INTRODUÇÃO AO ESTUDO DE REDES NEURAIS ARTIFICIAIS

MÓDULO REDES MULTI-LAYER PERCEPTRON E APRENDIZADO BACKPROPAGATION

Laboratório de Conexionismo e Ciências Cognitivas L3C
Grupo SICRES
INE - UFSC

Objetivo

- Oferecer ao aluno uma introdução à arquitetura de rede direta em camadas (multi-layer perceptron), descrevendo características de funcionamento, o aprendizado por retro-propagação de erros (backpropagation) e algumas variações, dicas e aplicações típicas.

Introdução

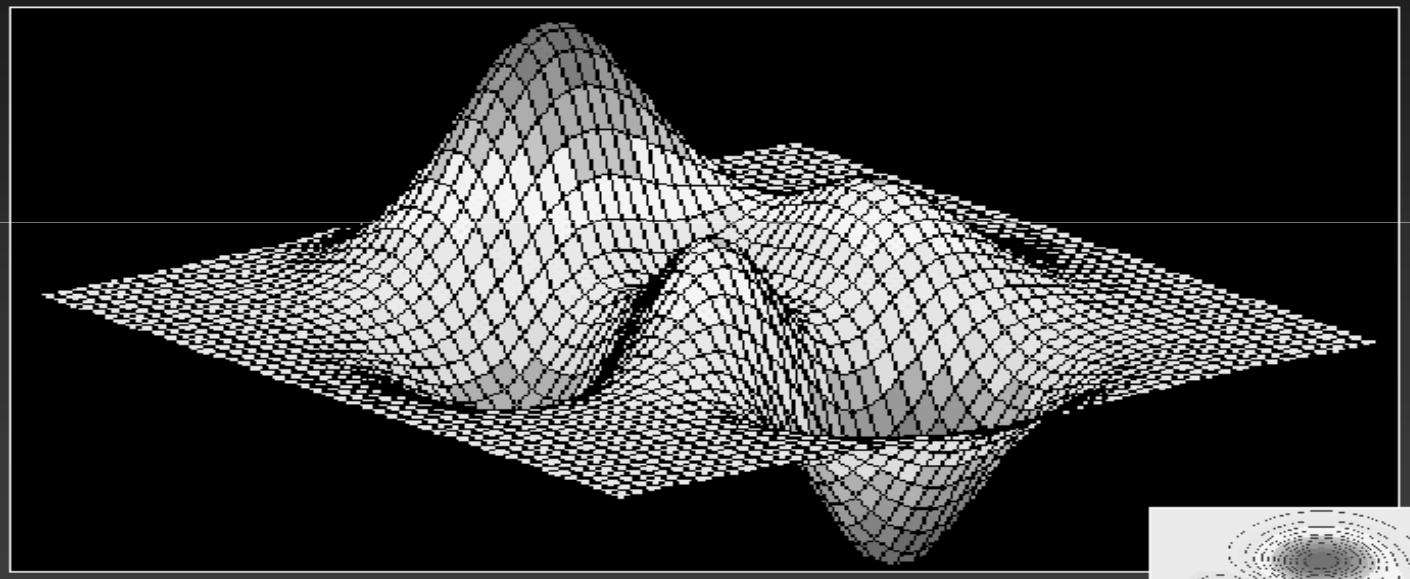
- Redes de apenas uma camada só representam funções linearmente separáveis
- Redes de múltiplas camadas solucionam essa restrição
  - Utilização de uma camada intermediária de neurônios, chamada Camada Intermediária (ou Escondida - "Hidden Layer"), de modo a poder implementar superfícies de decisão mais complexas.

Introdução

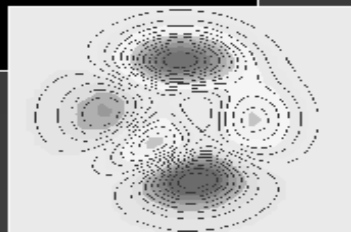
- Solução para superar o problema da classificação de padrões não-linearmente separáveis
  - Desvantagem em utilizar esta camada escondida: O aprendizado se torna muito mais difícil.
  - Nos anos 80, um algoritmo chamado Retro-propagação ou Backpropagation, veio fazer renascer o interesse geral pelas redes neurais.

Introdução

Um exemplo de Superfície Adaptativa



Vista superior →



Introdução

- Idéia Central:
  - Os erros dos elementos processadores da camada de saída (conhecidos pelo treinamento supervisionado) são retro-propagados para as camadas intermediárias.
- Utiliza o mesmo princípio da Regra Delta
  - A minimização de uma função custo, no caso, a soma dos erros médios quadráticos sobre um conjunto de treinamento, utilizando a técnica de busca do gradiente-descendente.

Introdução

- Também é chamado muitas vezes de Regra Delta Generalizada ("Generalized Delta-Rule").
  - A modificação principal em relação a Regra Delta foi a utilização de funções contínuas e suaves como função de saída dos neurônios ao invés da função de limiar lógico.
  - Como as funções de saída passaram a ser deriváveis, isto permitiu a utilização da busca do gradiente-descendente também para os elementos das camadas intermediárias.

Backpropagation

❑ CARACTERÍSTICAS BÁSICAS

- Regra de Propagação:
  - $net_j = \Sigma x_i.w_{ji} + \theta_j$
- Função de Ativação:
  - Função Não-Linear, diferenciável em todos os pontos.
- Topologia:
  - Múltiplas camadas.
- Algoritmo de Aprendizado:
  - Supervisionado
- Valores de Entrada/Saída:
  - Binários e/ou Contínuos

Backpropagation

- Processo de Aprendizado
  - Processo de minimização do erro quadrático pelo método do Gradiente Descendente

$$\Delta W_{ji} = -\eta \frac{\delta E}{\delta W_{ji}}$$

  - Cada peso sináptico i do elemento processador j é atualizado proporcionalmente ao negativo da derivada parcial do erro deste processador com relação ao peso.

Backpropagation

- Processo de Aprendizado
  - Basicamente, a rede aprende um conjunto pré-definido de pares de exemplos de entrada/saída em ciclos de propagação/adaptação.
  - Depois que um padrão de entrada foi aplicado como um estímulo aos elementos da primeira camada da rede, ele é propagado por cada uma das outras camadas até que a saída seja gerada. Este padrão de saída é então comparado com a saída desejada e um sinal de erro é calculado para cada elemento de saída.

