tag{3.97}$$

Com $K = \text{constante}$, foi proposto por Abbott e Kepler [Ab 89a] [Ab90] [Gr91], sendo o algoritmo com menor tempo de convergência para o caso de mapeamento aleatório. Com $K(\alpha') = C/J(\alpha')$, $\eta(\alpha') = C'/J(\alpha')$ torna-se local, $W_R = (C - \tilde{\Delta}_u) \Theta(C - \tilde{\Delta}_u)$ (é chamado de Adatron [An 89]) e mostra-se que o vetor de acoplamentos converge para o perceptron de ótima estabilidade.

Estudamos apenas o caso com K constante. Verificamos que, se $K \neq 0$ então o algoritmo estaciona para algum $\rho < 1$. Se $K = 0$, porém, o algoritmo aprende a regra se η pertencer ao intervalo $]0,3[$. O menor erro assintótico é produzido com $\eta = 1.5$ e vale $e_\eta \simeq 1.33/\alpha$.

Comentários:

Notamos em primeiro lugar que, ao contrário do caso de mapeamento aleatório (como no problema de autoassociação em redes de atratores), aqui não há sentido em se desejar máxima estabilidade K , uma vez que as estabilidades que queremos obter no final do processo de aprendizagem, $K_u = |b_u|$ ($\rho \rightarrow 1$), possuem uma distribuição Gaussiana truncada em zero

$$P(K_u) = \sqrt{\frac{2}{\pi}} \exp\left(-\frac{(K_u)^2}{2}\right) \Theta(K_u) \tag{3.98}$$

Assim, já podemos concluir que o perceptron de ótima estabilidade não deverá apresentar ótima generalização. O algoritmo de ótima estabilidade acabará dando aos exemplos situados perto do hiperplano de separação definido por $|B\rangle$, que possuem $K_u \simeq 0$, uma estabilidade K maior do que a necessária.

E por isso que os algoritmos Perceptron e Relaxação com $K=0$ sempre têm melhor desempenho. É interessante observar que, na aprendizagem iterada (repetição dos exemplos), o erro assintótico é $e_\eta(K=0) = 0.62/\alpha$ para ambos os algoritmos e vai com $1/\sqrt{\alpha}$ se K for diferente de zero.

Em segundo lugar, observamos que o algoritmo de estabilidade esperada depende de ρ e J . Para o primeiro ciclo de apresentação dos exemplos, já mostramos que essa dependência pode ser eliminada através da expressão analítica para $\rho(\alpha')$ e $J(\alpha')$. Mas o mesmo não é possível para os ciclos seguintes. Veremos no capítulo 4 como solucionar esse problema.

Propomos agora interpretar as diversas funções peso que definem os algoritmos conhecidos como sendo obtidas por aproximações ou médias feitas em K_μ e Δ_μ na expressão do algoritmo ideal $W^*(\rho, b_\mu, h_\mu)$. Usando a notação

$$\langle f \rangle_{x/y,z,\dots} = \frac{\int_{-\infty}^{\infty} dx f P(x, y, z, \dots)}{\int_{-\infty}^{\infty} dx P(x, y, z, \dots)}$$

para expressar o valor esperado de f dado $y, z, \dots$, as aproximações a serem feitas são listadas a seguir:

- A1. Usar o valor esperado de $f(C_\mu)$, onde $C_\mu \equiv |b_\mu|$, dado σ_B^μ e h_μ : $\langle f \rangle_{C/\sigma, h}$.
- A2. Usar o valor esperado de $f(\sigma_B^\mu)$ dado h_μ : $\langle f \rangle_{\sigma|h}$.
- A3. Usar o valor esperado de $f(h_\mu)$: $\langle f \rangle_h$.
- A4. Eliminar a dependência em α' de algum modo, por exemplo aproximar $\rho(\alpha')$ por uma constante r .
- A5. Eliminar a dependência em J_μ aproximando-o por $J(\alpha')$.

Tentaremos mostrar que aplicando-se estas aproximações em ordem diferente ou sobre apenas algum dos termos de W^* obtêm-se todos os algoritmos citados (e mais alguns novos que não tivemos tempo de estudar). Assim, teremos:

Algoritmo de estabilidade esperada: faz-se a aproximação A1 em W^* .

$$\bar{W}(\alpha', \Delta_\mu) = \langle K_\mu \rangle_{C/\sigma, h} - \Delta_\mu = \frac{1}{\sqrt{2\pi}} \lambda \exp\left(-\frac{(\Delta_\mu/\lambda)^2}{2}\right) \left(H(-\Delta_\mu/\lambda)\right)^{-1}$$

Algoritmo de estabilidade esperada independente de $\rho(\alpha)$: aplica-se A4 no algoritmo acima.

$$\bar{W}(r, \Delta_\mu) = \frac{1}{\sqrt{2\pi}} \lambda_r \exp\left(-\frac{(\Delta_\mu/\lambda_r)^2}{2}\right) \left(H(-\Delta_\mu/\lambda_r)\right)^{-1}$$

onde $\lambda_r = (1-r^2)^{1/2}/r$. Iremos considerar este algoritmo no próximo capítulo.

Algoritmo de peso simétrico: aplica-se a aproximação A2 no algoritmo de estabilidade esperada.

$$w_s(\alpha', h_\mu) = \left\langle \bar{w}(\rho, \sigma_\mu^u, h_\mu) \right\rangle_{\sigma/n} = \sqrt{\frac{2}{\pi}} \lambda \exp\left(-\frac{h_\mu^2}{2\lambda^2}\right)$$

Algoritmo de peso simétrico independente de $\rho(\alpha)$: aplica-se A4 no algoritmo acima.

$$w_s(r, h_\mu) = \sqrt{\frac{2}{\pi}} \lambda r \exp\left(-\frac{(h_\mu/\lambda r)^2}{2}\right)$$

Hebb ótimo (clássico): Aplica-se A3 ao algoritmo de peso simétrico.

$$w_H(\alpha') = \left\langle w_s(\alpha', h_\mu) \right\rangle_n = \sqrt{\frac{2}{\pi}} \frac{1-\rho^2}{\rho} = \frac{1}{J(\alpha')}$$

Classe Hebb (w_H const.): Aplica-se A4 ao algoritmo acima.

$$w_H = \text{constante}$$

Adaline ótimo (clássico): A2 e A3 é aplicado apenas ao primeiro termo no algoritmo de estabilidade esperada.

$$w_A(\alpha', \tilde{\Delta}_\mu) = \left\langle \left\langle \left\langle k_\mu \right\rangle_{\sigma/\sigma, n} \right\rangle_{\sigma/n} \right\rangle_n - \Delta_\mu = \sqrt{\frac{2}{\pi}} \frac{1}{\rho} - \Delta_\mu = \frac{1}{J(\alpha')} \left(\sqrt{\frac{2}{\pi}} - \tilde{\Delta}_\mu \right)$$

Classe Adaline (k const.): Aplica-se A4 ao algoritmo adaline ótimo.

$$w_A(k_r, \Delta_\mu) = k_r - \Delta_\mu, \quad k_r = \sqrt{\frac{2}{\pi}} \frac{1}{r}$$

Algoritmo de Relaxação: Aplica-se A4 ($r=1$) no algoritmo de estabilidade esperada independente de α' . Para ver isso examinamos o limite $\rho \rightarrow 1$ para o nosso algoritmo. Neste caso temos $\lambda \rightarrow \infty$ e devemos tomar um cuidado especial isolando o caso com estabilidade nula

$$\Delta_\mu > 0 : \quad H(-\Delta_\mu/\lambda) \rightarrow 1 \quad (3.99)$$

$$\Delta_\mu < 0 : \quad H(-\Delta_\mu/\lambda) \rightarrow \frac{-\lambda}{2\sqrt{2\pi} \Delta_\mu} \exp\left(-\frac{(\Delta_\mu/\lambda)^2}{2}\right) \quad (3.100)$$

$$\Delta_\mu = 0 : \quad H(-\Delta_\mu/\lambda) = \frac{1}{2} \quad (3.101)$$

A função peso, fica na forma

$$\bar{w}(\Delta u) = -2 \Delta u \Theta(-\Delta u) \quad (3.102)$$

que é exatamente a função peso do algoritmo de relaxação para $k = 0$ e $\eta = 2$.

Algoritmo Perceptron: Aplica-se A3 apenas na variável $|h_u|$ no algoritmo de relaxação.

$$w_p(\sigma_B^u \sigma_J^u) = -2 \sqrt{\frac{2}{\pi}} \sigma_B^u \sigma_J^u \Theta(-\sigma_B^u \sigma_J^u) = C \Theta(-\sigma_B^u \sigma_J^u)$$

onde C é uma constante positiva.

Comentários: Para α' finito ($\rho < 1$), o algoritmo de relaxação pode ser pensado como uma aproximação linear (truncada pela função Θ) para o algoritmo de estabilidade esperada. Neste caso, o parâmetro k do algoritmo de relaxação deverá ser maior que zero. Isso pode ser visto na figura (3.12), onde mostramos a função peso do algoritmo de estabilidade esperada para um certo valor de α' e uma aproximação feita ajustando-se os parâmetros η e k . Assim, um algoritmo de relaxação com η e k dependentes de α' imita a função peso $\bar{w}(\alpha', \Delta u)$.

Assim, podemos organizar os algoritmos conhecidos em uma árvore hierárquica (figura (3.13)) agrupando-os em famílias pelo modo como as aproximações foram feitas. Os algoritmos que usam $\rho(\alpha')$ podem ser implementados usando-se a expressão teórica para $\rho(\alpha')$ ou então usando-se uma estimativa de $\rho(\alpha)$ (aprendizagem auto-regulada). As versões locais são obtidas usando-se A5. Devemos observar que o modo de aplicar as aproximações A (ou outros tipos de aproximações não colocados naquela lista) podem gerar novos algoritmos.

Nas figuras (3.14a) e (3.14b) comparamos o desempenho de nossos algoritmos com o dos algoritmos Perceptron, Adaline e Hebb. As curvas para os algoritmos Adaline e Adaline ótimo são mostradas nas figuras (3.15a) e (3.15b).

Nas figuras (3.16a) e (3.16b) foram colocadas as curvas de generalização e aprendizagem do algoritmo de relaxação para vários valores de k . Mostramos a curva com parâmetro η ótimo na fig.(3.17a) e na figura (3.17b) mostramos os efeitos de tamanho finito da rede. Estes efeitos são observáveis em todas as simulações e estão de acordo com o que foi discutido no capítulo 2, seção 2.1.

Na figura (3.18) apresentamos curvas da classe Perceptron e finalmente na figura (3.19) comparamos os algoritmos perceptron e relaxação ótimos. Detalhes das simulações estão incluídos nas figuras.

FIGURA 3.13 — A "árvore de algoritmos" tem como raiz a função peso do perceptron linear $\psi^b = J \left(\frac{|b_\mu|}{\rho} - \sigma_E^\mu \sigma_J^\mu |h_\mu| \right)$. Obtêm-se os demais algoritmos fazendo-se as aproximações:

- A1. Média sobre a variável $|b_\mu|$ dado σ_E^μ e h_μ .
- A2. Média sobre a variável b_μ dado h_μ .
- A3. Média sobre a variável $|h_\mu|$.
- A4. Eliminação da dependência em α' ($\rho(\alpha') \rightarrow r$). (A4' : $\gamma \rightarrow 1$)
- A5. Eliminação da dependência em J .

O índice a ou b significa que a operação R é aplicada apenas no primeiro ou no segundo termo de ψ^b .

ALGORÍTMOS ADALINE, PERCEPTRON, HEBB: $G(\alpha)$

FIGURA 3.14 a — Desempenho em generalização $G(\alpha)$ na situação com única apresentação dos exemplos. Resultados de simulação ($N = 149$, $n = 30$): ($\square$) Adaline ($\eta = 1$, $K=1$), (∇) Perceptron ($\eta = 1$, $K=0$), (pontos) Hebb clássico ($W_H = 1/J$) teórico, (traços) Peso simétrico teórico, (sólido) Estabilidade Esperada teórico.

ALGORÍTMOS ADALINE, PERCEPTRON, HEBB: $L(\alpha)$

FIGURA 3.14 b — Desempenho em aprendizagem $L(\alpha)$ na situação com única apresentação dos exemplos. Resultados de simulação ($N = 149$, $n = 30$): ($\square$) Adaline ($\eta = 1$, $K=1$), (∇) Perceptron ($\eta = 1$, $K=0$), (pontos) Hebb clássico ($W_H = 1/J$) teórico, (o) Peso simétrico, (Δ) Estabilidade Esperada.

ALGORÍTMO ADALINE ÓTIMO: $G(\alpha)$

FIGURA 3.15 a — Algoritmo Adaline Desempenho em generalização $G(\alpha)$ na situação com única apresentação dos exemplos. Resultados de simulação ($N = 149$, $n = 30$): ($\square$) $k = 1.0$ e ($\circ$) Adaline ótimo (clássico) $k = 1/\sqrt{1-e^{-\alpha}}$. As curvas teóricas para o algoritmo com $k=1.0$ (pontos) e Adaline clássico (traços) foram calculadas no apêndice 3C. A curva para o algoritmo de Estabilidade Esperada (sólido) foi colocada para comparação.

ALGORÍTMO ADALINE ÓTIMO: $L(\alpha)$

FIGURA 3.15 b — Algoritmo Adaline Desempenho em aprendizagem $L(\alpha)$ na situação com única apresentação dos exemplos. Resultados de simulação ($N = 149$, $n = 30$): ($\square$) $k = 1.0$ e ($\circ$) Adaline ótimo (clássico) $k = 1/\sqrt{1-e^{-\alpha}}$. A curva para o algoritmo de (Δ) Estabilidade Esperada foi colocada para comparação.

ALGORÍTMO DE RELAXAÇÃO: $G(\alpha)$

FIGURA 3.16 a — Algoritmo de Relaxação ($\eta = 1$). Desempenho em generalização $G(\alpha)$ na situação com única apresentação dos exemplos. Resultados de simulação ($N = 145$, $n = 30$): ($\star$) $k = 0$, ($\square$) $k = 0.5$ (∇) $k = 1.0$. As curvas para os algoritmos de (traços) Peso Simétrico e (sólido) Estabilidade Esperada foram colocadas para comparação.

ALGORÍTMO DE RELAXAÇÃO: $L(\alpha)$

FIGURA 3.16 b — Algoritmo de Relaxação ($\eta = 1$). Desempenho em aprendizagem $L(\alpha)$ na situação com única apresentação dos exemplos. Resultados de simulação ($N = 145$, $n = 30$): ($\star$) $k = 0$, ($\square$) $k = 0.5$ (∇) $k = 1.0$. As curvas para os algoritmos de (o) Peso Simétrico e (Δ) Estabilidade Esperada foram colocadas para comparação.

ALGORÍTMO RELAXAÇÃO : $C(\alpha)$

FIGURA 3.17 a — Algoritmo de Relaxação ($k = 0$) Efeito do parâmetro η . Resultados de simulação ($n = 60$): (o) ótimo $\eta = 1.5$, (---) teórico $\eta = 1.5$, (Δ) $\eta = 1$, (.....) teórico $\eta = 1$. As curvas teóricas foram calculadas no apêndice 3E. A curva para o algoritmo de Estabilidade Eperadda (sólido) foi colocada para comparação.

ALGORÍTMO RELAXAÇÃO (N finito): $C(\alpha)$

FIGURA 3.17 b — Algoritmo de Relaxação Ótimo ($\eta = 1.5$). Efeitos de tamanho finito da rede. Resultados de simulação ($n = 60$): (Δ) $N = 19$, (o) $N = 49$, (∇) $K = 99$, (sólido) teórico.

ALGORÍTMO PERCEPTRON: $C(\alpha)$

FIGURA 3.18 — Algoritmo Perceptron ($k = 0$). Efeito do parâmetro η . Desempenho em generalização $G(\alpha)$ na situação com única apresentação dos exemplos. Resultados de simulação ($N = 99$, $n = 60$): (o) $\eta(\alpha)$ ótimo, (Δ) $\eta = 1$, (traços) teórico para $\eta(\alpha)$ ótimo (apêndice 3D). A curva para o algoritmo de Estabilidade Esperada (sólido) foi colocada para comparação.

ALGORÍTMOS PERCEPTRON E RELAXAÇÃO ÓTIMOS: $C(\alpha)$

FIGURA 3.19 — Algoritmos Perceptron e Relaxação ótimos. Desempenho em generalização $G(\alpha)$ na situação com única apresentação dos exemplos. Resultados de simulação ($N = 99$, $n = 60$): (o) Perceptron ótimo, (Δ) Relaxação ótimo, (-----) Perceptron teórico, (---) Relaxação teórico. A curva para o algoritmo de Estabilidade Esperada (sólido) foi colocada para comparação.

3.6 ANALOGIAS BIOLÓGICAS

Nesta seção iremos fazer algumas observações e apresentar certos resultados preliminares sobre os tópicos de "recordação seriada de listas" e "mudanças da regra durante o aprendizado".