Backpropagation

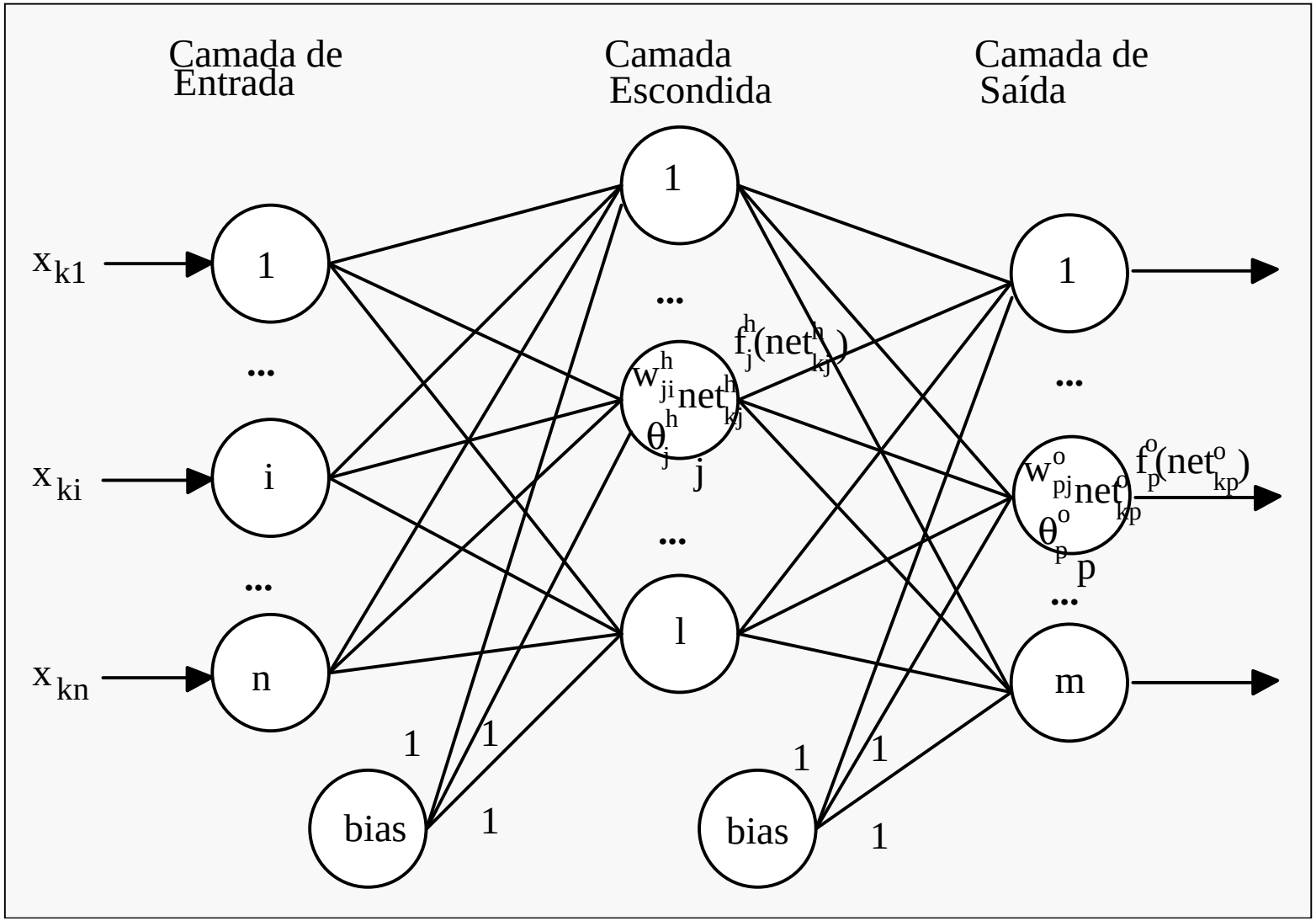
- Processo de Aprendizado
  - O sinal de erro é então retro-propagado da camada de saída para cada elemento da camada intermediária anterior que contribui diretamente para a formação da saída.
  - Cada elemento da camada intermediária recebe apenas uma porção do sinal de erro total, proporcional apenas à contribuição relativa de cada elemento na formação da saída original.

Backpropagation

- Processo de Aprendizado
  - Este processo se repete, camada por camada, até que cada elemento da rede receba um sinal de erro que descreva sua contribuição relativa para o erro total.
  - Baseado no sinal de erro recebido, os pesos das conexões são então atualizados para cada elemento de modo a fazer a rede convergir para um estado que permita a codificação de todos os padrões do conjunto de treinamento.

Backpropagation

- Processo de Aprendizado



Backpropagation

- Processo de Aprendizado
  - Suponhamos que tenhamos um conjunto de P pares de vetores $(\mathbf{X}_1,\mathbf{Y}_1)$, $(\mathbf{X}_2,\mathbf{Y}_2)$, ..., $(\mathbf{X}_P, \mathbf{Y}_P)$, no nosso conjunto de treinamento e que são exemplos de um mapeamento funcional definido como:

$$Y = \theta(X) \quad : \quad X \in \Re^n, Y \in \Re^m$$

  - Desejamos treinar a rede de modo que ela consiga aprender uma aproximação da forma:

$$O = Y' = \theta'(X) \quad : \quad X \in \Re^n, Y' \in \Re^m$$

Backpropagation

- Etapas
  - 1. Aplicar um vetor de entrada do conjunto de treinamento e propagar até a saída
    - Um vetor de entrada $\mathbf{X}_k=[x_{k1}\ x_{k2}\ \ldots\ x_{kn}]^T$ do conjunto de treinamento é apresentado à camada de entrada da rede. Os elementos de entrada distribuem os valores para os elementos da camada escondida. O valor do net para o j*ésimo* elemento da camada escondida vale:

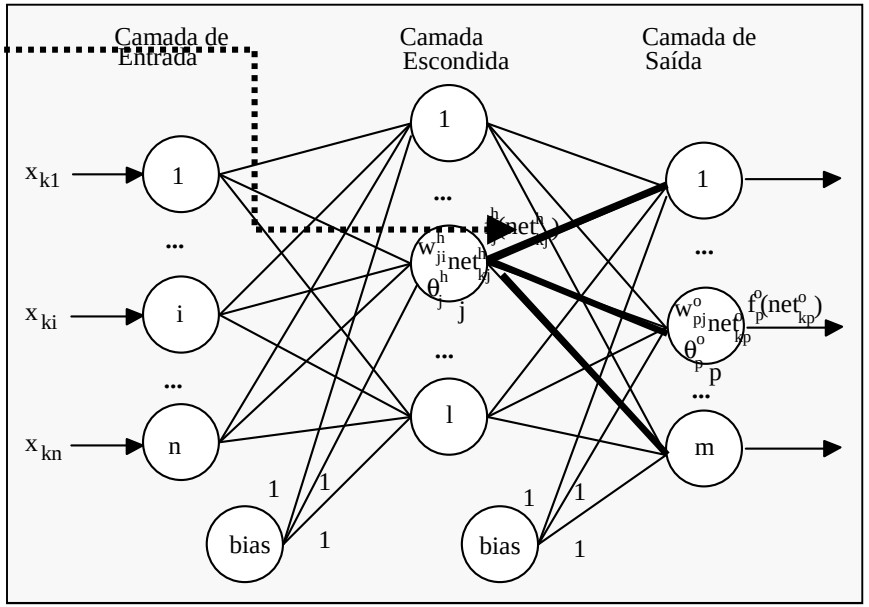
$$net_{kj}^{h} = \sum_{i=1}^{n} w_{ji}^{h} x_{ki} + \theta_{j}^{h}$$

    - onde w_{ji} é o peso da conexão entre o i*ésimo* elemento da camada de entrada e o j*ésimo* elemento da camada escondida *h*.

Backpropagation

- Etapas
  - 1. Aplicar um vetor de entrada do conjunto de treinamento e propagar até a saída
    - Como os neurônios são estáticos, o valor de saída para um neurônio da cada escondida vale:

$$i_{kj} = f_j^h(net_{kj}^h)$$

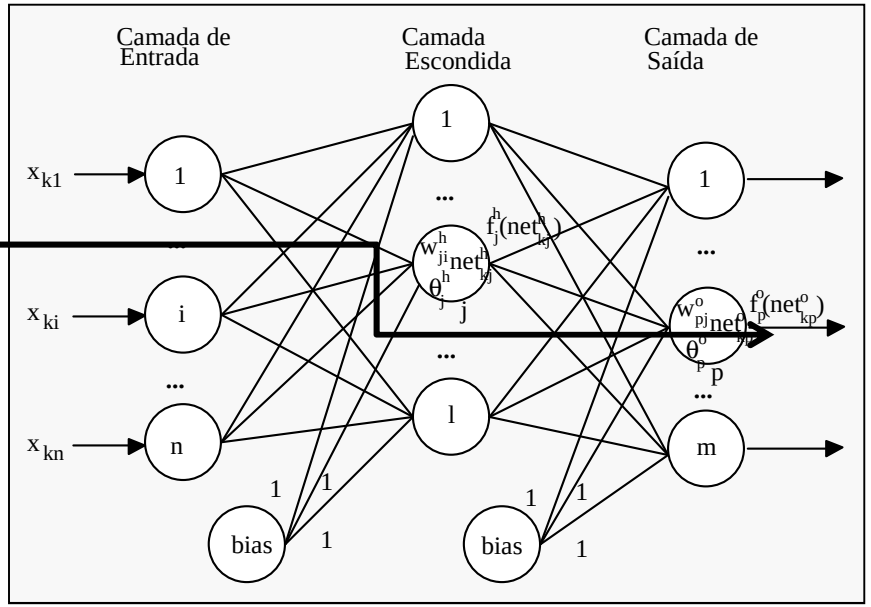


Backpropagation

- Etapas
  - 1. Aplicar um vetor de entrada do conjunto de treinamento e propagar até a saída
    - Do mesmo modo, as equações para os neurônios da camada de saída são:

$$net_{kp}^{o} = \sum_{j=1}^{l} w_{pi}^{o} i_{kj} + \theta_{p}^{o}$$

$$o_{kp} = f_{p}^{o}(net_{kp}^{o})$$



Backpropagation

- Etapas
  - 2. Calcular o erro entre a saída calculada pela rede e a saída desejada no conjunto de treinamento.
    - Definimos o erro para um único neurônio *p* na camada de saída para um vetor de entrada *k* como sendo:

$$E_{kp} = (y_{kp} - o_{kp})$$

    - E o erro a ser minimizado pelo algoritmo para todos os neurônios da camada de saída

$$E_k = \frac{1}{2}\sum_{p=1}^{m} E_{kp}^2$$

$$E_k = \frac{1}{2}\sum_{p=1}^{m} (y_{kp} - o_{kp})^2$$

Backpropagation

- Etapas
  - 3. Determinar em que direção a mudança de peso deverá ocorrer.
    - Para determinar a direção da modificação dos pesos, calculamos o negativo do gradiente de E_k, ∇E_k, com relação aos pesos w_{pj}

$$\frac{\partial E_k}{\partial w_{pj}^o} = -(y_{kp} - o_{kp}) \frac{\partial f_p^o}{\partial (net_{kp}^o)} \frac{\partial (net_{kp}^o)}{\partial w_{pj}^o}$$

    - Podemos escrever a derivada de f_p^o como:

$$f_p^{o'}(net_{kp}^o)$$

    - e o último termo da equação

$$\frac{\partial (net_{kp}^o)}{\partial w_{pj}^o} = (\frac{\partial}{\partial w_{pj}^o} \sum_{j=1}^{l} w_{pj}^o i_{kj} + \theta_p^o) = i_{kj}$$

Backpropagation

- Etapas
  - 3. Determinar em que direção a mudança de peso deverá ocorrer.
    - Combinando as equações, temos para o negativo do gradiente:

$$-\frac{\partial E_k}{\partial w_{pj}^o} = (y_{kp} - o_{kp}) f_p^{o'}(net_{kp}^o) i_{kj}$$

Backpropagation

- Etapas
  - 4. Determinar o valor da mudança de cada peso.
    - A atualização dos pesos dos neurônios da camada de saída se faz por:

$$w_{pj}^{o}(t+1) = w_{pj}^{o}(t) + \Delta_k w_{pj}^{o}(t)$$

$$\Delta_k w_{pj}^{o} = \eta\,(y_{kp} - o_{kp})\, f_p^{o'}(net_{kp}^{o})\, i_{kj}$$

    - onde η é a TAXA DE APRENDIZADO.

Backpropagation

- Etapas
  - AS FUNÇÕES DE SAÍDA -para que possamos implementar a busca do gradiente-descendente, é necessário que f_p^o seja diferenciável.
    - a função linear:

$$f_p^o(net_{jp}^o) = net_{jp}^o \qquad f_p^{o'} = 1$$

    - a função logística ou sigmoidal:

$$f_p^o(net_{jp}^o) = \frac{1}{1 + e^{-net_{jp}^o}} \qquad f_p^{o'} = f_p^o.(1 - f_p^o)$$

    - a função tangente hiperbólica:

$$f_p^o(net_{jp}^o) = tanh(net_{jp}^o) = \frac{1 - e^{-net_{jp}^o}}{1 + e^{-net_{jp}^o}} \quad f_p^{o'} = \frac{1}{2}(1 - f_p^{o\,2})$$

Backpropagation

- Etapas
  - 5. Repetir os procedimentos para os pesos da Camada Intermediária.
    - Desejamos repetir para a camada escondida os mesmos tipos de cálculos que realizamos para a camada de saída.
    - O problema aparece quando tentamos determinar uma medida para o erro das saídas dos neurônios da camada escondida.
    - Sabemos qual é a saída destes neurônios calculada pela rede, porém não sabemos a priori qual deveria ser a saída correta para estes elementos. Intuitivamente, o erro total, E_k, deve de alguma forma estar relacionado com o valor de saída dos neurônios da camada escondida.

Backpropagation

- Etapas
  - 5. Repetir os procedimentos para os pesos da Camada Intermediária.
    - Voltando a equação do Erro temos:

$$E_k = \frac{1}{2}\sum_p (y_{kp} - o_{kp})^2$$

$$= \frac{1}{2}\sum_p (y_{kp} - f_p^o(net_{kp}^o))^2$$

$$= \frac{1}{2}\sum_p (y_{kp} - f_p^o(\sum_j w_{pj}^o i_{kj} + \theta_p^o))^2$$

Backpropagation

- Etapas
  - 5. Repetir os procedimentos para os pesos da Camada Intermediária.
    - Sabendo que i_{pj} depende dos pesos da camada escondida, podemos utilizar este fato para calcular o gradiente de E_k com respeito aos pesos da camada escondida.

$$\frac{\partial E_k}{\partial w_{ji}^h} = \frac{1}{2}\sum_p \frac{\partial}{\partial w_{ji}^h}(y_{kp} - o_{kp})^2$$

$$= -\sum_p (y_{kp} - o_{kp}) \frac{\partial o_{\mathrm{kp}}}{\partial (net_{kp}^o)} \frac{\partial (net_{kp}^o)}{\partial i_{\mathrm{kj}}} \frac{\partial i_{\mathrm{kj}}}{\partial (net_{kj}^h)} \frac{\partial (net_{kj}^h)}{\partial w_{ji}^h}$$

Backpropagation

- Etapas
  - 5. Repetir os procedimentos para os pesos da Camada Intermediária.
    - Cada um dos fatores da equação pode ser calculado explicitamente das equações anteriores, assim como foi feito para o gradiente da camada de saída.
    - O resultado fica:

$$\frac{\partial E_k}{\partial w_{ji}^h} = -\sum_p (y_{kp} - o_{kp}) f_p^{o'}(net_{kp}^o) w_{pj}^o f_j^{h'}(net_{kj}^h) x_{ki}$$

Backpropagation

- Etapas
  - 5. Repetir os procedimentos para os pesos da Camada Intermediária.
    - Por fim, assim como no caso da camada de saída, atualizamos os pesos da camada escondida proporcionalmente ao valor negativo da equação.