Recordação seriada de listas:

A simples inclusão de um peso dependente do número de exemplos produziu um aumento considerável no desempenho de generalização do perceptron. A forma da função peso é muito sugestiva. A estabilidade esperada dos primeiros exemplos é avaliada de maneira bem mais precária do que a dos exemplos posteriores, que estão em um regime onde $|J|$ já está próximo de $|B|$. Mesmo assim, os primeiros exemplos são usados com maior peso na construção do vetor sináptico. Ao contrário da aprendizagem hebbiana simples, o peso de um exemplo depende do tempo (estágio μ) de aprendizagem.

Um problema interessante a ser estudado seria verificar de que modo essa assimetria temporal do algoritmo afeta a probabilidade de recordação $R(\alpha', \alpha)$ dos exemplos em função de sua localização no conjunto ou "lista" de aprendizagem L (ver na seção 2.2 a definição de $R(\alpha', \alpha)$, que não deve ser confundida com a probabilidade de aprendizagem de toda a lista $L(\alpha)$). Não tivemos oportunidade de estudar de forma adequada o assunto. Apresentamos apenas alguns dados (figura 3.20) para ilustrar este ponto. Podemos constatar que $R(\alpha', \alpha)$ não é constante: em geral os últimos exemplos são melhor lembrados, porém na aprendizagem ativa os primeiros exemplos também são bem recordados. É necessário maior estudo para entender a forma apresentada por estas curvas.

Devemos observar que estes efeitos, que lembram certos aspectos psicológicos de recordação de listas [Bu 90], não podem ser estudados no contexto de algoritmos iterados. O uso repetido do mesmo exemplo no processo de correção acaba por destruir qualquer vestígio da ordem temporal de apresentação dos exemplos da lista. Este é mais um motivo que justifica nossa ênfase na apresentação única dos exemplos.

O perceptron como modelo de um organismo simples

Para finalizar este capítulo, vamos fazer uma analogia biológica a fim de ilustrar as características da função peso do algoritmo de estabilidade esperada e do método de aprendizagem auto-regulada através do erro de generalização.

FIGURA 3.20 — Probabilidade de Recordação de Lista $R(\alpha', \alpha)$ em função da posição relativa do exemplo α'/α na lista. Os pontos representam médias feitas sobre blocos de $\Delta\alpha = 1$ exemplos sucessivos. Resultados de simulação ($N = 149$, $n = 10$): (□) distribuição uniforme dos exemplos ($\alpha = 10$) e (○) com seleção de exemplos ($\alpha = 5$).

Consideremos o perceptron como sendo o "sistema de decisão" de um organismo que enfrenta problemas $\{S^\mu\}$ aos quais deve dar uma resposta adequada σ^μ_E (que inclusive pode ser binária como "lutar ou fugir", "comer tal alimento ou não" etc.). A resposta adequada é obtida pela experiência, que faz o papel do "oráculo" ou "professor": se a resposta do organismo frente ao objeto $\{S^\mu\}$ que parecia apetitoso foi comê-lo, e isto acabou provocando uma indigestão, a própria fisiologia do organismo lhe serviu como "professor" indicando qual deveria ser a resposta correta frente a $\{S^\mu\}$.

Nosso organismo-perceptron usa a função peso do algoritmo de estabilidade esperada. Podemos pensar que isso acontece porque a população de organismos-perceptrons foi anteriormente submetida a um processo de seleção Darwiniana que otimizou o método de aprendizagem, favorecendo aqueles que produzissem melhor capacidade de generalização.

O sucesso ou fracasso da resposta para um problema ("exemplo") que está sendo experimentado tem um efeito importante na aprendizagem desse exemplo. Denotemos o erro no exemplo que está sendo aprendido pela variável

$$\epsilon_\mu = \Theta(-\sigma^\mu_E \sigma^\mu_J) \quad (3.103)$$

que vale 1 se houve erro de classificação no exemplo μ e zero caso contrário.

A recordação do sucesso ou fracasso da resposta para as experiências mais recentes também é guardada pelo organismo. Chamemos esse conjunto de recordações recentes de $\mathcal{R}$. O erro de generalização e_g pode ser avaliado como sendo o erro médio cometido nesses exemplos recentes,

$$e_g \simeq \langle \epsilon \rangle_{\mathcal{R}} \quad (3.104)$$

Por outro lado o campo local h_μ mede a ambiguidade do exemplo. Se o módulo de h_μ é pequeno, tomar uma decisão baseada nesse campo é sempre arriscado, pois qualquer ruído nas sinapses ou no exemplo $\{S^\mu\}$ poderia produzir variação na saída σ^μ_J do neurônio de decisão. Ou, no caso de um perceptron com saída analógica, o próprio valor de σ^μ_J seria pequeno, indicando pouca "confiança" na resposta.

Obviamente, se o desempenho do organismo tem sido satisfatório (e_g é muito pequeno), então um campo h_μ pequeno não causaria muita incerteza, pois o

organismo "confiará" mais no sinal emitido por σ_j^u . Assim, o "parâmetro de confiança" C deve ser a razão entre uma função do módulo do campo local e uma função do erro de generalização. No nosso caso será a função

$$C_u = \frac{|h_u|}{\text{tg}(\pi e_g)} \quad \frac{1}{2} \leq e_g \leq 1 \quad (3.105)$$

Pensemos agora no peso W como sendo o efeito de um mecanismo de "atenção". Quanto maior a "atenção", maior o peso dado ao exemplo que está sendo aprendido. Assim, o nível de atenção (peso W) dado ao exemplo por nosso organismo-perceptron será devido a três fatores:

$$W(e_g, \epsilon_u, C_u) = W_1(e_g) \cdot W_2(C) \cdot W_3(\epsilon_u, C_u)$$

que podemos chamar de:

Fator de desempenho: $W_1(e_g) = J \text{tg}(\pi e_g)$

É um prefator que não depende da dificuldade dos exemplos. Se o desempenho do organismo tem sido ruim ele dará mais atenção aos exemplos em geral, sejam fáceis ou difíceis, a fim de aprender melhor a regra. Se o desempenho tem sido bom, as novas experiências pouco afetarão o organismo.

Fator de confiança: $W_2(C_u) = \sqrt{\frac{2}{\pi}} \exp\left(-\frac{C_u^2}{2}\right)$

Quanto maior a "confiança" sobre a decisão a ser tomada menor a atenção dada ao exemplo. Por outro lado, os exemplos de difícil classificação, que produzem incerteza (baixo nível de confiança C), receberão maior atenção.

Fator de fracasso: $W_3(\epsilon_u, C_u) = \left(H\left(C(1 - Z\epsilon_u)\right) \right)^{-1}$

Ao contrário dos anteriores, este fator depende do erro ϵ_u cometido pelo perceptron no exemplo que está sendo aprendido. Muita atenção é dada ao exemplo se houve fracasso na resposta. Entretanto, fracassar em uma resposta em que não havia confiança é menos grave do que fracassar em uma resposta feita com grande confiança. Por isso o fator de fracasso depende da confiança C também, e assume um valor elevado para uma resposta feita com grande confiança que fracassa⁶.

Organismos mais sofisticados buscariam ativamente fazer experiências e enfrentar problemas para os quais seu grau de confiança é muito baixo ou nulo. Assim, os exemplos aprendidos não seriam espontâneos, mas **selecionados com C próximo de zero**. A velocidade de aprendizagem da regra seria bem maior neste caso.

Mudança de regra durante a aprendizagem

Talvez uma consideração mais importante seja a de que o algoritmo de estabilidade esperada, colocado nesta forma dependente de e_j , possui duas propriedades muito interessantes. Primeiro, fica solucionado o problema da separação entre a "fase de aprendizagem" e a "fase de desempenho" do perceptron. Na medida em que o desempenho melhora, automaticamente a aprendizagem (mudança nas conexões) vai diminuindo pois a função peso decresce rapidamente (na nossa analogia, o organismo-perceptron acaba por não mais "prestar atenção" aos novos exemplos, respondendo a estes apenas com o conhecimento já acumulado).

Segundo, o algoritmo se mostra robusto frente a mudanças na regra $|B\rangle$: neste caso, essa mudança será detectada pela medida de e_j e automaticamente o "nível de atenção" (peso W) se eleva, reiniciando a fase de aprendizagem. É um processo de autoregulação esquematizado na figura (3.21).

Infelizmente não tivemos oportunidade de estudar com mais detalhe este problema de adaptação do perceptron a regras mutáveis. Apresentamos, apenas para ilustração, uma simulação em que, no estágio em que $\alpha = 5$, a regra $|B\rangle$ é trocada por uma outra escolhida de forma totalmente aleatória. Comparamos o desempenho entre o nosso método e a regra de Hebb (figuras (3.22) e (3.23)).

Se chamarmos de tempo de recuperação t_R o número de exemplos necessários para que o perceptron volte ao nível de desempenho apresentado antes da mudança de regra, observamos que t_R é bem menor para o nosso algoritmo do que para a regra de Hebb simples. Na verdade, o desempenho de perceptron a partir de $\alpha = 5$ é praticamente o mesmo que o da aprendizagem a partir de tabula rasa. A regra $|B\rangle$ antiga é rapidamente "esquecida". Seria interessante estudar o caso em que a regra muda de forma contínua, e o desempenho de outros algoritmos nesta situação. Este tipo de problema foi considerado por [Hs 92].

⁶O fator W_2 está ausente do "algoritmo com peso simétrico". E mesmo assim, este algoritmo apresenta desempenho razoável. O mesmo acontece com o algoritmo com seleção de exemplos (que se reduz a W_1). No nosso conhecimento, estes algoritmos são os únicos (com exceção da regra de Hebb simples) que não usam uma comparação entre a saída atual σ_j^u e a desejada σ_j^u em sua função peso.

FIGURA 3.21 — Esquema representando os parâmetros usados nos diversos algoritmos de aprendizagem do Perceptron: o bloco superior representa o perceptron, e o bloco inferior o mecanismo de aprendizagem. A regra de Hebb usa apenas a entrada $|S^\mu\rangle$ e a saída desejada σ_B^μ como parâmetros de aprendizagem. O algoritmo perceptron usa uma retroalimentação do parâmetro σ_J^μ (traços). Os algoritmos Adaline, Relaxação e Estabilidade Esperada assumem que o campo local $\tilde{h}_\mu$ (sólido) e eventualmente o módulo de $|J\rangle$ (traço-ponto) podem ser usados na aprendizagem. O algoritmo auto-regulado assume que, além disso, um pequeno sistema de memória (pontilhado) pode fazer uma avaliação do desempenho atual do perceptron.

MUDANÇA DE REGRA (ESTABILIDADE ESPERADA)

FIGURA 3.22 — Efeito da mudança de regra durante a aprendizagem. Algoritmo de Estabilidade esperada auto-regulado com janela de memória de $\Delta\alpha=0.25$ ($N=149$, $n=1$): ($\square$) generalização $G(\alpha)$, (Δ) avaliação $1 - (e_g)$ usada pelo algoritmo, ($\circ$) aprendizagem $L(\alpha)$, (sólido) teórico Estabilidade Esperada. Após a mudança de regra de B para B' , medimos $L(\alpha)$ exigindo que as saídas exigidas para os vetores pertencentes ao conjunto de treinamento (mesmo aqueles aprendidos antes da mudança de regra) sejam as novas classificações $\sigma_B^\mu = \text{sgn}(\langle B' | S^\mu \rangle)$.

MUDANÇA DE REGRA (HEBB SIMPLES)

FIGURA 3.23 — Efeito da mudança de regra durante a aprendizagem. Regra de Hebb ($N=149$, $n=1$): ($\square$) generalização $G(\alpha)$ e ($\circ$) aprendizagem $L(\alpha)$. A curva para o algoritmo de Estabilidade Esperada (sólido) foi colocada para comparação.

CAPITULO 4

PROBLEMA DE GENERALIZAÇÃO

COM APRESENTAÇÃO REPETIDA DOS EXEMPLOS

Neste capítulo iremos estudar a situação de aprendizagem em que os exemplos podem ser apresentados várias vezes ao perceptron de modo a se minimizar uma "energia de aprendizagem" no conjunto de exemplos L .

Apresentaremos um cálculo do tipo feito por Gardner e Derrida para o espaço de acoplamentos $\{J\}$, para o problema de generalização com perceptron professor linear. O cálculo é simples mas (talvez por isso mesmo) ainda não foi publicado na literatura⁷.

Infelizmente, ainda não conseguimos adaptar o método de Gardner-Derrida para calcular o parâmetro de ordem $\rho(\alpha)$ de nosso algoritmo. Isso acontece por causa da forma da função energia do algoritmo de estabilidade esperada. Discutiremos isto na seção 4.2.

Apresentaremos simulações usando o algoritmo de estabilidade esperada apenas para o caso da dinâmica sequencial iterada. Mostraremos que o desempenho desse algoritmo aproxima muito bem o limite teórico dado pelo "algoritmo de Bayes".

Mostraremos também que o algoritmo pode ser usado de uma forma prática pois a convergência no erro de treinamento e_1 sinaliza a convergência em ρ , indicando quando se deve parar o processo de aprendizagem.

Finalmente, apresentaremos resultados **preliminares** para situação com seleção de exemplos.

⁷ O cálculo foi feito em nosso trabalho de mestrado, porém na fase final de elaboração da dissertação recebemos um preprint de Seung, Sompolinsky e Tishby [Se 92], que contém estes resultados e que deverá ser publicado nos próximos meses.

4.1 — APRENDIZAGEM COM PROFESSOR LINEAR

No problema de generalização para o professor linear mostramos que o melhor ganho em generalização em cada exemplo μ é obtido usando-se um peso

$$w_{\mu}^* = (K_{\mu} - \Delta_{\mu}) \quad (4.1)$$

Este algoritmo satisfaz a condição de caixa negra pois $K_{\mu} = |b_{\mu}|/\rho$ e a saída do perceptron linear é $\sigma_{\mu}^{\mu} = b_{\mu}$. Calculamos analiticamente $\rho(\alpha)$ para o primeiro passo do algoritmo iterado. Devemos considerar agora qual a habilidade de generalização obtida quando o número de ciclos de iteração é grande ($t \rightarrow \infty$).

Este algoritmo é do tipo Adaline e sua dinâmica foi estudada por Hertz [He 90] [Kr91] através da equação de Langevin correspondente. Hertz mostrou a existência de uma transição de segunda ordem para perfeita generalização quando $\alpha \rightarrow 1$, onde o tempo de relaxação do algoritmo diverge na forma $\tau \propto (1-\alpha)^{-1}$.

Iremos calcular, pelo método introduzido por Gardner e Derrida [Ga 88b], a expressão para $\rho(\alpha)$. Este tratamento se tornou padrão nos estudos do processo de aprendizagem e o cálculo que faremos visa apenas mostrar o método em uma situação bastante simples.

4.1.1 — TRATAMENTO ESTATÍSTICO DO PROCESSO DE APRENDIZAGEM

A "mecânica estatística" do processo de aprendizagem que se segue é uma extensão, para o caso de energia de treinamento finita, da análise de Gardner do espaço de acoplamentos de redes neurais [Ga 88a]. Foi proposta por inicialmente por Carnevali e Partanella [Ca 87], e desenvolvida por Gardner e Derrida [Ga 88b] e Tishby, Levin e Solla [Ti 89] [Le 89], tornando-se a abordagem padrão no estudo do processo de aprendizagem [Ha 89] [Gy 90a] [Gy 90b] [Gr 91]. Nas próximas seções seguiremos de perto a apresentação e notação de Györgyi e Tishby [Gy90a].

Considere uma versão estocástica da dinâmica de aprendizagem (em paralelo) vista nos capítulos anteriores. Os pesos sinápticos obedecem a equação de Langevin

$$\frac{d\hat{J}_j}{dt} = - \frac{\partial E}{\partial \hat{J}_j} - \frac{\partial V}{\partial \hat{J}_j} + \eta_j(t) \quad (4.2)$$

onde $E(\{\hat{J}_j\}, \alpha)$ é a energia de treinamento do algoritmo e está relacionada com o peso dado aos exemplos

$$- \frac{\partial E}{\partial \hat{J}_j} = \sum_{\mu=1}^{\alpha N} w_{\mu} \sigma_{\mu}^{\mu} s_j^{\mu} \quad (4.3)$$

(lembramos que estamos usando as variáveis normalizadas porque não é possível encontrar uma função energia deste tipo usando as variáveis J_j apenas).

$V(\{\hat{J}_j\})$ é um potencial relacionado com as restrições sobre as sinapses

$$- \frac{\partial V}{\partial \hat{J}_j} = \sum_{\mu=1}^{\alpha N} \Omega_{\mu} \hat{J}_j \quad (4.4)$$

A dinâmica de aprendizagem difere da estudada anteriormente apenas pelo termo $\eta_j(t)$ que assumimos ser ruído branco, ou seja, onde não há correlação espacial ou temporal

$$\langle \eta_i(t) \eta_j(t') \rangle = 2T \delta_{ij} \delta(t-t') \quad (4.5)$$

Como é bem conhecido, o estado estacionário desta dinâmica obedece uma distribuição de probabilidade de Gibbs

$$P_{\alpha}(\{\hat{J}_j\}) = \frac{e^{-\beta E}}{Z} \quad (4.6)$$

onde a variância do ruído está relacionada com a "temperatura de aprendizado" $T = 1/\beta$. O índice α lembra que a energia de aprendizado depende dos αN exemplos.

A função de partição, que normaliza $P_{\alpha}(\{\hat{J}_j\})$, é

$$Z(\alpha, \beta) = \int_{-\infty}^{\infty} \left(\prod_{j=1}^N d\hat{J}_j \right) P_0(\{\hat{J}_j\}) \exp(-\beta E(\alpha)) \quad (4.7)$$

onde incluímos a contribuição de V na distribuição "a priori" $P_0(\{\hat{J}_j\})$ das redes $|\hat{J}\rangle$ no espaço dos acoplamentos $\mathbb{R}^N$. De agora em diante vamos assumir que a única restrição sobre os valores das sinapses seja a condição de normalização ("perceptron esférico")

$$\langle J | J \rangle = \langle B | B \rangle = N \quad (4.8)$$

Com esta condição, podemos trabalhar com as variáveis não normalizadas J_j . A distribuição a priori no espaço de acoplamentos é

$$P_0(\{J_j\}) = (2\pi e)^{-N/2} \delta\left(\sum_{j=1}^N J_j^2 - N\right) \quad (4.9)$$

Neste caso a função de partição fica na forma

$$Z(\alpha, \beta) = \int_{-\infty}^{\infty} \left(\prod_{j=1}^N \frac{dJ_j}{\sqrt{2\pi e}} \right) \delta\left(\sum_{j=1}^N J_j^2 - N\right) \exp(-\beta E(\alpha)) \quad (4.10)$$

Lembramos que

$$E(\alpha) = \frac{1}{2} \sum_{\mu=1}^{\alpha N} E_{\mu} = \frac{1}{2} \sum_{\mu=1}^{\alpha N} (b_{\mu} - h_{\mu})^2 \quad (4.11)$$

onde

$$b_{\mu} = \frac{\langle B | S^{\mu} \rangle}{\sqrt{N}}, \quad h_{\mu} = \frac{\langle J | S^{\mu} \rangle}{\sqrt{N}} \quad (4.12)$$

Agora podemos usar o formalismo da mecânica estatística de equilíbrio para calcular médias com respeito a distribuição $P_{\alpha}(|J\rangle)$. Essas médias "térmicas" serão denotadas por $\langle \bullet \rangle_T$ e, no limite termodinâmico, representam quantidades típicas das redes geradas pela dinâmica de aprendizagem para $t \rightarrow \infty$.

Porém, ainda subsiste a dependência nos exemplos do conjunto de treinamento. Estes exemplos $|S^{\mu}\rangle$, uma vez escolhidos aleatoriamente, são fixos durante o processo de aprendizagem e representam uma "desordem congelada" ("quenched") no sistema. Assim, é necessário considerar esta desordem fazendo-se uma média sobre a distribuição dos exemplos $P(|S^{\mu}\rangle)$, que será denotada por $\langle \bullet \rangle_E$.

A energia livre por sinapse é

$$f(\alpha, \beta) = - \frac{1}{\beta N} \langle \ln Z(\alpha, \beta) \rangle_E \quad (4.13)$$

A energia de treinamento por exemplo é

$$e_t(\alpha) = \frac{1}{\alpha N} \langle \langle E(\alpha, \mathbf{L}) \rangle_T \rangle_E = \frac{1}{\alpha} \frac{\partial(\beta f)}{\partial \beta} \quad (4.14)$$

A quantidade de interesse é a densidade de estados ou volume fracionário típico do espaço de acoplamentos $\{J_j\}$ com uma dada energia de treinamento

$$V(\alpha, \beta) = \exp(N S(\alpha, \beta)) \quad (4.15)$$

onde S é a entropia por sinapse, obtida por uma transformação de Legendre da energia livre

$$S(\alpha, \beta) = \beta (\alpha e_t - f) = \frac{1}{N} \langle \langle \ln \frac{P_0(\{J_j\})}{P_\alpha(\{J_j\})} \rangle_T \rangle_E \quad (4.16)$$

Györgyi e Tishby mostraram que no caso de haver ruído nos exemplos a generalização ótima é obtida usando-se a estratégia de aprendizagem com energia de treinamento finita (notar que no caso deles essa energia é o próprio erro binário de treinamento). Porém, se não há ruído nos exemplos, a estratégia ótima é simplesmente aprender com energia de treinamento nula. Como este é o nosso caso poderíamos fazer, diretamente o cálculo da $V(\alpha)$ por

$$V(\alpha) = \exp \langle \ln(V(\alpha, \mathbf{L})) \rangle_E \quad (4.17)$$

onde

$$\begin{aligned} V(\alpha, \mathbf{L}) &= \frac{\int_{-\infty}^{\infty} \left(\prod_{j=1}^N dJ_j \right) \delta \left(\sum_{j=1}^N J_j^2 - N \right) \prod_{\mu=1}^{\alpha N} \delta(b_\mu - h_\mu)}{\int_{-\infty}^{\infty} \left(\prod_{j=1}^N dJ_j \right) \delta \left(\sum_{j=1}^N J_j^2 - N \right)} \\ &= \frac{Z(\alpha, \infty)}{Z(\alpha, 0)} \end{aligned} \quad (4.18)$$

em vez de obtê-lo por

$$V(\alpha) = \lim_{\beta \rightarrow \infty} V(\alpha, \beta) \quad (4.19)$$

O cálculo feito dessa forma fica mais simples e é o método idealizado inicialmente por Gardner [Ga 88a]. Porém iremos continuar a considerar o caso com β finito a fim de, além de trabalhar em um contexto mais geral, mostrar que usar $\beta \rightarrow \infty$ realmente nos dá uma generalização ótima.

4.1.2 MÉTODO DE RÉPLICAS

Queremos calcular a função de energia livre dada pela equação (4.13). Usando o método de réplicas o problema de se calcular a média de um logaritmo é convertido no problema mais tratável para uma potência

$$-\beta f N = \langle \ln Z(\alpha) \rangle_E = \lim_{n \rightarrow 0} \frac{\langle Z^n(\alpha) \rangle_E - 1}{n} \quad (4.20)$$

O método consiste em calcular (4.20) para n inteiro no limite $N \rightarrow \infty$ e fazer uma continuação analítica para $n = 0$. A função $\langle Z^n(\alpha) \rangle_E$ pode ser pensada como a função de partição de n cópias (réplicas) do nosso sistema que não interagem entre si.

Usando uma representação integral para a função delta

$$\delta(x) = \int_{-\infty}^{\infty} \frac{dy}{2\pi} \exp(iy x)$$

escrevemos

$$\langle Z^n(\alpha) \rangle_E = \int_{-\infty}^{\infty} \prod_{\nu=1}^n \left(\prod_{j=1}^N \frac{dJ_j^\nu}{\sqrt{2\pi e}} \right) \int_{-\infty}^{\infty} \frac{d\epsilon^\nu}{2\pi} \exp\left(i \sum_{j=1}^N \epsilon^\nu ((J_j^\nu)^2 - 1)\right) \times \otimes \quad (4.21)$$

com

$$\mathcal{Q} = \left\langle \prod_{\nu, \mu} \exp(-\beta E_{\mu}^{\nu}) \right\rangle_E = \left\langle \prod_{\nu, \mu} \exp\left(-\frac{\beta}{2N} \sum_{i,j=1}^N (S_i^{\mu} D_i^{\nu})(S_j^{\mu} D_j^{\nu})\right) \right\rangle_E \quad (4.22)$$

onde introduzimos o vetor diferença

$$|D^{\nu}\rangle = |B\rangle - |J^{\nu}\rangle \quad (4.23)$$

e usamos a notação simplificada $\prod_{\nu, \mu} \equiv \prod_{\nu=1}^n \prod_{\mu=1}^{\alpha N}$.

Linearizando por uma transformação gaussiana temos

$$\mathcal{Q} = \int_{-\infty}^{\infty} \left(\prod_{\nu, \mu} \frac{dz_{\nu\mu}}{\sqrt{2\pi}} \right) \left\langle \exp\left(\sum_{\mu=1}^{\alpha N} \sum_{\nu=1}^n \left(i \sqrt{\frac{\beta}{N}} z_{\nu\mu} \sum_{j=1}^N S_j^{\mu} D_j^{\nu} - \frac{z_{\nu\mu}^2}{2} \right) \right) \right\rangle_E \quad (4.24)$$

Feita a média sobre os exemplos (lembrando que S_j^{μ} tem valor ± 1 com igual probabilidade $P = \frac{1}{2}$) aparece na integral o termo

$$\begin{aligned} \cos\left(\sum_{\nu=1}^n \sqrt{\frac{\beta}{N}} z_{\nu\mu} D_j^{\nu} \right) &\simeq 1 - \frac{\beta}{2N} \sum_{\nu, \sigma=1}^n z_{\nu\mu} z_{\sigma\mu} D_j^{\nu} D_j^{\sigma} \\ &\simeq \exp\left(-\frac{\beta}{2N} \sum_{\nu, \sigma=1}^n z_{\nu\mu} z_{\sigma\mu} D_j^{\nu} D_j^{\sigma} \right) \end{aligned} \quad (4.25)$$

onde foi feita a aproximação para N grande. Ficamos com

$$\mathcal{Q} = \int_{-\infty}^{\infty} \left(\prod_{\nu, \mu=1}^{\alpha N} \frac{dz_{\nu\mu}}{\sqrt{2\pi}} \right) \exp\left(-\frac{1}{2} \sum_{\mu=1}^{\alpha N} \sum_{\nu, \sigma=1}^n z_{\nu\mu} \left(\delta^{\nu\sigma} + \frac{\beta}{N} \sum_j D_j^{\nu} D_j^{\sigma} \right) z_{\sigma\mu} \right) \quad (4.26)$$

Note que esse procedimento introduziu interação entre as réplicas. Para efetuar esta integração introduzimos os parâmetros

$$R^{\nu} = \frac{1}{N} \sum_j B_j J_j^{\nu} \quad (4.27)$$

$$q^{\nu\sigma} = \frac{1}{N} \sum_j J_j^{\nu} J_j^{\sigma} \quad (4.28)$$

e a matriz

$$M^{\nu\sigma} = \delta^{\nu\sigma} + \frac{1}{N} \sum_j D_j^{\nu} D_j^{\sigma} = \delta^{\nu\sigma} + \beta (1 - R^{\nu} - R^{\sigma} + q^{\nu\sigma}) \quad (4.29)$$

Estes parâmetros são introduzidos na forma de vínculos através de integrais sobre $q^{\nu\sigma}$ (para $\nu < \sigma$). Escrevemos

$$\begin{aligned} \Theta &= \int_{-\infty}^{\infty} \left(\prod_{\nu=1}^n N dR^{\nu} \delta \left(N R^{\nu} - \sum_j B_j J_j^{\nu} \right) \right) \left(\prod_{\nu < \sigma} N dq^{\nu\sigma} \delta \left(N q^{\nu\sigma} - \sum_j J_j^{\nu} J_j^{\sigma} \right) \right) \times \\ &\times \int_{-\infty}^{\infty} \left(\prod_{\nu, \mu} \frac{dz_{\nu\mu}}{\sqrt{2\pi}} \right) \exp \left(-\frac{1}{2} \sum_{\nu, \sigma=1}^n z_{\nu\mu} M_{\nu\sigma} z_{\sigma\mu} \right) \end{aligned} \quad (4.30)$$

A integral Gaussiana é

$$(\det \mathbf{M})^{-\alpha N/2} = \exp \left(-\frac{\alpha N}{2} \sum_{\nu} \ln \lambda_{\nu} \right) = \exp \left(-\frac{\alpha N}{2} \text{Tr} \ln \mathbf{M} \right) \quad (4.31)$$

onde λ_{ν} são os autovalores da matriz $\mathbf{M}$.

Usando representações integrais para as funções delta e voltando para a expressão de Z^n temos

$$\begin{aligned} \langle Z^n \rangle_E &= (2\pi e)^{-nN/2} \int_{-\infty}^{\infty} \left(\prod_{\nu=1}^n \frac{N d\epsilon^{\nu} dK^{\nu} dR^{\nu}}{4\pi^2} \right) \left(\prod_{\nu < \sigma} \frac{N d\phi^{\nu\sigma} dq^{\nu\sigma}}{2\pi} \right) \\ &\times \exp iN \left(\sum_{\nu=1}^n (K^{\nu} R^{\nu} - \epsilon^{\nu}) + \sum_{\nu < \sigma} \phi^{\nu\sigma} q^{\nu\sigma} \right) \\ &\times \int_{-\infty}^{\infty} \left(\prod_{j=1}^N \left(\prod_{\nu=1}^n dJ_j^{\nu} \right) \exp i \left(\sum_{\nu=1}^n (\epsilon^{\nu} (J_j^{\nu})^2 - K^{\nu} J_j^{\nu} B_j) - \sum_{\nu < \sigma} \phi^{\nu\sigma} J_j^{\nu} J_j^{\sigma} \right) \right) \\ &\times \exp \left(-\frac{\alpha N}{2} \text{Tr} \ln \mathbf{M} \right) \end{aligned} \quad (4.32)$$

Note que podemos escrever o penúltimo termo como

$$\begin{aligned} \prod_{j=1}^N \int_{-\infty}^{\infty} \left(\prod_{\nu=1}^n dJ_j^{\nu} \right) \exp i \left(\sum_{\nu=1}^n (\epsilon^{\nu} (J_j^{\nu})^2 - K^{\nu} J_j^{\nu} B_j) - \sum_{\nu < \sigma} \phi^{\nu\sigma} J_j^{\nu} J_j^{\sigma} \right) &= \\ = \exp -N \ln \left(\int_{-\infty}^{\infty} \left(\prod_{\nu=1}^n dJ^{\nu} \right) \exp i \left(\sum_{\nu=1}^n (\epsilon^{\nu} (J^{\nu})^2 - K^{\nu} J^{\nu}) - \sum_{\nu < \sigma} \phi^{\nu\sigma} J^{\nu} J^{\sigma} \right) \right) \end{aligned}$$

já que temos N integrais independentes e podemos eliminar B_j por uma mudança de variáveis.

Assim, ficamos com

$$\begin{aligned} \langle Z^n \rangle_E &= \int_{-\infty}^{\infty} \left(\prod_{\nu=1}^n \frac{d\epsilon^\nu dK^\nu dR^\nu}{(2\pi/N)^{3/2}} \right) \left(\prod_{\nu < \sigma}^n \frac{d\phi^{\nu\sigma} dq^{\nu\sigma}}{2\pi/N} \right) \\ &\times \exp \left(-N \left(G_1(n) + G_2(n) + \alpha G_3(n) \right) \right) \end{aligned} \quad (4.33)$$

onde

$$G_1(n) = -\frac{n}{2} \ln(e) + \sum_{\nu=1}^n \left(E^\nu - K^\nu R^\nu \right) - \sum_{\nu < \sigma}^n F^{\nu\sigma} q^{\nu\sigma} \quad (4.34)$$

$$G_2(n) = \ln \int_{-\infty}^{\infty} \left(\prod_{\nu=1}^n \frac{dJ^\nu}{\sqrt{2\pi}} \right) \exp \left(\sum_{\nu=1}^n K^\nu J^\nu - \frac{1}{2} \sum_{\nu, \sigma=1}^n \left(2 E^\nu \delta^{\nu\sigma} - F^{\nu\sigma} \right) J^\nu J^\sigma \right) \quad (4.35)$$

$$G_3(n) = -\frac{1}{2} \text{Tr} \ln M \quad (4.36)$$

(note que distribuimos o termo $(2\pi e)^{-n/2}$ entre $G_1(n)$ e $G_2(n)$).

As integrais (4.33) podem ser calculadas para N grande pelo método de ponto de sela

$$\int_{-\infty}^{\infty} \prod_{i=1}^m du^i e^{-N G(u)} \simeq \left(\frac{2\pi}{N} \right)^{m/2} (\text{Det } H)^{-1/2} e^{-N G(u_0)} \quad (4.37)$$

onde $u_0 \equiv \{u_0^i\}$ é o ponto estacionário no qual $\partial G / \partial u^i = 0$ e a matriz Hessiana H com componentes $\partial^2 G / \partial u^\nu \partial u^\sigma$ calculada em u_0 é assumida ser positiva definida.

No nosso caso devemos determinar os valores estacionários das variáveis ϵ^ν , R^ν , K^ν , $q^{\nu\sigma}$, $\phi^{\nu\sigma}$, ou das coordenadas transformadas

$$K^\nu = -i k^\nu, \quad E^\nu = -i \epsilon^\nu, \quad F^{\nu\sigma} = -i \phi^{\nu\sigma}, \quad F^{\sigma\sigma} = 0 \quad (4.38)$$

lembrando que existem apenas $n(n-1)/2$ variáveis independentes $q^{\nu\sigma}$.