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \Delta_k w_{ji}^{h}(t)$$

    - Onde:

$$\Delta_k w_{ji}^{h} = \eta f_j^{h'}(net_{kj}^{h}) x_{ki} \sum_p (y_{kp} - o_{kp}) f_p^{o'}(net_{kp}^{o}) w_{pj}^{o}$$

    - η novamente é a taxa de aprendizado.

Backpropagation

- Etapas
  - 6. Voltar ao passo 1, escolhendo um novo vetor de entrada do conjunto de treinamento e repetir os passos de 1 a 5, somando o erro.
  - 7. Após todos os vetores do entrada do conjunto de treinamento terem sido apresentados (uma "ÉPOCA"), calcular o erro médio quadrático (MSE).

    Se for aceitável parar,

    senão voltar ao passo 1.

Backpropagation

- Este procedimento de aprendizado é repetido diversas vezes, até que, para todos os processadores da camada de saída e para todos os padrões de treinamento, o erro seja menor do que o especificado.
- Em geral a avaliação se dá pelo ERRO MÉDIO QUADRÁTICO
- Mais adiante veremos critérios de parada do algoritmo.

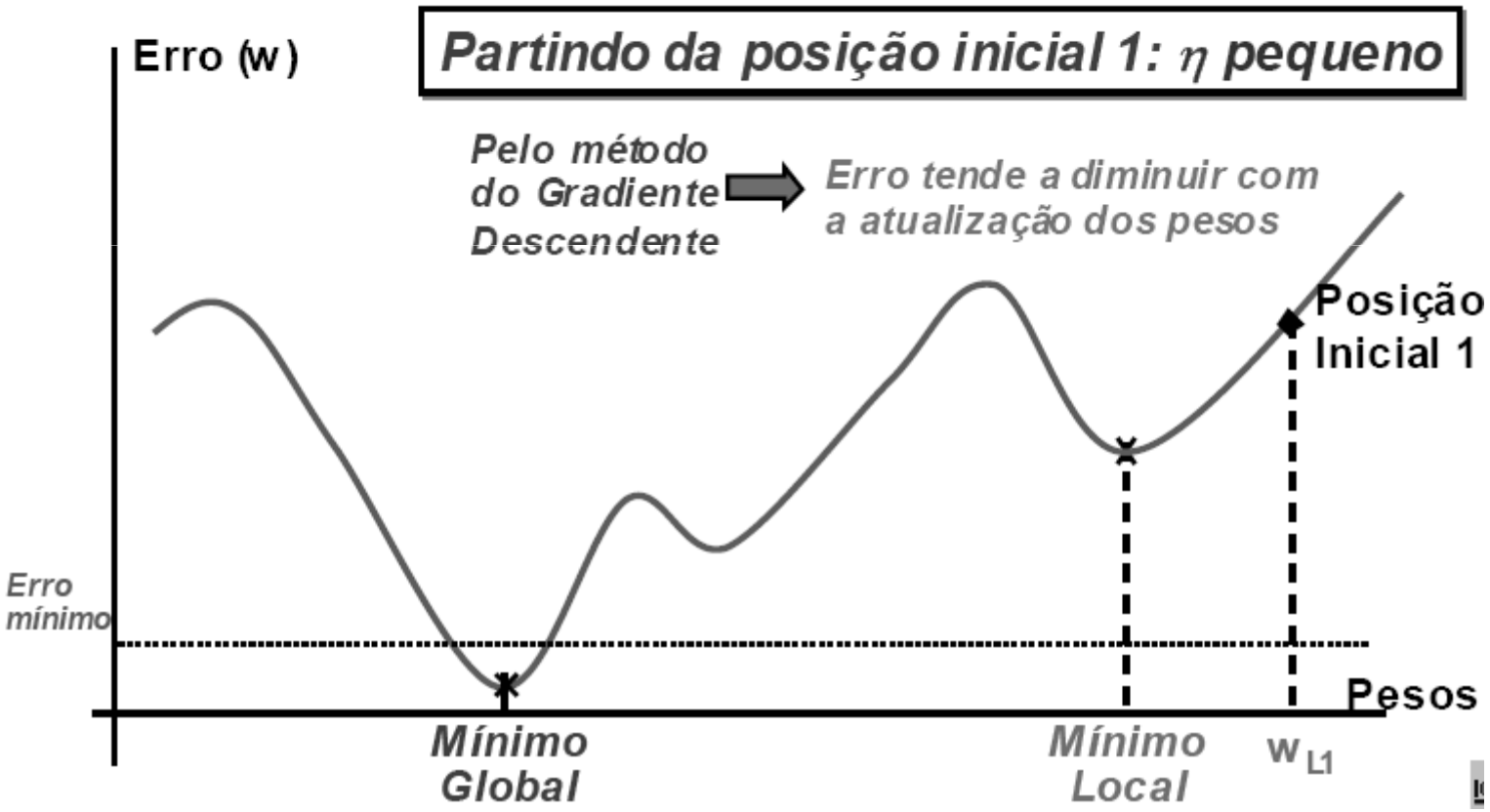
Backpropagation

- Foi demonstrado (Cybenko 1989) que a rede MLP é um Aproximador Universal, isto é, pode representar qualquer função contínua com o grau de precisão desejado.

- É o algoritmo de Redes Neurais mais utilizado em aplicações práticas de previsão, classificação e reconhecimento de padrões em geral.