Encontramos que, a menos de constantes, a energia livre por acoplamento é

$$f = -\frac{1}{\beta N} \lim_{n \rightarrow 0} \frac{\langle Z^{n(\alpha)} \rangle_E - 1}{n}$$

$$\begin{aligned}
&= -\frac{1}{\beta N} \lim_{n \rightarrow 0} \frac{\exp\left(-N \left(G_1(n) + G_2(n) + \alpha G_3(n) \right) \right) - 1}{n} \\
&= -\frac{1}{\beta} (G_1 + G_2 + \alpha G_3)
\end{aligned} \tag{4.40}$$

onde

$$G_i = \lim_{n \rightarrow 0} \frac{1}{n} G_i(n), \quad (i = 1, 2, 3) \tag{4.41}$$

4.1.3 - SOLUÇÃO DE RÉPLICAS SIMÉTRICAS

A solução de réplicas simétricas assume a independência em relação ao índice ν dos valores estacionários de E^ν , R^ν , K^ν , $q^{\nu\sigma}$, $F^{\nu\sigma}$. Calculando $G_i(n)$ e tomando o limite $n \rightarrow 0$ (Apêndice 4.A), temos

$$G_1 + G_2 = E - KR + \frac{1}{2} \left(Fq + \frac{F+K^2}{2E+F} - \ln(2E+F) - 1 \right) \tag{4.42}$$

$$G_3 = -\frac{1}{2} \left(\ln(1+\beta-\beta q) + \frac{\beta(1-2R+q)}{1+\beta(1-q)} \right) \tag{4.43}$$

Usando as condições de estacionariedade para E , R e K em (4.42) obtemos

$$2E+F = \frac{1}{1-q}, \quad \frac{F+K^2}{2E+F} = \frac{q}{1-q}, \quad K = \frac{R}{1-q} \tag{4.44}$$

Substituindo em (4.42), a energia livre (4.40) é

$$-\beta f = \frac{1}{2} \left(\frac{q-R^2}{1-q} + \ln(1-q) - \alpha \ln(1+\beta(1-q)) - \frac{\alpha\beta(1-2R+q)}{1+\beta(1-q)} \right) \tag{4.45}$$

As condições de estacionariedade $\frac{\partial f}{\partial R}$ e $\frac{\partial f}{\partial q}$ podem ser colocadas na seguinte forma

$$R = \alpha \frac{\beta(1-q)}{1+\beta(1-q)} \tag{4.46}$$

$$q = \frac{(2R-\alpha-1)R^2}{R^2-\alpha} \tag{4.47}$$

Estes parâmetros de ordem possuem uma interpretação simples. R é o overlap típico entre o perceptron aluno e o professor

$$R = \frac{1}{N} \left\langle \left\langle \langle B | J \rangle \right\rangle_T \right\rangle_E \quad (4.48)$$

e q é um parâmetro do tipo Edward-Anderson

$$q = \frac{1}{N} \left\langle \left\langle \langle J | \right\rangle_T \left\langle | J \rangle \right\rangle_T \right\rangle_E \quad (4.49)$$

Para α grande temos $q \sim R^2$ e

$$R = 1 - \frac{T}{2\alpha} + O(\alpha^{-2}) \quad (4.50)$$

O overlap entre o perceptron aluno e professor, e portanto a habilidade de generalização, é maior no limite $\beta \rightarrow \infty$ que corresponde ao processo de aprendizagem com energia de treinamento nula. Nesse limite, obtemos simplesmente

$$R = q = \alpha \quad (4.51)$$

A entropia $S(\alpha) = -\beta f$ diverge para $-\infty$ quando $\alpha \rightarrow 1$, sinalizando a transição de fase dada pelo colapso do volume

$$V(\alpha) = \lim_{\beta \rightarrow \infty} V(\alpha, \beta) = e^{NS(\alpha)} \quad (4.52)$$

de redes compatíveis com energia de treinamento nula. Como existe uma rede $|B\rangle$ com energia nula, concluímos que as redes encontradas pelo algoritmo não diferem de forma macroscópica de $|B\rangle$. A partir de $\alpha=1$ teremos perfeita generalização, $R=1$.

Deveríamos agora analisar a validade da solução de réplicas simétricas estudando a estabilidade da energia livre com respeito a pequenas flutuações. Isso é feito calculando o espectro da matriz de segundas derivadas da energia livre f avaliada nos valores réplica simétricos (4.46) e (4.47). Mas, uma vez que a solução encontrada é coerente com os resultados de Hertz [He 90], não iremos nos estender nesse ponto. (Ver [Se92]).

4.2 APRENDIZAGEM COM PROFESSOR BOOLEANO

O algoritmo de estabilidade esperada para o perceptron booleano com aprendizagem em paralelo é definido por

$$\frac{d\hat{J}_j}{dt} = - \left\langle \frac{\partial \hat{E}(\alpha, t)}{\partial \hat{J}_j} \right\rangle_{\alpha|\Delta} = \sum_{\mu=1}^{\alpha N} \bar{w}(\alpha, t) \sigma_B^\mu S_j^\mu \quad (4.53)$$

onde $\hat{E}(t)$ é a energia de aprendizagem do perceptron linear

$$\hat{E}(\alpha, t) = \frac{1}{2} \sum_{\mu=1}^{\alpha N} (K_\mu - \Delta_\mu(t-1))^2 \quad (4.54)$$

e $\langle \bullet \rangle_{\alpha|\Delta}$ denota o valor esperado de $\bullet$ dado Δ_μ . Vimos no capítulo 3 que

$$\bar{w}(\alpha, t) = \sqrt{\frac{1}{2\pi}} \lambda \sum_{\mu=1}^{\alpha N} \exp\left(-\frac{(\Delta_\mu/\lambda)^2}{2}\right) \left(H\left(-\Delta_\mu/\lambda\right)\right)^{-1} \quad (4.55)$$

onde

$$\lambda(\alpha, t) = \frac{\sqrt{1 - \rho^2(\alpha, t)}}{\rho(\alpha, t)} \quad (4.56)$$

Podemos encontrar uma energia de aprendizagem para o algoritmo (que chamaremos de $\bar{E}(t)$) notando que

$$\frac{d \ln(H(x))}{dx} = \frac{-1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) (H(x))^{-1} \quad (4.57)$$

Assim,

$$\bar{E}(\alpha, t) = - \lambda^2 \sum_{\mu=1}^{\alpha N} \ln\left(H\left(-\Delta_\mu/\lambda\right)\right) \quad (4.58)$$

Neste ponto deveríamos usar esta energia para encontrar, pelo método de Gardner e Derrida, uma expressão para $\rho(\alpha)$. Mas isto se revelou difícil dada a forma da função energia e este problema ainda está sendo estudado.

Podemos entretanto antecipar que não haverá transição de fase para generalização perfeita no perceptron booleano. Isso pode ser visto facilmente escrevendo

$$\begin{aligned}\frac{dJ_j}{dt} &= J \sum_{\mu=1}^{\alpha N} \left(\langle k_{\mu} \rangle_{0|\Delta} - \Delta_{\mu} \right) \sigma_{\theta}^{\mu} s_j^{\mu} \\ &= J \sum_{\mu=1}^{\alpha N} \left(k_{\mu} - \Delta_{\mu} \right) \sigma_{\theta}^{\mu} s_j^{\mu} + \eta_j(t)\end{aligned}\quad (4.60)$$

onde

$$\eta_j(t) = J \sum_{\mu=1}^{\alpha N} \left(\langle k_{\mu} \rangle_{0|\Delta} - k_{\mu} \right) \sigma_{\theta}^{\mu} s_j^{\mu} \quad (4.61)$$

Isso significa que a dinâmica do algoritmo de estabilidade esperada sem ruído é equivalente à dinâmica do perceptron linear com ruído! O mesmo vale para os outros algoritmos de aprendizagem. Ora, encontramos na seção anterior que o perceptron linear com ruído não apresenta transição para generalização perfeita.

Existe ainda um problema que devemos enfrentar em primeiro lugar: vimos que, na apresentação única de exemplos, havia a possibilidade de se trocar a dependência em ρ por uma dependência em α' . Para os demais ciclos de iteração porém, isso não foi possível. É essencial encontrar uma forma de evitar a dependência em ρ .

Neste trabalho não iremos calcular pelo método de réplicas as curvas de aprendizagem do algoritmo ou estudar seus aspectos dinâmicos. Iremos nos contentar em:

- a) Resolver o problema da dependência em ρ a fim de que colocar o algoritmo em uma forma utilizável para a situação com iteração.
- b) Estudar através de simulações o desempenho assintótico do algoritmo.

4.2.1 COMPARAÇÃO COM ALGORÍTMO DE BAYES

Queremos examinar o desempenho potencial de nosso algoritmo. **Iremos estudar apenas a dinâmica sequencial** (a dinâmica em paralelo leva ao mesmo desempenho final, porém necessita de maior número de iterações). Para isso realizamos uma simulação usando a função peso $\bar{\psi}_\mu$ na forma

$$\bar{\psi} = \frac{1}{\sqrt{2\pi}} \lambda \exp\left(-\frac{(\Delta_\mu/\lambda)^2}{2}\right) \left(H\left(-\Delta_\mu/\lambda\right)\right)^{-1} \quad (4.62)$$

onde

$$\lambda = \frac{\sqrt{1-\rho^2}}{\rho} = \operatorname{tg}(\pi e_\theta) \quad (4.63)$$

e ρ é função de μ e t . Note que eliminamos o prefator J de modo a diminuir o passo dado em cada iteração.

O valor de ρ , conforme medido durante a simulação, é colocado na função peso. Enfatizamos que isto será feito apenas para observar qual seria o desempenho possível do algoritmo, ou seja, o desempenho caso ρ fosse um parâmetro acessível durante a aprendizagem.

O resultado é mostrado na figura (4.1), junto com o limite teórico de desempenho para o perceptron booleano dado pelo algoritmo de Bayes. O algoritmo parece atingir, ou pelo menos aproximar bem, esse desempenho ótimo.

O algoritmo de classificação ótima de Bayes, por vezes chamado de algoritmo de maioria ponderada, foi estudado para o perceptron por Oppen e Haussler [Op 91]. O algoritmo de Bayes não pode ser implementado em um perceptron de camada única, porém fornece o limite teórico de $\rho(\alpha)$ para o problema de generalização com exemplos com distribuição uniforme, servindo como padrão de comparação para outros algoritmos de aprendizagem.

Basicamente, o algoritmo consiste em calcular, para um vetor não pertencente ao conjunto de treinamento, as probabilidades P_+ e P_- de se encontrar, após treinamento em αN exemplos, uma rede cuja saída seja $\sigma_j = +1$ ou $\sigma_j = -1$ respectivamente. Escolhe-se então a resposta com maior probabilidade posterior. Estas probabilidades correspondem aos volumes V_+ e V_- no espaço de acoplamentos das redes que produzem uma ou outra classificação.

DESEMPENHO POTENCIAL : ALGORÍTMO ESTABILIDADE ESPERADA

FIGURA 4.1 — Desempenho potencial em generalização do algoritmo de Estabilidade Esperada: o parâmetro $\rho(\alpha', t)$ é medido durante a simulação e usado na função peso $\bar{W}$. Note o efeito de tamanho finito devido ao modo de se medir $G(\alpha)$ ($N = 149$, $n = 20$, 40 iterações): (o) $G(\alpha)$ medido sobre conjunto de teste estatisticamente independente, (Δ) $G(\alpha)$ medido por $1 - \pi^{-1} \text{acos}(\rho)$ e (sólido) algoritmo de Bayes.

FIGURA 4.2 — Algoritmo de Bayes [Op 91]: a saída da rede é definida por uma regra de maioria sobre $n \rightarrow \infty$ redes treinadas independentemente.

A implementação do algoritmo de Bayes em uma rede neural é mostrada na figura (4.2): consiste em se fazer uma regra de maioria sobre um conjunto de n redes ($n \rightarrow \infty$) treinadas independentemente por certo algoritmo (chamado algoritmo de Boltzmann) que será descrito a seguir. Fazendo-se assim, são medidos os volumes V_+ e V_- pela frequência pós-treinamento das redes geradas pelo algoritmo. A classificação σ que possui o maior volume V de redes compatíveis com os αN exemplos será a escolhida e apresentada pelo neurônio de saída.

Definição do algoritmo de Boltzmann: O espaço de acoplamentos (ou seja, o espaço dos vetores $|J\rangle$) possui uma densidade de redes dv_0 a priori (ou seja, pré-treinamento) que reflete apenas restrições geométricas. Após o treinamento, o volume fracionário de redes compatíveis com os exemplos é $V(\alpha)$. O algoritmo usado no treinamento deve ser capaz de produzir aleatoriamente redes compatíveis com os exemplos de acordo com a densidade posterior

$$dv(\alpha) = \frac{dv_0}{V(\alpha)} \quad (4.64)$$

Um algoritmo com esta propriedade é chamado de algoritmo de Boltzmann.

Opper e Haussler calcularam, usando o método de réplicas, o overlap típico ρ_B do algoritmo de Bayes. Ele é obtido a partir da equação [Op 91]

$$\frac{\rho_B^2}{\sqrt{1-\rho_B^2}} = \frac{\alpha}{\pi} \int_{-\infty}^{\infty} Dt \frac{\exp(-t^2 \rho_B^2 / 2)}{H(t \rho_B)} \quad (4.65)$$

Assim, a curva de desempenho ótimo em generalização para o perceptron booleano sem seleção de exemplos será dada por

$$G_B(\alpha) = 1 - \frac{1}{\pi} \arccos(\rho_B) \quad (4.66)$$

O erro de generalização assintótico ($\rho_B \rightarrow 1$) é

$$e_g^B \simeq \frac{1}{\alpha} \left(\int_{-\infty}^{\infty} Dt \frac{\exp(-t^2 \rho_B^2 / 2)}{H(t \rho_B)} \right)^{-1} \simeq \frac{0.441}{\alpha} \quad (4.67)$$

É interessante notar que o erro assintótico e_g^B do algoritmo de Bayes é exatamente metade do erro assintótico e_g do algoritmo de estabilidade esperada para o primeiro passo da dinâmica sequencial (seção 3.2.3).

Vimos que o algoritmo de estabilidade esperada potencialmente atinge (ou pelo menos aproxima bem) a curva de desempenho ótimo (Bayes). Será que temos algo mais do que simples evidência numérica disso?

Até agora não. Entretanto, mostraremos a seguir que o algoritmo de estabilidade esperada é equivalente à maximização da probabilidade condicional

$$P(\sigma_B|h) = \prod_{\mu=1}^{\alpha N} P(\sigma_B^\mu|h_\mu) \quad (4.68)$$

onde usamos a notação vetorial $\sigma_B \equiv (\sigma_B^1, \sigma_B^2, \dots, \sigma_B^{\alpha N})$ e $h \equiv (h_1, h_2, \dots, h_{\alpha N})$ (note que não existe correlação entre σ_B^μ e h_ν para $\mu \neq \nu$).

Acreditamos que o fato do algoritmo de estabilidade esperada envolver a maximização da probabilidade condicional $P(\sigma_B|h)$ poderá fornecer a conexão com o algoritmo de Bayes discutido anteriormente (porém isto ainda não está claro e precisa ser melhor estudado). Em todo caso, esta demonstração constitui uma nova (e mais simples) derivação do algoritmo.

Considere duas variáveis gaussianas reduzidas b_μ e h_μ com probabilidade condicional

$$P(b_\mu|h_\mu) = \frac{1}{\sqrt{2\pi(1-\rho^2)}} \exp\left(-\frac{(b_\mu - \rho h_\mu)^2}{2(1-\rho^2)}\right) \quad (4.69)$$

A probabilidade condicional $P(b_\mu|h_\mu)$ nos diz que, dado h_μ , a variável b_μ/ρ está gaussianamente distribuída em torno de h_μ com uma variância igual a λ^2 . Vamos escrever essa probabilidade na seguinte forma

$$P(b_\mu|h_\mu) = Z^{-1} e^{-\beta' E(b_\mu, h_\mu)} \quad (4.70)$$

onde

$$E(b_\mu, h_\mu) = \frac{1}{2} \left(\frac{b_\mu}{\rho} - h_\mu \right)^2, \quad \beta' = \frac{\rho^2}{1-\rho^2} = \frac{1}{\lambda^2} \quad (4.71)$$

$$Z = \int_{-\infty}^{\infty} db e^{-\beta' E(b_\mu, h_\mu)} = \sqrt{2\pi(1-\rho^2)} \quad (4.72)$$

Podemos pensar neste problema como um sistema físico cujo microestado é simbolizado por b_μ e cuja energia é dada por E . Quando a temperatura $T = \lambda^2$ vai para zero o sistema se congela no estado h_μ , que é o estado de mínima energia.