Backpropagation

- O Problema do Mínimo Local



Backpropagation

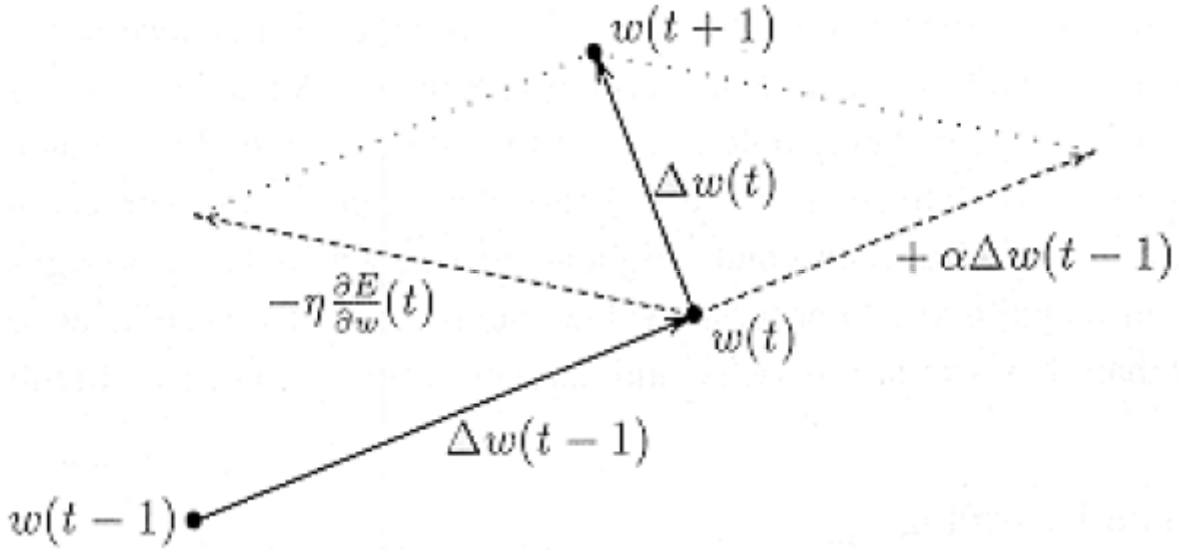
- Como escapar de mínimos locais?
  - Podemos escapar de mínimos locais usando na atualização dos pesos um termo proporcional a última direção de alteração do peso. (Alteração do Peso no passo anterior do algoritmo Backpropagation) - idéia de inércia ou um "empurrão" para sair de buracos.

$$w_{pj}^{o}(t+1) = w_{pj}^{o}(t) + \Delta_k w_{pj}^{o}(t) + \alpha \Delta_k w_{pj}^{o}(t-1)$$

  - onde α é o parâmetro conhecido como "momento".

Backpropagation

- O Termo de Momento



-1)

Backpropagation

- Formas de Aprendizado
  - Existe dois métodos básicos de aplicação do algoritmo Backpropagation:
    - Aprendizado em Batch ("Batch Learning", por ciclo, por época, etc)
    - Aprendizado Incremental ("on-line", "pattern-mode", por padrão, etc)

Backpropagation

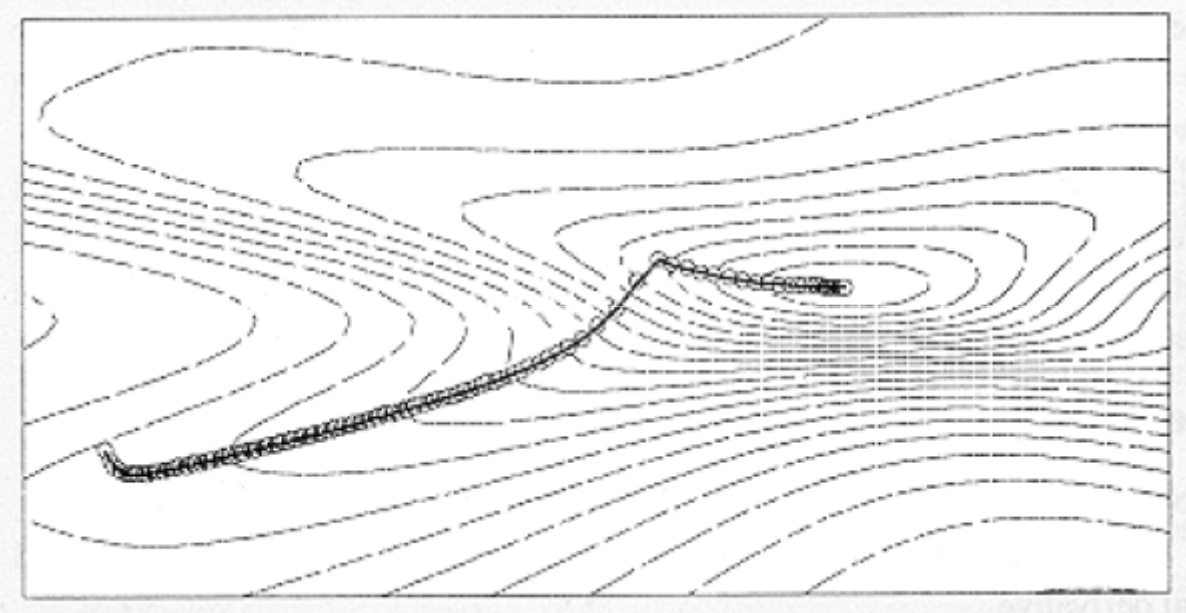
- Aprendizado em "Batch"
  - Somente ajusta os pesos após a apresentação de TODOS os padrões
  - Cada padrão é avaliado com a MESMA configuração de pesos
  - Obtém-se os termos derivativos δEp/δw e depois obtém-se a soma total do algoritmo:

$$\frac{\partial E}{\partial w} = \sum_{p} \frac{\partial E_p}{\partial w}$$

  - Cálculo correto do gradiente.

Backpropagation

- Aprendizado em "Batch"

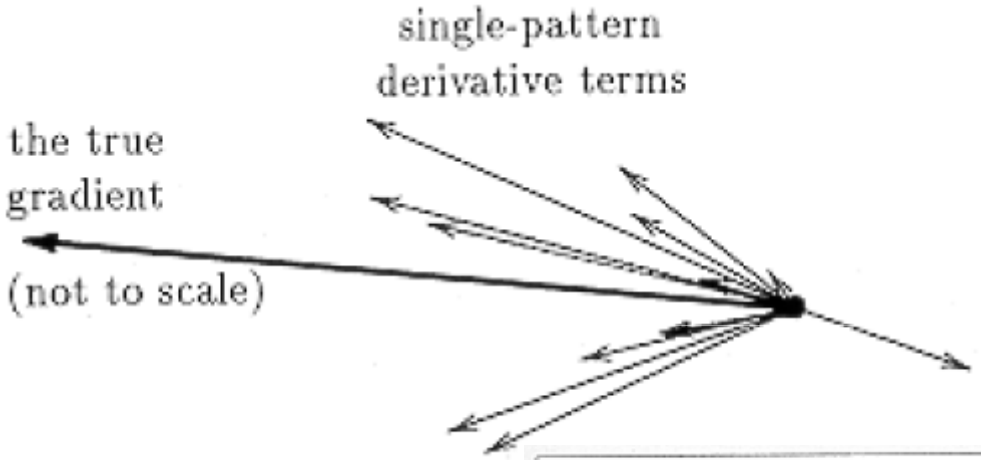


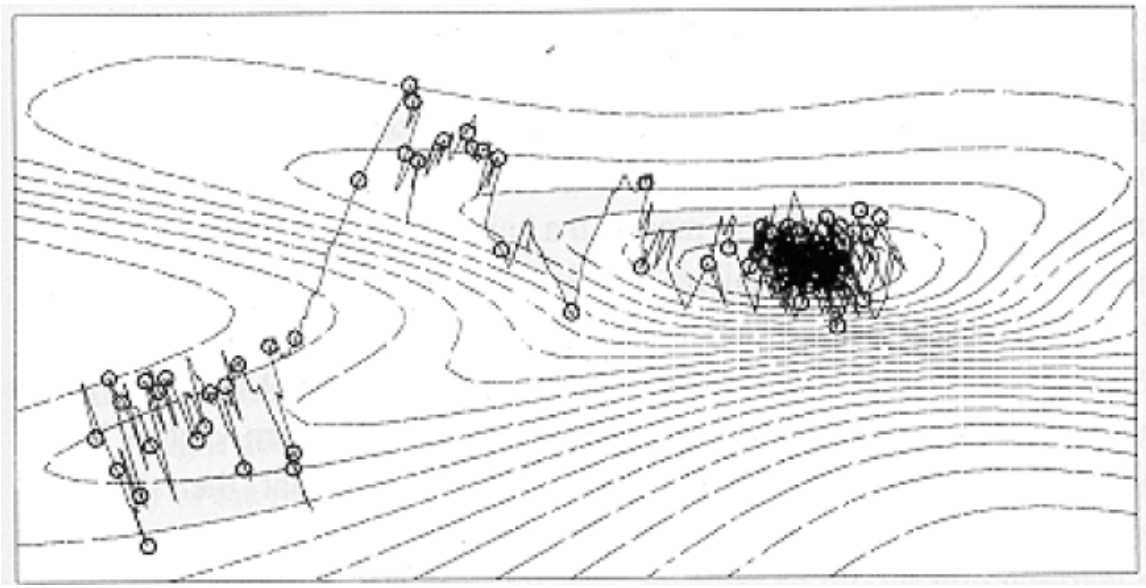
Backpropagation

- Aprendizado Incremental
  - Atualiza os pesos a cada apresentação de um novo padrão
  - Os pesos são atualizados usando o gradiente do erro de um único padrão
  - O gradiente de um único padrão pode ser visto como uma estimativa ruidosa do verdadeiro gradiente;
  - Ele pode ter projeções negativas sobre o verdadeiro gradiente;
  - Na média, ele se move "downhill";

Backpropagation

- Aprendizado Incremental



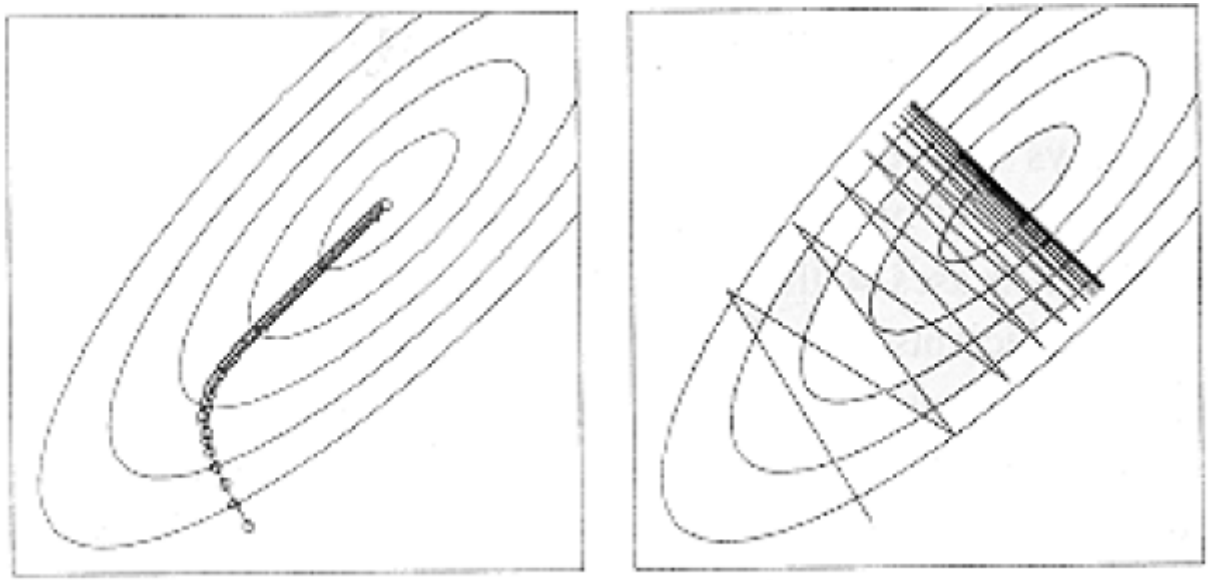


Backpropagation

- Batch X Incremental
  - O modo Batch necessita de menos atualizações de pesos →Tende a ser mais rápido
  - Batch fornece uma medida mais precisa da mudança necessária dos pesos
  - Batch necessita de mais memória
  - Incremental tem menos chance de ficar preso em um mínimo local devido à apresentação aleatória dos padrões → natureza estocástica de busca no espaço de pesos
  - Tende a ser mais rápido se o conjunto de treinamento for grande e ruidoso.

Backpropagation

- Inicialização dos Pesos
  - Aleatória entre -0.1 e +0.1 (Problema de paralisia da rede)
- Taxa de Aprendizado
  - Valores pequenos: 0.05 a 0.1



(a) small step-size (b) large step-size

Backpropagation

- Taxa de Momento
  - Valores grandes: 0.8 a 0.9
- Se a função de saída for sigmoidal, escalar os valores de saída.
  - As saídas nunca atingem exatamente 0 ou 1. Usar valores como 0.1 e 0.9 para representar o menor e o maior valor de saída.
- Se a função de saída for tangente hiperbólica, escalar os valores de saída.
  - As saídas nunca atingem exatamente -1 ou 1. Usar valores como -0.9 e 0.9 para representar o menor e o maior valor de saída.

Backpropagation

- Normalização dos Dados de Entrada
  - O algoritmo funciona bem para valores de entrada no intervalo [-1,+1].
  - Se a distribuição for uniforme, os dados podem ser normalizados usando uma função linear.
  - Se a distribuição for não-uniforme, pode-se normalizar usando uma função não-linear:
    - Gaussiana
    - Logarítmica