O importante é notar que a energia encontrada tem a forma da energia de treinamento ótima $\hat{E}$ do perceptron linear, de modo que o algoritmo de aprendizagem

para o perceptron linear com aprendizagem em paralelo é definido simplesmente pela minimização da energia do sistema

$$\frac{d\hat{J}_i}{dt} = - \frac{\partial E(\mathbf{b}, \mathbf{h})}{\partial \hat{J}_i} = - \lambda^2 \frac{\partial \ln P(\mathbf{b}|\mathbf{h})}{\partial \hat{J}_i} \quad (4.73)$$

com

$$\ln P(\mathbf{b}|\mathbf{h}) = \sum_{\mu=1}^{\alpha N} \ln(P(b_\mu|h_\mu)) \quad (4.74)$$

onde levamos em conta que existem αN variáveis b e h .

Ou seja, $|J\rangle$ é modificado de tal forma que o logaritmo da probabilidade condicional para os estados b_μ seja maximizada³. Note que neste formalismo β' não está relacionado com a variança do ruído de treinamento usado na abordagem de Gardner-Derrida (usamos o apóstrofe para diferenciá-los). Aqui não há ruído de treinamento, e $\beta' > 0$ simplesmente indica que a correlação entre o perceptron aluno e o professor não é perfeita.

No caso do perceptron booleano, cada componente de nosso sistema físico possui apenas dois estados, com probabilidade

$$P(\sigma_B^\mu|h_\mu) = H(-\sigma_B^\mu h_\mu/\lambda) = \frac{e^{-\beta' E(\sigma_B^\mu, h_\mu)}}{Z} \quad (4.75)$$

$$E(\sigma_B^\mu, h_\mu) = - \lambda^2 \ln H(-\sigma_B^\mu h_\mu/\lambda) \quad (4.76)$$

$$Z = \sum_{\sigma_B^\mu = \pm 1} H(-\sigma_B^\mu h_\mu/\lambda) = 1 \quad (4.77)$$

de modo que o algoritmo de estabilidade esperada é definido por

$$\frac{d\hat{J}_i}{dt} = - \lambda^2 \frac{\partial \ln P(\sigma_B|\mathbf{h})}{\partial \hat{J}_i} \quad (4.78)$$

O algoritmo maximiza o logaritmo de $P(\sigma_B|\mathbf{h})$. Isto lembra o espírito da estimativa de Bayes, porém a equivalência (ou não) entre os dois procedimentos ainda fica por ser elucidada.

³ A grandeza $\ln P(\mathbf{b}|\mathbf{h})$ está relacionada com a entropia condicional S de nosso sistema através de

$$S = - \int_{-\infty}^{\infty} \prod_{\mu=1}^{\alpha N} db_\mu \quad P(\mathbf{b}|\mathbf{h}) \ln P(\mathbf{b}|\mathbf{h})$$

4.2.2 APRENDIZAGEM USANDO APROXIMAÇÃO DE BAYES

Nossa proposta de solução para o problema da dependência em ρ é usar no lugar de $\rho(\alpha', t)$ o valor assintótico ($t \rightarrow \infty$) $\rho_B(\alpha)$ do algoritmo de Bayes (note que $\alpha = P/N$ se refere ao tamanho do conjunto de treinamento enquanto que $\alpha' = \mu/N$ é a posição do exemplo μ dentro desse conjunto).

A simulação foi feita usando-se a função peso (4.68), onde o valor $\lambda(\alpha', t)$ é substituído simplesmente por

$$\lambda_B(\alpha) = \frac{\sqrt{1 - \rho_B^2(\alpha)}}{\rho_B(\alpha)} \quad (4.79)$$

Com este procedimento a função peso fica na forma $\bar{W}(\alpha, \Delta_\mu)$.

Na figura (4.3) comparamos o valor de $\rho(\alpha)$ obtido pelo algoritmo com o valor teórico máximo $\rho_B(\alpha)$. Apresentamos o desvio padrão das medidas da simulação junto com a diferença $\Delta\rho = \rho_B - \rho$ na figura (4.4). Esta diferença mostrou-se muito menor que o desvio padrão de modo que, para a quantidade de medidas feitas (um número razoável de 30 redes), os dois valores de ρ são numericamente indistinguíveis. Conjecturamos que esta versão do algoritmo de estabilidade esperada converge para o valor $\rho_B(\alpha)$, mas isto precisa ser demonstrado analiticamente.

Outra questão importante é: como determinar o momento de encerrar o algoritmo? A questão surge porque pode ser verificado que a função custo (energia) e sua derivada nunca se anulam. E no entanto existe uma convergência rápida no valor de ρ , como será mostrado a seguir. Queremos encerrar o processo de aprendizado quando atingirmos um valor de ρ (ou e_1) estacionário, porém não temos acesso a esses valores. Felizmente veremos que é possível usar a evolução erro de aprendizado e_1 para determinar o momento da convergência em e_1 .

Isso é mostrado nas figuras (4.5), (4.6) e (4.7) onde foram colocados, para vários valores de α , o erro de aprendizado e_1 , e as diferenças $\Delta e_1 \equiv e_1(t) - e_1(t-1)$ e $\Delta e_2 \equiv e_2(t) - e_2(t-1)$ em função do número de apresentações t . Observamos que os dois tipos de erro possuem aproximadamente o mesmo tempo de convergência.

LOC(1- ρ) X NÚMERO DE EXEMPLOS

FIGURA 4.3 — Desempenho em generalização do algoritmo de Estabilidade Esperada usando $\rho_B(\alpha)$ como parâmetro na função peso ($N = 149$, $n = 30$, 40 iterações): (o) $\log_{10}(1-\rho(\alpha))$ e (sólido) algoritmo de Bayes. Usamos escala logarítmica para maior visibilidade.

DP and $\Delta\rho$ ($l=40$) X NÚMERO DE EXEMPLOS

FIGURA 4.4 — Desvio padrão e diferença $\rho_B(\alpha) - \rho(\alpha)$ para duas simulações com algoritmo de Estabilidade Esperada usando $\rho_B(\alpha)$ como parâmetro ($N = 149$, $n = 30$, 40 iterações): (o) Desvio Padrão e (Δ) $\rho_B(\alpha) - \rho(\alpha)$.

FIGURA 4.5 — Convergência do algoritmo de Estabilidade Esperada para $\alpha = 1.0$ ($N = 149$, $n = 30$, 40 iterações): (o) erro de aprendizagem e_t , ($\square$) $\Delta e_t = e_t(t) - e_t(t-1)$, e ($\diamond$) $\Delta e_g = e_g(t) - e_g(t-1)$.

FIGURA 4.6 — O mesmo da figura anterior para $\alpha = 1.3$.

FIGURA 4.7 — O mesmo da figura anterior para $\alpha = 2.0$.

FIGURA 4.8 — Desempenho em aprendizagem $L(\alpha)$ do algoritmo de Estabilidade Esperada usando $\rho_B(\alpha)$ como parâmetro ($N = 149$, $n = 30$, 40 iterações): (o) $L(\alpha)$ (note a ordem de grandeza da escala vertical).

Um tópico importante é o valor final do erro de aprendizado. Algoritmos que, para exemplos sem ruído, produzem $e_1 = 0$ são chamados de "consistentes", e "inconsistentes" se $e_1 > 0$. Meir e Fontanari [Me 92] mostraram que vários algoritmos atingem seu melhor desempenho em generalização com erro de aprendizado e_1 maior que zero. Estes autores conjecturam que isso se deve a um ruído efetivo de amostragem produzido pelo fato de que α é finito.

De fato encontramos também que, a partir de certo α_c , nosso algoritmo apresenta erro de treinamento não nulo (figura (4.8)). Entretanto, o erro e_1 encontrado é **duas ordens de magnitude menor** que o encontrado no algoritmo de relaxação estudado por Meir e Fontanari. Por ser este erro tão pequeno, fica difícil medir através da simulação o valor exato de α onde ocorre a transição, porém estimamos que $\alpha_c \sim 1$.

O que acontece se, em vez de usarmos o valor $\rho_B(\alpha)$ na função peso, usarmos um parâmetro fixo r ($0 \leq r \leq 1$)? Procedendo assim estaremos definindo uma família de algoritmos, com função peso na forma $\bar{W}(r, \Delta_\mu)$ e podemos examinar o desempenho para vários valores de r .

Como é mostrado nas figuras (4.9) e (4.10) esta aproximação produz uma deterioração no desempenho de generalização, embora este seja ainda superior, pelo menos para α finito, ao dos algoritmos Perceptron e equivalente, ou talvez superior, ao algoritmo de relaxação (com $K \neq 0$, ver [Me92]).

As curvas de generalização tangenciam a curva ótima de Bayes para certo valor α^* . Seria interessante determinar se $\rho_B(\alpha^*)$ é igual a r para este ponto de tangência. Entretanto, nossas estatísticas não foram suficientes para mostrar isso, embora o valor α^* e o valor α_B (definido por $\rho_B(\alpha_B) = r$) estejam sempre próximos.

Outro aspecto curioso dessa família de algoritmos é que a curva de aprendizagem $L(\alpha)$ não possui necessariamente valor máximo para α pequeno, como é o caso da totalidade dos algoritmos conhecidos. O valor máximo de $L(\alpha)$ (que é igual a 1!) é alcançado para carga igual a α^* (figura (4.11)).

O comportamento da família de algoritmos com $\bar{W}(r, \Delta_\mu)$ parece indicar que a característica distintiva do algoritmo de estabilidade esperada (com $\bar{W}(\rho_B, \Delta_\mu)$) em relação aos outros é a dependência de um parâmetro de sua função peso no tamanho α do conjunto de treinamento. Este resultado é coerente com o encontrado por Meir e Fontanari [Me 92], que estudaram um algoritmo de relaxação com parâmetro K também dependente de α e que também apresenta resultados

numericamente indistinguíveis da curva de Bayes. Acreditamos que os dois algoritmos estejam relacionados e que talvez seja possível encontrar uma relação entre o parâmetro ótimo $K(\alpha)$ do algoritmo de Meir e Fontanari e o valor $\rho_B(\alpha)$ usado por nosso algoritmo. Não tivemos tempo de esclarecer esta relação porque estes resultados são muito recentes. Este tópico merece ser estudado futuramente.

Finalmente, apresentamos o desempenho do algoritmo no caso de mapeamento aleatório (figura (4.12)). Embora melhor que a regra de Hebb, o desempenho é inferior ao que pode ser obtido com outros algoritmos. Parece que, assim como os algoritmos que geram o perceptron de ótima estabilidade não são bons para o problema de generalização, o inverso também acontece: **nosso algoritmo não é ótimo para o problema de memorização.**

Mas isto parece se dever ao fato de que nosso algoritmo foi deduzido supondo-se que a distribuição de probabilidade das estabilidade no professor era uma distribuição Gaussiana truncada em zero. Ou seja, o algoritmo irá produzir exemplo com estabilidade muito pequena, o que é correto no problema de generalização mas incorreto no problema de memorização (mapeamento aleatório) pois aumenta a probabilidade de má classificação. Fica por examinar o caso em que se estima K_μ no algoritmo de estabilidade esperada assumindo outra distribuição, por exemplo,

$$P(C_\mu) = \sqrt{\frac{2}{\pi}} \exp\left(-\frac{(C_\mu - K)^2}{2}\right) \Theta(C_\mu - K) \quad (4.80)$$

onde C_μ é a estabilidade do exemplo no perceptron professor e K é uma constante (ver discussão em [Ab89a]).

ALGORÍTMOS $W(r)$: $G(\alpha)$

ALGORÍTMOS $W(r)$: $G(\alpha)$

FIGURAS 4.9 e 4.10 — Desempenho em generalização $G(\alpha)$ da família de algoritmos $W(r)$ ($N = 149$, $n = 30$, 40 iterações): ($\diamond$) $r = 0.8$ ($\square$) $r = 0.95$ (Δ) $r = 0.98$. Note que os valores de r com melhor desempenho para α grande apresentam desempenho ruim para α pequeno.

FIGURA 4.11 — Desempenho em aprendizagem $L(\alpha)$ da família de algoritmos $W(r)$ ($N = 149$, $n = 30$, 40 iterações): (★) $r = 0.8$ (□) $r = 0.95$ (◊) $r = 0.98$. Note que o valor máximo de $L(\alpha)$ não é obtido necessariamente para α pequeno.

FIGURA 4.12 — Desempenho em aprendizagem $L(\alpha)$ do algoritmo de Estabilidade Esperada para o caso de mapeamento aleatório ($N = 149$, $n = 30$, 40 iterações): (sólido) regra de Hebb e (o) Estabilidade Esperada

4.3 SELEÇÃO DE EXEMPLOS

Vamos agora examinar o desempenho potencial do algoritmo de estabilidade esperada na situação de aprendizagem com seleção de exemplos. Dissemos "potencial" porque iremos examinar o desempenho do algoritmo usando o valor de ρ medido durante a simulação.

Para esta situação com seleção de exemplos ainda não foi calculada a curva de generalização ótima, ou seja, ainda não foi calculado o desempenho do algoritmo de Bayes para este caso. Sendo assim, não poderemos usar a estratégia aplicada na seção anterior nem verificar se o nosso algoritmo aproxima o comportamento ótimo.

O primeiro passo da dinâmica foi descrito na seção 3.2.2. Neste passo os exemplos são selecionados satisfazendo a condição de ortogonalidade $\langle J(1, \mu) | S^\mu \rangle = 0$. Como vimos, no primeiro passo a função peso toma a forma

$$\bar{W}(\rho) = \frac{1}{\sqrt{2\pi}} \sqrt{1-\rho^2} \quad (4.81)$$

Note que, em geral, a condição de ortogonalidade não será mais satisfeita nos próximos passos, ou seja $\langle J(t, \mu) | S^\mu \rangle \neq 0$ para $t > 1$. Assim, para os próximos ciclos de apresentação a função peso completa $\bar{W}(\rho, h_\mu)$ deve ser usada.

Entretanto, existe um algoritmo melhor ainda! O procedimento anterior (que chamaremos de algoritmo A) seleciona todos os exemplos no primeiro passo da dinâmica, e depois é que se dá a iteração. Podemos pensar em um procedimento alternativo onde o processo de seleção ocorra após o aprendizado iterativo. Considere o seguinte algoritmo (B):

- Os primeiros $\Delta \alpha N$ exemplos são aprendidos pelo método A exposto acima.
- Após a convergência do algoritmo para estes $\Delta \alpha N$ exemplos obtemos um vetor $|J(\infty, \Delta \alpha)\rangle$ que certamente é melhor que $|J(1, \Delta \alpha)\rangle$. Este vetor será usado para selecionar os próximos $\Delta \alpha N$ exemplos ($\mu = \Delta \alpha N, \dots, 2\Delta \alpha N$).
- O processo de ajuste agora usa todos os $2 \Delta \alpha N$ exemplos, ou seja, todos esses exemplos são reapresentados até convergência. O novo vetor $|J(\infty, 2\Delta \alpha)\rangle$ é então usado para selecionar os próximos $\Delta \alpha N$ exemplos e assim por diante.

No limite $\Delta\alpha N \rightarrow 1$ teremos que um novo exemplo é selecionado apenas quando houve aprendizagem iterativa convergente com todos os exemplos anteriores a ele. Acreditamos que este limite deve produzir o melhor desempenho para esta classe de algoritmos. Entretanto não tivemos oportunidade de estudar esta situação adequadamente. Realizamos apenas uma simulação com $\Delta\alpha$ pequeno porém macroscópico. As simulações (figura (4.13)) mostram que realmente o desempenho do algoritmo B é melhor que o de A (sendo também igual ou superior ao desempenho dos algoritmos da literatura [Wt 92a]). Já o erro de aprendizagem de B é ligeiramente maior porque os exemplos aprendidos são de mais difícil classificação.

O trabalho com este algoritmo está incompleto, portanto. Para que o algoritmo possa ser usado numa forma prática precisamos calcular analiticamente $\rho(t \rightarrow \infty, \alpha)$ e usar a estratégia da seção anterior, ou seja, fazer a substituição $\rho(t, \alpha)$ por $\rho(\infty, \alpha)$. É claro que sempre podemos fazer alguma simplificação, como por exemplo usar (para $t > 1$) um valor fixo r em vez de $\rho(t, \alpha)$ como na família de algoritmos estudada na seção anterior.

É aqui que o conceito de aprendizagem auto-regulada pode ser útil. Entretanto, neste caso com aprendizagem iterativa, a avaliação do erro de generalização deve ser feita de forma diferente. Não podemos usar os erros recentes para avaliar e_g como na situação com apresentação única dos exemplos, pois neste caso estaremos na verdade avaliando o erro de aprendizagem e_l já que os exemplos são apresentados repetidamente ao perceptron. Assim, um possível algoritmo com aprendizagem auto-regulada teria a seguinte forma:

- Escolhemos um pequeno conjunto de teste $\mathcal{T}$ com $\Delta\alpha N$ exemplos que serão usados para avaliar e_g .
- Utilizamos o algoritmo B, ou seja, só selecionamos um novo exemplo ortogonal a $|J\rangle$ após o algoritmo estacionar para os exemplos anteriores.
- Após a seleção do exemplo, avaliamos e_g pelo erro médio no conjunto $\mathcal{T}$ e usamos o novo valor de e_g no processo de aprendizagem do novo exemplo (e dos anteriores se for o caso).

É claro que, para avaliar o desempenho do algoritmo, devemos contabilizar os exemplos de $\mathcal{T}$ como pertencentes ao conjunto de treinamento, embora eles não sofram processo de seleção. Assim, devemos fazer $\Delta\alpha$ tão pequeno quanto possível sem que isso comprometa a representatividade do conjunto $\mathcal{T}$ na avaliação de e_g .

SELEÇÃO DE EXEMPLOS: ALGORÍTMO ITERADO

FIGURA 4.13 — Desempenho em generalização e aprendizagem do algoritmo de Estabilidade Esperada na situação com seleção de exemplos ($N = 149$, $n = 30, 40$ iterações): ($\bullet$) $L(\alpha)$ método A, ($\blacktriangle$) $G(\alpha)$ método A, ($\circ$) $L(\alpha)$ método B, ($\square$) $G(\alpha)$ método B, (sólido) primeiro passo da dinâmica do método A.

CAPÍTULO 5

CONCLUSÕES E PERSPECTIVAS

É realmente notável que, tendo sido proposto há 34 anos, um modelo de rede neural tão simples como o perceptron só recentemente tenha tido algumas de suas propriedades básicas entendidas. A curva de ótima estabilidade para o problema de memorização de associações aleatórias foi encontrada por Gardner em 1987 e um algoritmo que atinge essa ótima estabilidade foi proposto pouco depois. A curva de ótima generalização para o problema mais simples possível (sinapses contínuas, função linearmente separável e distribuição uniforme dos exemplos) só foi obtida em 1991 por Oppen e Haussler. Com este trabalho estamos propondo um novo algoritmo de aprendizagem que parece atingir ou pelo menos aproximar bem esta ótima generalização: o algoritmo de estabilidade esperada. Nossos resultados principais são resumidos a seguir:

1. Encontramos os limites teóricos para a probabilidade de generalização média no primeiro passo da dinâmica de aprendizagem com ou sem seleção de exemplos. Isso foi feito tanto para o caso de aprendizagem com professor linear como para o professor booleano.

2. Uma nova família de algoritmos de aprendizagem foi proposta cujo desempenho no problema de generalização com professor booleano é muito bom. Mostramos que o algoritmo produz a curva de generalização ótima para o primeiro passo da dinâmica de aprendizagem. Para aprendizagem iterativa ainda não encontramos uma demonstração de que a curva de Bayes realmente represente o comportamento assintótico do algoritmo. Em todo caso, a curva ótima de Bayes pode ser aproximada por nosso algoritmo com uma precisão da ordem de 10^{-3} .

3. A aprendizagem com seleção de exemplos, embora de forma alguma artificial já que é análoga à prática científica de "experimentação", ainda não teve sua curva de generalização ótima calculada para o caso de aprendizagem iterativa. No caso da apresentação única dos exemplos o algoritmo de estabilidade esperada atinge a generalização ótima. Além disso, fornecemos uma demonstração de que, na aprendizagem com seleção de exemplos, estes devem ser escolhidos com estabilidade nula a fim de maximizar o ganho em generalização.

4. A ênfase que demos ao primeiro passo da dinâmica de aprendizagem ("problema de apresentação única dos exemplos") também é nova na literatura, embora o método para encontrar uma equação diferencial para $\rho(\alpha)$ tenha sido usado antes [Ab 89a] [Ki 90]. Mostramos que este primeiro passo é facilmente calculado para vários algoritmos e que fornece informação interessante sobre os mesmos: os algoritmos cujo desempenho em função de α estaciona na primeira apresentação dos exemplos (Adaline, Perceptron e relaxação com $K > 0$) apresentam, com aprendizagem iterada, erro de generalização com comportamento assintótico $\alpha^{-1/2}$; já os algoritmos cujo erro de generalização decai com α^{-1} (estabilidade esperada, peso simétrico, Perceptron e relaxação com $K = 0$), no primeiro passo da dinâmica mantêm esse desempenho com a iteração.

5. Mostramos também que essa grande diversidade de desempenhos possíveis depende muitas vezes apenas de um "parâmetro de passo" $\eta(\alpha')$, ao qual não tem sido dada muita atenção na literatura (ver porém [Hs92]).

6. O uso do erro de generalização como parâmetro de aprendizagem também é novo no campo de redes neurais. A versão "auto-regulada" do algoritmo é simples, dispensa o uso explícito do valor teórico de $\rho(\alpha')$ e pode ser adaptada para aprendizagem iterativa. O algoritmo se presta muito bem para o problema de adaptação do perceptron a regras que mudam com o tempo. Um mecanismo de retroalimentação que sinaliza quando a regra muda (reiniciando assim o processo de aprendizagem) aparece naturalmente no algoritmo sem que este tenha sido idealizado especificamente para esta tarefa. Esta versão sugere também uma abordagem para o problema da separação entre "fase de aprendizagem" e "fase de utilização" de uma rede neural.

7. A distinção entre as duas possíveis definições de habilidade de generalização $G(\alpha)$ e a proposta de uma nova medida de desempenho $P_E(\alpha, m)$ feitas no capítulo 2 fornecem uma abordagem para o problema de generalização em pequenas redes. Mostramos que, medindo-se $G(\alpha)$ pela definição corrente na literatura (probabilidade de reconhecimento de exemplos independentes de L), o desempenho de redes pequenas será necessariamente superior ao de redes maiores. Isso foi confirmado pelas simulações.

8. Finalmente, acreditamos que a organização de uma "árvore de algoritmos" obtidos de forma dedutiva (através de certas médias e limites) a partir do algoritmo de estabilidade esperada produz uma certa unificação no estudo dos algoritmos de aprendizagem e uma compreensão maior da natureza e possibilidades

de cada um deles. Distingue-se uma hierarquia entre os algoritmos e famílias de algoritmos aparentados. Demos alguns exemplos de novos algoritmos que podem ser obtidos a partir desta "árvore".

Quanto às perspectivas de trabalho, acreditamos ser possível calcular o desempenho para tempos longos de nosso algoritmo adaptando-se devidamente o método de Gardner-Derrida. Trabalhamos algum tempo neste problema sem conseguir avançar muito. Este é um ponto essencial que continuamos estudando ativamente. Também os aspectos dinâmicos (convergência, tempo de relaxação etc.) não receberam a devida atenção em nosso trabalho e precisam ser melhor estudados.

Outros problemas que poderiam ser investigados são:

1. Desempenho do algoritmo de estabilidade esperada no problema de generalização a partir de exemplos com ruído e de funções que não podem ser aprendidas exatamente pelo perceptron.

2. Estudo do problema de adaptação a regras que mudam com o tempo.

3. Estudo do problema de memorização de listas (apresentação única dos exemplos) e do efeito da ordem de apresentação dos exemplos na sua probabilidade de recordação.

4. Definição e estudo do problema de "extração de regra" em redes de atratores. Aplicação do algoritmo de estabilidade esperada para este problema.

Finalmente, gostaríamos de chamar a atenção para a possível relevância dos resultados e problemas discutidos na área de aprendizagem em redes neurais para os problemas estudados pela Epistemologia e Filosofia da Ciência. Afinal, o que se tem feito é uma "epistemologia" para redes neurais artificiais: estuda-se o problema da aprendizagem através de exemplos ("indução"), o tipo de regularidades ou regras ("conhecimento") acessíveis a essas redes, o problema do ruído ("incerteza") nos exemplos ("dados experimentais"), a eficiência de algoritmos de aprendizagem ("metodologias e estratégias de pesquisa"), o problema de mínimos locais no processo de aprendizagem e sua superação ("revoluções científicas?").

Seria muito interessante se uma análise comparada das várias propostas metodológicas em Filosofia da Ciência (Popper, Kuhn, Lakatos, Feyrabend etc.) viesse a sugerir novos algoritmos de aprendizagem para máquinas (ou vice-versa!). Por exemplo, poderíamos fazer uma analogia entre o "anarquismo epistemológico" de Feyrabend e a "aprendizagem a alta temperatura". E, sem dúvida, a função peso do algoritmo, que dá muito mais valor aos fracassos de previsão, lembra Popper.

Afinal, uma questão permanente em epistemologia tem sido: "em Ciência, que métodos e estratégias produzirão ótima generalização?"

"Precisamos tirar o melhor partido desse pouco que podemos atingir diretamente. É preciso que cada experiência nos possibilite o maior número possível de previsões e com o mais alto grau de probabilidade que se possa alcançar. O problema é, por assim dizer, o aumento do rendimento da máquina científica."

Henri Poincaré

Referências

- [Ab 89a] ABBOTT L. F. and KEPLER T. B., "Universality in the Space of Interactions for Network Models", *J. Phys. A: Math. Gen.* **22**, 2031 (1989).
- [Ab 89b] ABBOTT L. F. and KEPLER T. B., "Optimal Learning in Neural Network Memories", *J. Phys. A: Math. Gen.* **22**, 711 (1989).
- [Ab 90] ABBOTT L. F., "Learning in Neural Network Memories", *Network* **1**, 105 (1990).
- [Al 90] ALONI-LAVI R., GUTFREUND H. and YEKUTIELI I., "Effect of the Learning Procedure on Local Stabilities in Neural Networks", *Europhys. Lett.* **13**, 665 (1990).
- [Am 85a] AMIT D. J., GUTFREUND H. and SOMPOLINSKY H., "Spin-Glass Models of Neural Networks", *Phys. Rev.* **A32**, 1007 (1985).
- [Am 85b] AMIT D. J., GUTFREUND H. and SOMPOLINSKY H., "Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks", *Phys. Rev. Lett.* **55**, 1530 (1985).
- [Am 86] AMIT D. J., "The Properties of Models of Simple Neural Networks", in *Proceedings of the Heidelberg Symposium on Glassy Dynamics 1986*.
- [Am 87] AMIT D. J., GUTFREUND H. and SOMPOLINSKY H., "Statistical Mechanics of Neural Networks near Saturation", *Annals of Physics* **173**, 30 (1987).
- [Am 89] AMIT D. J., *Modeling Brain Functions*, Cambridge Univ. Press (New York, 1989).
- [An 89] ANLAUF J. K. and BIEHL M., "The AdaTron: an Adaptive Perceptron Algorithm", *Europhys. Lett.* **10**, 687 (1989).
- [As 89] ANSHELEVICH V. V., AMIRIKIAN B. R., LUKASHIN A. V. and FRANK-KAMENETSKII M. D., "On the Ability of Neural Networks to Perform Generalization by Induction", *Biol. Cybern.* **61**, 125 (1989).
- [Ar 87] ARBIB M. A., *Brains, Machines and Mathematics*, Springer-Verlag (New York, 1987).
- [Bl 62a] BLOCK H. D., "The Perceptron: A Model for Brain Functioning", *Rev. of Mod. Phys.* **34**, 123 (1962).
- [Bl 62b] BLOCK H. D., KNIGHT B. W., JR. and ROSENBLATT F., "Analysis of Four-Layer Series-Coupled Perceptron", *Rev. of Mod. Phys.* **34**, 135 (1962).
- [Bu 90] BURGESS N., MOORE M. A. and SHAPIRO J. L., "Human-like Forgetting in Neural Network Models of Memory", in *Neural Networks and Spin Glass, Proceedings Porto Alegre 1989 STATPHYS 17*, W. Theumann and R. Koberle (Eds.), World Scientific (Singapore, 1990).
- [Ca 87] CARNEVALI P. and PATARNELLO S., "Exhaustive Thermodynamical Analysis of Boolean Learning Networks", *Europhys. Lett.* **4**, 1199 (1987).
- [Co 73] COOPER, L. N., "A Possible Organization of Animal Memory and Learning", in *Proceedings of the Nobel Symposium on Collective Properties of Physical System*, B. Lundquist and S. Lundquist (Eds.), Academic Press (New York, 1973).

- [Cv 85] COVER T. M., "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition", *IEEE Trans. on Electronic Computers* **14**, 326 (1965).
- [De 87] DENKER J., SCHWARTZ D., WITTNER B., SOLLA S., HOPFIELD J. J., HOWARD R. and JACKEL L., "Large Automatic Learning, Rule Extraction and Generalization", *Complex Systems* **1**, 877.
- [dG 89] DEL GIUDICE P., FRANZ S. and VIRASORO M. A., "Perceptron Beyond the Limit of Capacity", *J. Phys. France* **50**, 121 (1989).
- [Di 87] DIEDERICH S. and OFFER M., "Learning of Correlated Patterns in Spin-Glass Networks by Local Learning Rules", *Phys. Rev. Lett.* **58**, 949 (1987).
- [Fa 90] FARMER J. D., "A Rosetta Stone for Connectionism", *Physica D* **42**, 153 (1990).
- [Fe 68] FELLER W., *An Introduction to Probability Theory and Its Applications*, John Wiley & Sons (New York, 1968).
- [Fo 90] FONTANARI J. F., "Generalization in a Hopfield Network", *J. Phys. France* **51**, 2421 (1990).
- [Fr 90] FRANZ S., AMIT D. J. and VIRASORO M. A., "Prosopagnosia in High Capacity Neural Networks Storing Uncorrelated Classes", *J. Phys. France* **51**, 387 (1990).
- [Ga 87] GARDNER E., "Maximum Storage Capacity in Neural Networks", *Europhys. Lett.* **4**, 481 (1987).
- [Ga 88a] GARDNER E., "The Space of Interactions in Neural Network Models", *J. Phys. A: Math. Gen.* **21**, 257 (1988).
- [Ga 88b] GARDNER E. and DERRIDA B., "Optimal Storage Properties of Neural Network Models", *J. Phys. A: Math. Gen.* **21**, 271 (1988).
- [Ga 89] GARDNER E. and DERRIDA B., "Three Unfinished Works on the Optimal Storage Capacity of Networks", *J. Phys. A: Math. Gen.* **22**, 1983 (1989).
- [Gr 91] GRINIASTY M. and GUTFREUND H., "Learning and Retrieval in Attractor Neural Networks above Saturation", *J. Phys. A: Math. Gen.* **24**, 715 (1991).
- [Gu 88] GUYON I., PERSONNAZ L., NADAL J.P., and DREYFUS G., "Storage and Retrieval of Complex Sequences in Neural Networks", *Phys. Rev.* **A38**, 6365 (1988).
- [Gu 91] GUYON I., "Neural Networks and Applications Tutorial", *Phys. Report.* **207** n° 3-5, 215 (1991).
- [Gy 90a] GYÖRGYI G., "Inference of a Rule by a Neural Network with Thermal Noise", *Phys. Rev. Lett.* **64**, 2957 (1990).
- [Gy 90b] GYÖRGYI G., "First-Order Transition to Perfect Generalization in a Neural Network with Binary Synapses", *Phys. Rev.* **A41**, 7097 (1990).
- [Ha 90] HANSEL D. and SOMPOLINSKY H., "Learning from Examples in a Single-Layer Neural Network", *Europhys. Lett.* **11**, 687 (1990).
- [He 90] HERTZ J. A., "Statistical Dynamics of Learning", *preprint NORDITA* (1990).

- [Ho 82] HOPFIELD J. J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Natl. Acad. Sci. USA* **79**, 2554 (1982).
- [KC 91] KINOUCI D. and CATICHA N., "Improving Generalization by Biased Learning in Perceptrons", *aceito para publicação no Physica A*.
- [KC 92] KINOUCI D. and CATICHA N., "Optimal Generalization in Perceptrons", *aceito para publicação no J. Phys. A: Math. Gen.*
- [Ki 89] KINZEL W., "Statistical Mechanics of Neural Networks", *Physica Scripta* **T25**, 144 (1989).
- [Ki 90] KINZEL W. and RUJÁN P., "Improving a Network Generalization Ability by Selecting Examples", *Europhys. Lett.* **13**, 473 (1990).
- [Kra 87] KRAUTH W. and MÉZARD M., "Learning Algorithms with Optimal Stability in Neural Networks", *J. Phys. A: Math. Gen.* **20**, 745 (1987).
- [Kr 92a] KROGH A., "Learning with noise in a Linear Perceptron", *J. Phys. A: Math. Gen.* **25**, 1119 (1992).
- [Kr 92b] KROGH A. and HERTZ J. A., "Generalization in a Linear Perceptron in the presence of Noise", *J. Phys. A: Math. Gen.* **25**, 1135 (1992).
- [Ku 91] KURCHAN J. and DOMANY E., "Critical Capacity of Noisy and Asymmetrically Constrained Perceptrons", *J. Phys. A: Math. Gen.* **24**, 1347 (1991).
- [Li 74] LITTLE W. A., *Math. Biosci.* **19**, 101 (1974).
- [McC 43] McCULLOCH W. S. and PITTS W., "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bull. Math. Biophys* **9**, 127 (1943).
- [McN 72] McNAB R. M. and KOSHLAND D. E., "The Gradient Sensing Mechanism in Bacterial Chemotaxis", *Proc. Nat. Acad. Sci. USA* **69**, 2509 (1972).
- [Me 92] MEIR R. and FONTANARI J. F., "On the Calculation of Learning Curves for Inconsistent Algorithms", *Phys. Rev. A* **45**
- [Mi 69] MINSKY M. and PAPERT S., *Perceptrons*, MIT Press (Cambridge, MA, 1969).
- [Mr 91] MIRANDA E. N., "Generalization in the Hopfield Model: Numerical Results", *J. Phys. I France* **1**, 999 (1991).
- [Mu 90] MULLER B. and REINHARDT J., *Neural Networks: an Introduction*, Springer-Verlag (Berlin, 1990).
- [Op 88] OPPER M., "Learning Times of Neural Networks: Exact Solution for a PERCEPTRON Algorithm", *Phys. Rev. A* **38**, 3824 (1988).
- [Op 89] OPPER M., "Learning in Neural Networks: Solvable Dynamics", *Europhys. Lett.* **8**, 389 (1989).
- [Op 90] OPPER M., KINZEL W., KLEINZ J. and NEHL R., "On the Ability of the Optimal Perceptron to Generalise", *J. Phys. A: Math. Gen.* **23**, L581 (1990).
- [Op 91] OPPER M. and HAUSSLER D., "Generalization Performance of Bayes Optimal Classification Algorithm for Learning a Perceptron", *Phys. Rev. Lett.* **66**, 2677 (1991).