Backpropagation

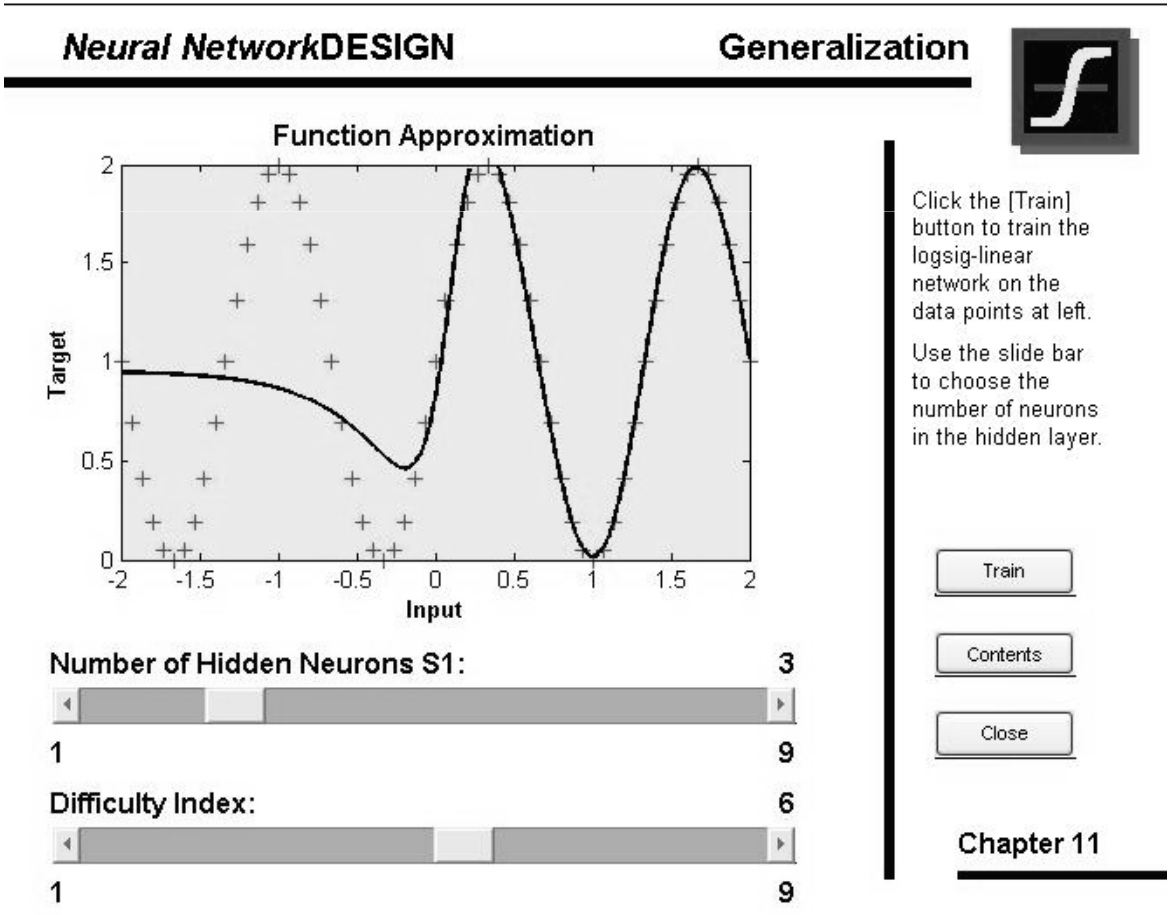
- Tamanho da Rede
  - Compromisso entre Convergência e Generalização.
  - Também conhecido como "bias and variance dilemma".

Backpropagation

- Tamanho da Rede
  - Convergência
    - É a capacidade da Rede Neural de aprender todos os padrões do conjunto de treinamento.
    - Se a rede neural for pequena, não será capaz de armazenar todos os padrões necessários.
    - Isto é, a rede não deve ser rígida a ponto de não modelar fielmente os dados.

Backpropagation

- Tamanho da Rede
  - Convergência

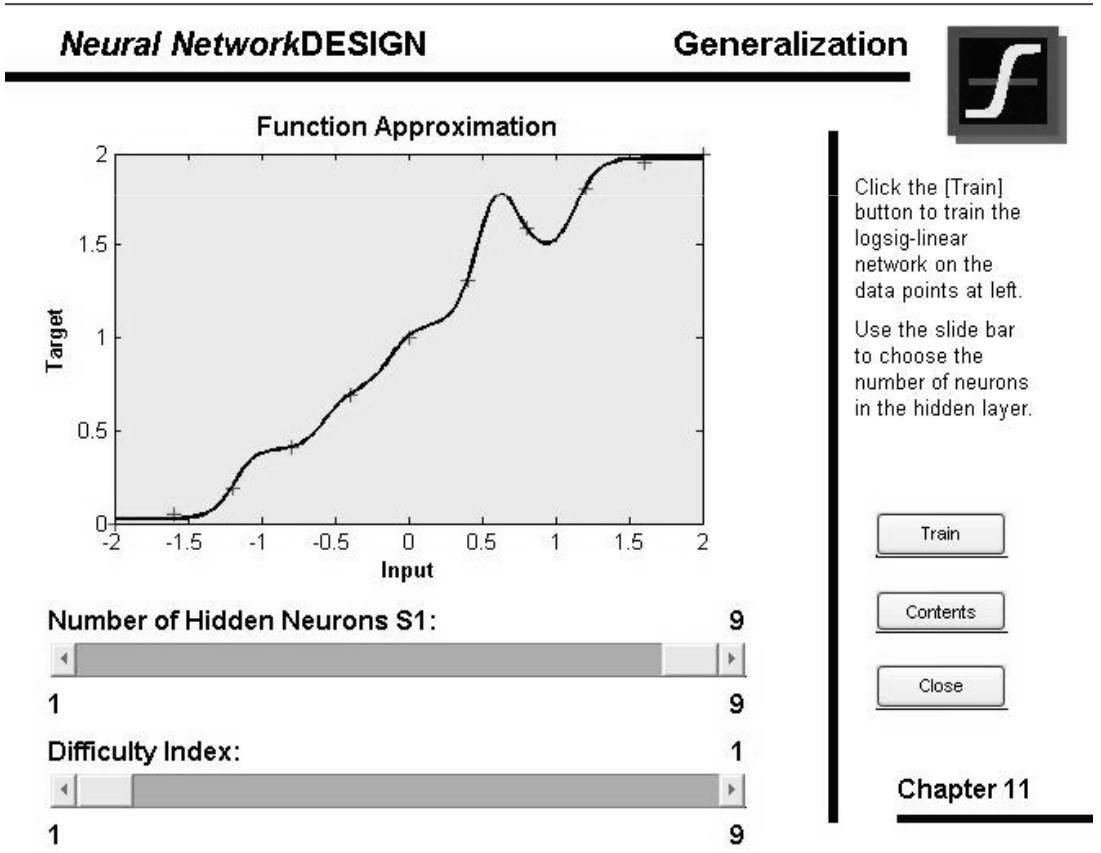


Backpropagation

- Tamanho da Rede
  - Generalização
    - Se a rede for muito grande (muitos parâmetros = pesos), não responderá corretamente aos padrões nunca vistos.
    - Isto é, a rede não deve ser excessivamente flexível a ponto de modelar também o ruído

Backpropagation

- Tamanho da Rede
  - Generalização



Backpropagation

- Técnicas para Reduzir a complexidade da rede:
  - Weight Decay
  - Análise de Sensibilidade

Backpropagation

- Técnicas para Reduzir a complexidade da rede:
  - Weight Decay
    - Adiciona um termo denominado weight decay
    - Usado para ajustar a complexidade da rede à dificuldade do problema;
    - Se a estrutura da rede for muito complexa, pode-se remover alguns pesos sem aumentar o erro significativamente;
    - O método fornece aos pesos uma tendência de se dirigir ao valor zero, reduzindo a sua magnitude um pouco a cada iteração.

Backpropagation

- Técnicas para Reduzir a complexidade da rede:
  - Weight Decay
    - Adiciona um termo denominado weight decay

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w}(t) - \rho w(t)$$

    - onde $0 \leq \rho << 1$
    - Equivale a modificar a definição de E incluindo um termo de penalidade correspondente à magnitude dos pesos da rede.

$$E = \frac{1}{2}\sum_{i=1}^{k}(d_i - y_i)^2 + \frac{1}{2}\lambda\|w\|^2$$

Backpropagation

- Técnicas para Reduzir a complexidade da rede:
  - Análise de Sensibilidade
    - Retira-se uma variável de entrada ou elementos processadores;
    - Treina-se novamente a rede;
    - Verifica-se e variação do erro de saída.
    - Se a retirada do elemento não causar grande variação, então a rede é pouco sensível a este elemento, podendo ser retirado sem perda da capacidade de modelar os dados.