- [Pe 85] PERSONNAZ L., GUYON I. and DREYFUS G., "Information Storage and Retrieval in Spin-Glass like Neural Networks", *Phys. Lett.* **46**, 359 (1985).
- [Pu 79] PUTNAN H., "Probabilidade Confirmação", in *Filosofia da Ciência*, org. Morgenbasser S., CULTRIX (São Paulo, 1979).
- [PDP 86] RUMELHART, D. and McCLELLAND J. (Eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, The MIT Press/Bradford Books (1986).
- [Re 89] REFREGIER PH. and VIGNOLLE J.-M., "An Improved Version of the Pseudo-Inverse Solution for Classification and Neural Networks", *Europhys.Lett.* **10**, 387 (1989).
- [Ro 58] ROSENBLATT F., "The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review* **65**, 386 (1958).
- [Ro 62] ROSENBLATT F., *Principles of Neurodynamics*, Spartan Books (New York 1962).
- [Se 91] SEUNG, H. S., SOMPOLINSKY H. and TISHBY N., submetido ao *J. Phys. A* (1991).
- [Sj 85] ACKLEY D. H., HINTON G. E. and SEJNOWSKY T. J., "A Learning Algorithm for Boltzmann Machines", *Cognitive Science* **9**, 147 (1985).
- [So 90] SOMPOLINSKY H. and TISHBY N., "Learning from Examples in Large Neural Networks", *Phys. Rev. Lett.* **65**, 1683 (1990).
- [Ti 89] TISHBY N., LEVIN E. and Solla S. A., "Consistent Inference of Probabilities in Layered Networks: Prediction and Generalization", in the *Proc. of Inter. Joint Conf. on Neural Net. Washington D.C. vol.2* (1989).
- [Ti 90] TISHBY N. and GYÖRGYI G., "Statistical Theory of Learning a Rule", in *Neural Networks and Spin Glass, Proceedings Porto Alegre 1989 STATPHYS 17*, W. Theumann and R. Koberle (Eds.), World Scientific (Singapura, 1990).
- [Va 84] VALIENT G. L., "A Theory of the Learnable", *Comm. ACM* **27**, 1134 (1984).
- [V1 89a] VALLET F., "The Hebb Rule for Learning Linearly Separable Boolean Functions: Learning and Generalization", *Europhys. Lett.* **8**, 747 (1989).
- [V1 89b] VALLET F., CAILTON J.-G. and REFREGIER Ph., "Linear and Nonlinear Extension of the Pseudo-Inverse Solution for Learning Boolean Functions", *Europhys. Lett.* **9**, 315 (1989).
- [V1 90] VALLET F. and CAILTON J.-G., "Recognition Rates of the Hebb Rule for Learning Boolean Functions", *Phys. Rev.* **441**, 3059 (1990).
- [van He 86] VAN HEMMEN J. L. e KÜHN R., "Nonlinear Neural Networks", *Phys. Rev. Lett.* **57**, 913 (1986).
- [Wa 69] WATANABE S., *Knowing and guessing - A Formal and Quantitative Study*, John Wiley & Sons (New York, 1969).
- [Wt 92] WATKIN T. L. H. and RAU A., "Selecting Examples for Perceptrons", *J. Phys. A: Math. Gen.* **25**, 113 (1992).
- [Wi 60] WIDROW B. and HOFF M. E., "Adaptive Switching Circuits", *WESCON Convention Record IV*, IRE (New York, 1960).

APÊNDICES

APÊNDICE 2.A: Valor médio e variância de Z_μ

Queremos obter o valor médio da variável

$$Z_\mu = \frac{1}{N} \operatorname{sgn}(b) \operatorname{sgn}(b_\mu) \langle S | S^\mu \rangle \quad (2A.1)$$

Escrevemos os vetores $|S\rangle$ e $|S^\mu\rangle$ numa base ortonormal $\{| \hat{B}_j \rangle\}$ onde $|B\rangle \equiv |B_1\rangle$ é um dos vetores de base

$$|S^\mu\rangle = b_{1\mu} |B_1\rangle + \sum_{j=2}^N b_{j\mu} | \hat{B}_j \rangle \quad (2A.2)$$

Ficamos com (lembrando que $b_\mu = b_{1\mu}$)

$$Z_\mu = \frac{1}{N} |b| |b_\mu| + \frac{1}{N} \operatorname{sgn}(b) \operatorname{sgn}(b_\mu) \sum_{j=2}^N b_{j\mu} b_j \quad (2A.3)$$

Fazendo-se a média sobre a variável $|b_\mu|$ o segundo termo irá desaparecer pois não há correlação entre $b_{1\mu}$ e os outros $b_{j\mu}$, de modo que

$$Z_\mu = \frac{1}{N} |b| \int_{-\infty}^{\infty} db_\mu P(|b_\mu|) |b_\mu| = \frac{1}{N} \sqrt{\frac{2}{\pi}} |b| \quad (2A.4)$$

que é o resultado usado no capítulo 2. Para obter a variância

$$\sigma^2 = \langle (Z_\mu - \bar{Z}_\mu)^2 \rangle_\mu$$

das variáveis Z_μ seguimos um procedimento similar (que não será repetido aqui), observando que o termo dominante será

$$\sigma^2 \simeq \langle Z_\mu^2 \rangle_\mu = \frac{1}{N} \quad (2A.5)$$

APÊNDICE 3.A: Equação de evolução para $J(\alpha)$

Para se obter a evolução do módulo do vetor de acoplamento

$$J(\mu-1) = \langle J(\mu-1) | J(\mu-1) \rangle \quad (3A.1)$$

no caso do problema linear de generalização multiplicamos a equação de evolução por $\langle J(\mu) |$ obtendo (no limite $N \rightarrow \infty$ e fazendo-se a média sobre as possíveis escolhas de $|S^\mu\rangle$,

$$\frac{dJ}{d\alpha'} = J \int_{-\infty}^{\infty} Dh_\mu \int_{-\infty}^{\infty} \tilde{D}b_\mu \left(W_\mu \sigma_B^\mu h_\mu + \frac{W_\mu^2}{2} \right) \quad (3A.2)$$

onde $W_\mu = \sigma_B^\mu F_\mu / J$.

Perceptron Linear: substituímos W_μ por W_μ^* obtemos

$$\frac{dJ}{d\alpha'} = \frac{J}{2} \left\langle \left\langle \frac{b_\mu^2}{\rho^2} - h_\mu^2 \right\rangle_{b/h} \right\rangle_h = \frac{J}{2 \rho^2} (1 - \rho^2) \quad (3A.3)$$

Mas

$$\frac{d\rho}{d\alpha'} = \frac{1 - \rho^2}{2 \rho} \quad (3A.4)$$

e usando a regra da cadeia finalmente obtemos

$$J = c \cdot \rho \quad (3A.5)$$

onde c é uma constante de integração que pode ser feita igual a 1 autoconsistentemente uma vez que uma constante na definição de F_μ não afeta a equação de evolução de ρ .

Algoritmo de Peso Simétrico: Usando $W_{\mu} = \sqrt{\frac{2}{\pi}} \lambda \exp\left(-\frac{h_{\mu}^2}{2\lambda^2}\right)$ obtemos

$$\frac{dJ}{d\alpha'} = J \left(\sqrt{\frac{2}{\pi}} \lambda A + \frac{\lambda^2}{\pi} B \right) \quad (3A.6)$$

onde

$$A = \int_{-\infty}^{\infty} Db \operatorname{sgn}(b) \int_{-\infty}^{\infty} \frac{dh}{\sqrt{2\pi(1-\rho^2)}} \exp\left(-\frac{(h/\rho - b)^2 - h^2}{2\lambda^2}\right) \quad (3A.7)$$

$$B = \int_{-\infty}^{\infty} Dh \int_{-\infty}^{\infty} \tilde{D}b \exp\left(-\frac{h_{\mu}^2}{2\lambda^2}\right) \quad (3A.8)$$

Com um pouco de álgebra obtém-se

$$A = \sqrt{\frac{2}{\pi}} \rho \frac{(1-\rho^2)}{\sqrt{1+\rho^2}}, \quad B = \sqrt{\frac{1-\rho^2}{1+\rho^2}} \quad (3A.9)$$

e assim temos finalmente

$$\frac{dJ}{d\alpha'} = \left(2J\rho + \frac{J}{\rho} \right) \frac{d\rho}{d\alpha'} \quad (3A.10)$$

onde usamos o resultado obtido no capítulo 3

$$\frac{d\rho}{d\alpha'} = \frac{1}{\pi} \frac{(1-\rho^2)^{3/2}}{\sqrt{1-\rho^2}} \quad (3A.11)$$

A solução da equação diferencial (3A.10) é

$$J = \rho e^{\rho^2 + c} \quad (3A.12)$$

Escolhemos $c = -1$ usando a liberdade já discutida anteriormente, de modo que $J = 1$ para $\rho = 1$.

Nos casos do algoritmo de estabilidade esperada com ou sem seleção de exemplos obtemos um resultado igual ao do perceptron linear, $J = \rho$. Os cálculos são parecidos com os feitos acima, de modo que não os repetiremos aqui.

APÊNDICE 3.B: Evolução de $\rho(\alpha)$ para algoritmos da classe Hebb

Vamos calcular a curva $\rho(\alpha)$ para os algoritmos da classe da regra de Hebb, ou seja, cujas funções peso não dependem de Δ_μ ou σ_μ^2 ,

$$W_H = \frac{1}{J} W_H(\alpha') \quad (3B.1)$$

Lembremos que a regra de Hebb clássica usa $W_H(\alpha') = 1/J(\alpha')$, ou seja, $W_H = 1$. O desempenho dessa regra foi analisado no capítulo 2. Outras escolhas para a função $W_H(\alpha')$ produzirão algoritmos com desempenhos diferentes.

A equação de evolução de $\rho(\alpha)$ (seção 3.2.1) para este caso é

$$\begin{aligned} \frac{d\rho}{d\alpha'} &= \rho \int_{-\infty}^{\infty} Dh_\mu \int_{-\infty}^{\infty} \tilde{D}b_\mu W_H \left(k_\mu - \Delta_\mu - \frac{W_H}{2} \right) \\ &= \sqrt{\frac{2}{\pi}} (1 - \rho^2) W_H - \frac{\rho W_H^2}{2} \end{aligned} \quad (3B.2)$$

Uma vez dada a forma da função $W_H(\alpha')$, podemos calcular o desempenho do algoritmo integrando numericamente a eq. (3B.2).

Vamos estudar o caso $W_H = C$ (constante). Neste caso $d\rho/d\alpha'$ se anula para algum valor $\rho_\infty < 1$, dado por (igualando a zero a eq. (3B.2)),

$$\rho_\infty = \sqrt{\frac{\pi C^2}{32} + 1} - \frac{C}{4} \sqrt{\frac{\pi}{2}} \quad (3B.3)$$

Assim, a função peso $W_H = C$ não permite a extração completa da regra. Este comportamento também será encontrado em outros algoritmos (por exemplo, Adaline).

Que escolha de $W_H(\alpha')$ produzirá o melhor desempenho? Usando a eq. (3B.2) obtemos que $d\rho/d\alpha'$ será maximizado se usarmos a função peso

$$W_H^*(\alpha') = \sqrt{\frac{2}{\pi}} \frac{1 - \rho^2}{\rho} \quad (3B.4)$$

Neste caso temos

$$\frac{d\rho}{d\alpha'} = \frac{(1 - \rho^2)^2}{\pi \rho} \quad (3B.5)$$

$$\int \frac{2 \rho d\rho}{(1-\rho^2)^2} = \frac{2}{\pi} \int_0^{\alpha} d\alpha' \quad (3B.6)$$

Integrando (e usando a condição $\rho(0) = 0$) obtemos finalmente

$$\rho = \left(1 + \frac{\pi}{2\alpha} \right)^{-1/2} \quad (3B.7)$$

Este resultado corresponde à curva $\rho(\alpha)$ da regra de Hebb encontrada no capítulo 2. Vamos mostrar agora que o uso da função peso ótima (3B.4) implica em

$$J(\alpha') = c/W_H^*(\alpha') \quad (c = \text{const.}) \quad (3B.8)$$

Usando a equação de evolução para J (ver apêndice 3.A) obtemos

$$\begin{aligned} \frac{dJ}{d\alpha'} &= J \left(\frac{2}{\pi} (1-\rho^2) + \frac{1}{\pi} \frac{(1-\rho^2)^2}{\rho^2} \right) \\ &= J \left(\frac{2\rho}{1-\rho^2} + \frac{1}{\rho} \right) \frac{d\rho}{d\alpha'} \end{aligned} \quad (3B.9)$$

onde usamos (0.5). Supondo $J = c/W_H^*$ temos

$$\begin{aligned} \frac{dJ}{d\alpha'} &= \frac{d}{d\alpha'} \left(\sqrt{\frac{\pi}{2}} \frac{c\rho}{1-\rho^2} \right) = c \sqrt{\frac{\pi}{2}} \left(\frac{1}{1-\rho^2} + \frac{2\rho}{(1-\rho^2)^2} \right) \frac{d\rho}{d\alpha'} \\ &= J \left(\frac{2\rho}{1-\rho^2} + \frac{1}{\rho} \right) \frac{d\rho}{d\alpha'} \end{aligned} \quad (3B.10)$$

Assim, concluímos que a regra de Hebb clássica ($\mathcal{W}_H = J W_H^* = c$) corresponde à função peso ótima para esta classe de algoritmos em que W_H independe de Δ_H .

APÊNDICE 3.C: Evolução de $\rho(\alpha)$ para algoritmos da classe Adaline

A função peso dos algoritmos da classe Adaline é

$$\mathcal{W}_\mu = J W_\mu = J (K - \Delta_\mu) \quad (3C.1)$$

onde K independe de Δ_μ (porém pode depender eventualmente de $\alpha' = \mu N$). Lembremos que o algoritmo Adaline clássico usa a função peso $\mathcal{W}_\mu = 1 - \langle J | S^\mu \rangle$, o que corresponde à escolha $K(\alpha') = 1/J(\alpha')$.

Introduzindo na equação de evolução de ρ obtemos

$$\begin{aligned} \frac{d\rho}{d\alpha'} &= \int_{-\infty}^{\infty} D h_\mu \int_{-\infty}^{\infty} \tilde{D} b_\mu \left(K |b_\mu| - \frac{\rho k^2}{2} - h_\mu b_\mu + \frac{\rho h_\mu^2}{2} \right) \\ &= \sqrt{\frac{2}{\pi}} K - \frac{\rho(1+k^2)}{2} \end{aligned} \quad (3C.2)$$

Integrando e usando a condição inicial $\rho(0) = 0$ temos finalmente

$$\rho = \rho_\infty \left(1 - e^{-\alpha/\tau} \right) \quad (3C.3)$$

onde

$$\rho_\infty(k) = \sqrt{\frac{2}{\pi}} \frac{2k}{1+k^2}, \quad \tau(k) = \frac{2}{1+k^2} \quad (3C.4)$$

Obtemos assim o resultado interessante de que o algoritmo Adaline, na situação de apresentação única dos exemplos, não consegue aprender a regra, ou seja, ρ é sempre menor que 1 mesmo para $\alpha \rightarrow \infty$. Encontramos também que o melhor desempenho assintótico ρ_∞ é obtido pelo algoritmo com $k = 1$, que atinge um overlap máximo

$$\rho_\infty(1) = \sqrt{\frac{2}{\pi}} \quad (3C.5)$$

com um tempo de convergência $\tau(1) = 1$.

Isto para um parâmetro k fixo. Podemos porém permitir que k dependa de α' e assim determinar a função $K(\alpha')$ que maximiza a velocidade de aprendizado da regra. A condição de máximo na eq. (3C.2) fornece

$$k(\alpha') = \frac{1}{\rho(\alpha')} \sqrt{\frac{2}{\pi}} \quad (3C.6)$$

A função peso ótima para a classe Adaline é então

$$W_A^* = \left(\frac{1}{\rho(\alpha')} \sqrt{\frac{2}{\pi}} - \Delta u \right) \quad (3C.7)$$

A equação diferencial para ρ fica

$$\frac{d\rho}{d\alpha'} = \frac{2/\pi - \rho^2}{2\rho} \quad (3C.8)$$

e integrando obtemos

$$\rho(\alpha) = \sqrt{\frac{2}{\pi}} \sqrt{1 - e^{-\alpha}} \quad (3C.9)$$

Assim, usar um valor de k dependente de $\alpha' = \mu N$ não melhora o comportamento assintótico ρ_∞ , mas apenas produz um menor tempo de convergência em relação à escolha $k = 1$. O desempenho do algoritmo para estes dois casos é mostrado nas figuras (3.15a) e (3.15b) na seção 3.4.

Vamos agora calcular a evolução de J para este caso ótimo. Usando a equação de evolução de J (ver apêndice 3.A) obtemos

$$\begin{aligned} \frac{dJ}{d\alpha'} &= J \left(\frac{k^2 - 1}{2} \right) = \frac{J}{\rho} \left(\frac{1}{\pi\rho} - \frac{\rho}{2} \right) \\ &= \frac{J}{\rho} \frac{d\rho}{d\alpha'} \end{aligned} \quad (3C.10)$$

cuja solução é

$$J(\alpha') = C \rho(\alpha') \quad (3C.11)$$

A constante C é arbitrária e afeta apenas a escala de J , não apresentando nenhum efeito sobre $\rho(\alpha)$.

Assim, a função peso ótima fica na forma

$$\mathcal{W}_A = J W_A^* = C \sqrt{\frac{2}{\pi}} - \langle J | S^\mu \rangle \quad (3C.12)$$

que corresponde ao algoritmo Adaline clássico de Widrow e Hoff se $C = \sqrt{\pi/2}$. O algoritmo Adaline clássico é ótimo dentro do conjunto de algoritmos definidos por (3C.1).

APÊNDICE 3.D: Evolução de $\rho(\alpha)$ para algoritmos da classe Perceptron

Vamos agora calcular $\rho(\alpha)$ para os algoritmos da classe do algoritmo Perceptron. Mostraremos os detalhes de cálculo apenas por interesse pedagógico. Para o algoritmo de Relaxação (apêndice 3.E) apresentaremos somente os resultados já que o cálculo é parecido com o que vem a seguir.

A função peso que define os algoritmos da classe Perceptron é

$$W_p = J W_p = J \eta(\alpha') \Theta(k - \Delta_\mu) \quad (3D.1)$$

onde $k = \text{constante}$ e $\Theta(x)$ é a função degrau (Heaviside). A equação de evolução para ρ assume a forma

$$\frac{d\rho}{d\alpha'} = \int_{-\infty}^{\infty} Db_\mu \int_{-\infty}^{\infty} \tilde{D}h_\mu \Theta(k - \Delta_\mu) \left(\eta |b_\mu| - \rho \eta \sigma_B^\mu h_\mu - \frac{\rho \eta^2}{2} \right) \quad (3D.2)$$

onde usamos a propriedade $(\Theta(x))^2 = \Theta(x)$ e

$$Db \equiv (2\pi)^{-1/2} \exp(-b^2/2) db$$

$$\tilde{D}h \equiv \left(2\pi(1-\rho^2) \right)^{-1/2} \exp\left(-\frac{(h-\rho b)^2}{2(1-\rho^2)} \right) dh$$

Podemos eliminar a função de Heaviside escrevendo

$$\frac{d\rho}{d\alpha'} = 2 \int_{-\infty}^0 Db_\mu \int_{-\kappa}^{\infty} \tilde{D}h_\mu \left(\eta |b_\mu| - \rho \eta \sigma_B^\mu h_\mu - \frac{\rho \eta^2}{2} \right) \quad (3D.3)$$

Fazendo a integração em h encontramos

$$\frac{d\rho}{d\alpha'} = \int_{-\infty}^0 Db_\mu \left(\left(2 \rho^2 \lambda^2 \eta |b_\mu| - \rho \eta^2 \right) H(v) - \sqrt{\frac{2}{\pi}} \rho^2 \lambda \eta \sigma_B^\mu e^{-v^2/2} \right) \quad (3D.4)$$

onde

$$v = \frac{-k - \rho b_\mu}{\sqrt{1 - \rho^2}}, \quad \lambda = \frac{\sqrt{1 - \rho^2}}{\rho}$$

$$H(x) \equiv (2\pi)^{-1/2} \int_x^{\infty} dt \exp(-t^2/2)$$

O fator σ_B^μ muda o sinal do último termo em (3D.2). Trocando os extremos da integral e fazendo a mudança $b \rightarrow -b$ escrevemos

$$\frac{d\rho}{d\alpha'} = \int_0^\infty Db_\mu \left((2\rho^2\lambda^2\eta|b_\mu| - \rho\eta^2) H(u) + \sqrt{\frac{2}{\pi}} \rho^2\lambda\eta e^{-u^2/2} \right) \quad (3D.5)$$

onde agora

$$u = \frac{-k + \rho b_\mu}{\sqrt{1 - \rho^2}}$$

Fazendo a mudança de variáveis acima (note que $du = \lambda^{-1}db_\mu$), escrevemos finalmente

$$\frac{d\rho}{d\alpha'} = \int_{-K/\rho\lambda}^\infty \tilde{D}u \left((2\rho^2\lambda^3\eta|\lambda u + k/\rho| - \rho\lambda\eta^2) H(u) + \sqrt{\frac{2}{\pi}} \rho^2\lambda^2\eta e^{-u^2/2} \right) \quad (3D.6)$$

onde

$$\tilde{D}u = (2\pi)^{-1/2} \exp\left(-\frac{(\lambda u - k/\rho)^2}{2}\right) du$$

A curva $\rho(\alpha)$ pode ser obtida integrando-se numericamente (3D.6) para uma dada função $\eta(\alpha')$. Se η for constante, teremos um termo de ordem λ negativo enquanto que os termos positivos serão de ordem $\geq \lambda^2$. Sendo assim, existirá um valor $\rho_\infty < 1$ tal que $d\rho/d\alpha'$ se anula, de modo que a regra não poderá ser aprendida. Isto é mostrado na figura (3.18) na seção (3.4).

Assumindo que o limite $\lambda \rightarrow 0$ exista para uma função $\eta(\alpha')$ genérica, vamos encontrar o comportamento assintótico do erro de generalização. Conservando apenas os termos de ordem superior em λ temos

$$\frac{d\rho}{d\alpha'} = \sqrt{\frac{2}{\pi}} \rho^2 \lambda^2 H\left(-\frac{K}{\rho\lambda}\right) \eta - \frac{\rho\lambda}{2\pi} I\left(-\frac{K}{\rho\lambda}\right) \eta^2 \quad (3D.7)$$

onde

$$I(x) = \sqrt{2\pi} \int_{-K/\rho\lambda}^\infty du H(u)$$

No caso em que $K \neq 0$ esta integral diverge, sinalizando que o limite $\lambda \rightarrow 0$ foi assumido erradamente (ou seja, o algoritmo estaciona para algum $\rho < 1$).

No caso em que $K = 0$ temos $I(0) = 1$ e $H(0) = 1/2$, ou seja

$$\frac{d\rho}{d\alpha'} = \sqrt{\frac{1}{2\pi}} \rho^2 \lambda^2 \eta - \frac{\rho \lambda}{2\pi} \eta^2 \quad (3D.8)$$

Se $\eta = \text{constante}$ (por exemplo, $\eta=1$) a dominância do termo negativo sinaliza que mesmo com $K = 0$ a evolução dada por (3D.6) também estaciona para algum $\rho < 1$. Porém se permitirmos que η dependa de α' , podemos obter a função η que otimiza o erro de generalização assintótico

$$\eta(\alpha') = \sqrt{\frac{\pi}{2}} \rho \lambda = \sqrt{\frac{\pi}{2}} \left(1 - \rho^2(\alpha')\right)^{-1/2} \quad (3D.9)$$

Substituindo na equação anterior ficamos com

$$\frac{d\rho}{d\alpha'} = -\frac{\rho^3 \lambda^3}{4} \quad (3D.10)$$

Vamos agora encontrar o erro de generalização assintótico. Assumindo que, para α grande,

$$\rho \simeq 1 - C \alpha^{-2} + \dots \quad (3D.11)$$

temos

$$\rho \lambda \simeq \sqrt{2C} \alpha^{-1} \quad (3D.12)$$

Substituindo em (3D.10) encontramos $\sqrt{2C} = 4$. O erro de generalização será então (ver capítulo 2)

$$e_g(\text{ótimo}) \simeq \frac{\rho \lambda}{\pi} = \frac{4}{\pi \alpha} \simeq \frac{1.27}{\alpha} \quad (3D.13)$$

Chamaremos de algoritmo "perceptron ótimo" ao algoritmo definido pelo peso (3D.1) com o fator $\eta(\alpha')$ encontrado acima.

Substituindo em (3D.6) vem

$$\frac{d\rho}{d\alpha'} = \rho^3 \lambda^3 \int_0^\infty \tilde{D}u \left(\left(\sqrt{2\pi} \rho \lambda^2 u - \frac{\pi}{2} \right) H(u) + e^{-u^2/2} \right) \quad (3D.14)$$

Integração numérica desta equação fornece a curva teórica mostrada na figuras (3.18) e (3.19) na seção (3.4), junto com resultados de simulação.

APÊNDICE 3.E: Evolução de $\rho(\alpha)$ para algoritmos da classe Relaxação

O cálculo para os algoritmos tipo relaxação definidos por

$$W_R = J W_R = J \eta(\alpha') (k - \Delta u) \Theta(k - \Delta u) \quad (3E.1)$$

é similar ao desenvolvido no apêndice anterior. A equação de evolução é

$$\begin{aligned} \frac{d\rho}{d\alpha'} = \int_{-K/\lambda}^{\infty} \tilde{D}u \left(\sqrt{\frac{2}{\pi}} \rho \lambda^2 \eta(k + \lambda u) e^{-u^2/2} - \right. \\ \left. - \rho \lambda \eta \left((2\lambda u + \eta)k + \lambda^2 \eta u^2 \right) H(u) \right) \end{aligned} \quad (3E.2)$$

Como na classe Perceptron, o algoritmo estaciona em $\rho < 1$ se $K \neq 0$. No caso $K=0$ temos

$$\frac{d\rho}{d\alpha'} = \eta \rho \lambda^3 \int_0^{\infty} \tilde{D}u \left(\sqrt{\frac{2}{\pi}} u e^{-u^2/2} - \eta u^2 H(u) \right) \quad (3E.3)$$

que para $\lambda \rightarrow 0$ se torna

$$\frac{d\rho}{d\alpha'} = \left(\eta - \frac{\eta^2}{2} I' \right) \frac{\rho \lambda^3}{\pi} \quad (3E.4)$$

onde

$$I' = \sqrt{2\pi} \int_0^{\infty} du H(u) = \frac{2}{3} \quad (3E.5)$$

Vemos que, ao contrário do algoritmo perceptron, podemos ter η constante (pertencendo ao intervalo $[0,3]$). Por exemplo, se $\eta=1$, teremos

$$e_g \simeq \frac{3}{2\alpha} = \frac{1.50}{\alpha} \quad (3E.6)$$

Existe porém um valor ótimo para η , dado por $\eta = I'^{-1} = 3/2$. Neste caso, que chamaremos de relaxação ótimo, o erro assintótico será

$$e_g(\text{ótimo}) \simeq \frac{4}{3\alpha} \simeq \frac{1.33}{\alpha} \quad (3E.7)$$

Concluimos que o algoritmo perceptron ótimo apresenta um desempenho assintótico melhor do que o algoritmo de relaxação ótimo para o caso de única apresentação dos exemplos.

APÊNDICE 4.A

Neste apêndice calculamos os termos $G_i = \lim_{n \rightarrow 0} G_i(n)$ na aproximação de réplicas simétricas.

Cálculo de G_3 : Temos

$$G_3(n) = -\frac{1}{2} \sum_{\nu=1}^n \ln \lambda_{\nu} \quad (4A.1)$$

onde λ_{ν} são os autovalores de M .

Na aproximação de réplicas simétricas podemos escrever

$$M = (1 + 2\beta(1-R))I + \tilde{M} \quad (4A.2)$$

onde todos os elementos não diagonais da matriz $\tilde{M}$ são iguais (os elementos diagonais são nulos)

$$\tilde{M}^{\nu\sigma} = \beta(1-2R+q) \quad (\nu \neq \sigma) \quad (4A.3)$$

Pode ser facilmente verificado que para n par a matriz $\tilde{M}$ possui um autovalor não degenerado

$$\tilde{\lambda}_1 = (n-1)\beta(1-2R+q) \quad (4A.4)$$

cujo autovetor é $(1,1,\dots,1)^t$, e $(n-1)$ autovalores degenerados

$$\tilde{\lambda}_{\nu} = \beta(1-2R+q) \quad (\nu=2,\dots,n) \quad (4A.5)$$

cujos autovetores possuem o mesmo número de componentes $+1$ e -1 . Considerando agora a matriz M , ela terá autovalores

$$\lambda_1 = 1 + 2\beta(1-R) + \tilde{\lambda}_1 = 1 + \beta(1-q) + n\beta(1-2R+q)$$

$$\lambda_{\nu} = 1 + 2\beta(1-R) + \tilde{\lambda}_{\nu} = 1 + \beta(1-q) \quad (\nu=2,\dots,n)$$

Temos então que

$$G_3(n) = \frac{1}{2} \sum_{\nu=1}^n \ln \lambda_{\nu} = \frac{1}{2} \ln \left[(1 + \beta(1-q)) \left(1 + \frac{n\beta(1-2R+q)}{1 + \beta(1-q)} \right) \right]$$

$$\begin{aligned}
& + \frac{1}{2} (n-1) \ln (1 + \beta(1-q)) \\
& = \frac{1}{2} \left(\ln \left(1 + \frac{n \beta(1-2R+q)}{1 + \beta(1-q)} \right) + n \ln (1 + \beta(1-q)) \right) \quad (4A.7)
\end{aligned}$$

No limite $n \rightarrow 0$ usamos a aproximação $\ln(1+x) \simeq x$, obtendo

$$G_3 = \lim_{n \rightarrow 0} \frac{1}{n} G_3(n) = \frac{1}{2} \left(\frac{\beta(1-2R+q)}{1 + \beta(1-q)} + \ln (1 + \beta(1-q)) \right) \quad (4A.8)$$

Cálculo de G_2 :

O procedimento para se calcular a integral em

$$G_2(n) = \ln \int_{-\infty}^{\infty} \left(\prod_{\nu=1}^n \frac{dJ^\nu}{\sqrt{2\pi}} \right) \exp \left(\sum_{\nu=1}^n K^\nu J^\nu - \frac{1}{2} \sum_{\nu=1}^n J^\nu (2E^\nu \delta^{\nu\sigma} - F^{\nu\sigma}) J^\sigma \right) \quad (4A.9)$$

consiste em completar o quadrado e efetuar a integração Gaussiana, o que fornece

$$G_2(n) = \frac{1}{2} \left(\frac{n K^2}{2E+F} - \ln \det L \right) \quad (4A.10)$$

onde

$$L^{\nu\sigma} = 2E^\nu \delta^{\nu\sigma} - F^{\nu\sigma} \quad (4A.11)$$

Assumindo simetria de réplicas e procedendo como no caso anterior encontramos (lembre que $F^{\nu\nu} = 0$),

$$\ln \det L = \sum_{\nu=1}^n \ln \lambda_\nu = n \left(\ln(2E+F) - \frac{F}{2E+F} \right) \quad (4A.12)$$

obtendo-se finalmente

$$G_2 = \lim_{n \rightarrow 0} \frac{1}{n} G_2(n) = \frac{1}{2} \left(\frac{K^2+F}{2E+F} - \ln (2E+F) \right) \quad (4A.13)$$

Cálculo de G_1 :

Finalmente, para réplicas simétricas escrevemos

$$\begin{aligned} G_1(n) &= \sum_{\nu=1}^n E^\nu - K^\nu R^\nu + \sum_{\nu < \sigma} \sum_{\sigma=1}^n F^{\nu\sigma} q^{\nu\sigma} \\ &= n(E - KR) - \frac{n(n-1)}{2} Fq \end{aligned} \quad (4A.14)$$

$$G_1 = \lim_{n \rightarrow 0} \frac{1}{n} G_1(n) = E - KR + Fq \quad (4A.15)$$

Note a mudança do sinal no último termo produzida por este limite.