Backpropagation

- Critérios de Parada do Treinamento
  - Até que o erro de treinamento seja inferior a um certo valor especificado.
    - Se este valor for muito pequeno: super treinamento ("overfitting").
  - super treinamento ("overfitting"):
    - os pesos estão sendo modificados para ajustar também o ruído existente nos padrões de treinamento;
    - quanto maior o número de pesos, maior o problema de overfitting;
    - Deve-se verificar o comportamento na generalização

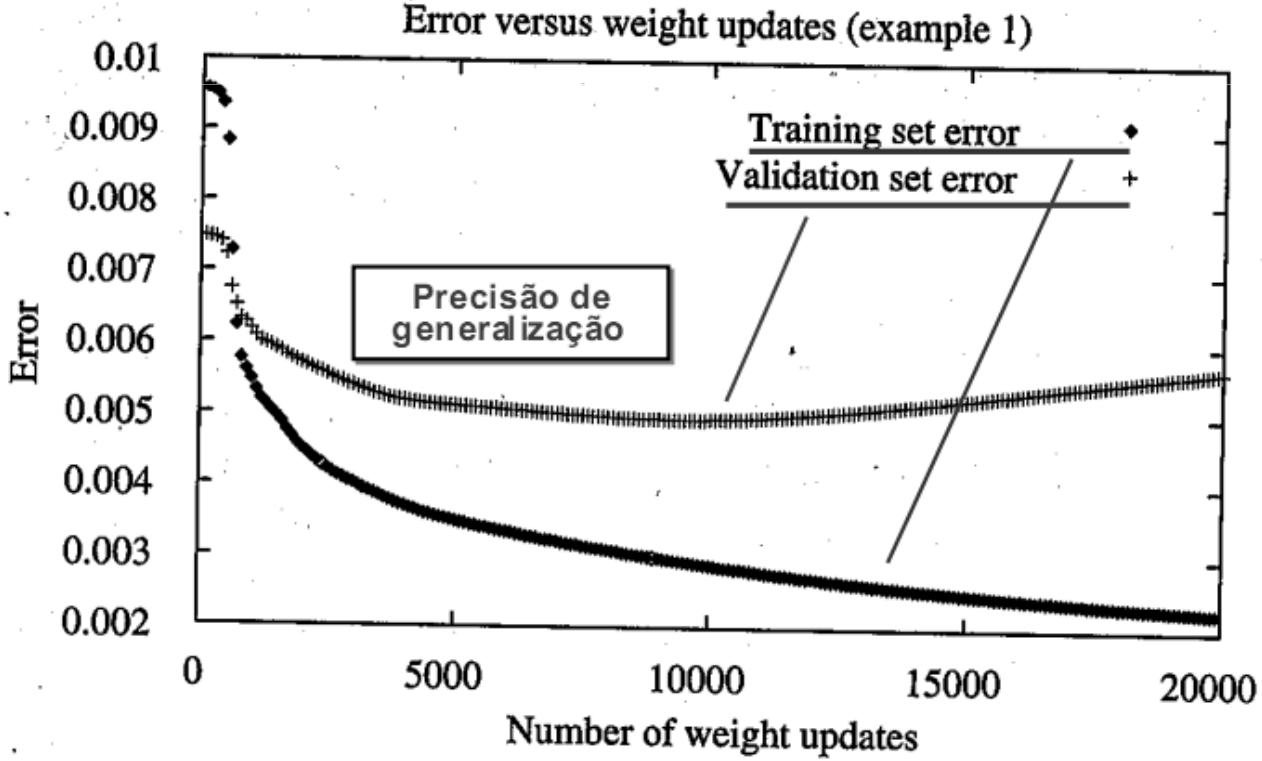
Backpropagation

- Critérios de Parada do Treinamento
  - **Validação Cruzada**
  - Deve-se dividir os padrões em três conjuntos:
    - Treinamento: padrões usados para usados para modificar os pesos; (60%-70%)
    - Validação: padrões usados para verificar o problema de overfitting; (15%-20%)
    - Teste: padrões para testar o desempenho do modelo final. (15-20%)

    - Obs.: Muitas vezes o conjunto de validação e teste são compostos pelos mesmos dados

Backpropagation

- Critérios de Parada do Treinamento
  - O treinamento deve ser interrompido quando o erro na validação começar a subir de forma consistente.



Backpropagation

- Validação Cruzada: O que fazer quando o conjunto de padrões é pequeno?
  - Particione os m padrões disponíveis para treinamento e validação em k conjuntos disjuntos, cada um com m/k padrões;
  - Efetue o procedimento de cross-validation k vezes, cada vez utilizando um conjunto diferente para efetuar a validação e os outros k-1 para fazer o treinamento;
  - Determine para cada um o número ideal i de iterações;
  - Calcula-se a média das iterações i_m nos k experimentos;
  - Executa-se um treinamento final, com os m padrões (sem padrões para validação) por i_m iterações.

Variações do Backpropagation

- A principal crítica ao algoritmo Backpropagation é o longo tempo de treinamento.
- Existem uma série de proposta para acelerar o processo de aprendizado:
  - Usando Heurísticas
    - Taxa de Aprendizagem Adaptativa
    - Resilient Backpropagation
  - Usando Métodos de Otimização Numérica
    - Métodos de Newton
    - Levenberg-Marquadt

Variações do Backpropagation

- Taxa de Aprendizagem Adaptativa
  - Aumentar η quando o erro decresce de forma consistente e diminuir η quando o erro aumenta consideravelmente.
  - Pequenos aumentos no erro são tolerados.
  - Taxa de aprendizado adaptativa, onde η(t) (global) no tempo t é determinada por:

$$\eta(t) = \begin{cases} \phi\eta(t-1) & se\ E(t) < E(t-1) \\ \beta\eta(t-1) & se\ E(t) > 1.05E(t-1) \\ \eta(t-1) & caso\ contrário \end{cases}$$

  - onde φ>1 e β<1 são constantes φ = 1.05 e β = 0.7

Variações do Backpropagation

- Taxa de Aprendizagem Adaptativa
  - Além de atualizar a taxa de aprendizado quando o erro cresce:
    - a atualização anterior nos pesos é desfeita e a taxa de momento é colocada em zero para a próxima iteração (isto evita que a atualização do peso vá para a mesma direção da iteração anterior);
    - após uma iteração bem sucedida (o valor do erro diminui), a taxa de momento volta ao valor anterior.

Variações do Backpropagation

- Resilient Backpropagation
  - MLP normalmente usa função sigmóide. Essas funções se caracterizam pelo fato da derivada para argumentos grandes ser muito pequena.
  - Isso causa morosidade na velocidade de convergência quando os argumentos são grandes e ainda a rede estiver longe de valores ótimos.
  - O Resilient Backpropagation elimina esse efeito. Somente o sinal da derivada é usado para determinar a direção da atualização do peso.
  - A quantidade da atualização é determinado de outra forma.

Variações do Backpropagation

- Resilient Backpropagation
  - O valor da atualização para pesos e bias é incrementado de um fator delt_inc sempre que a derivada for na mesma direção para duas iterações sucessivas.
  - O valor da atualização é decrementado por um fator delt_dec sempre que a derivada muda de sinal em relação a iteração anterior.
  - Sempre que os pesos estiverem oscilando a mudança nos pesos é reduzida.
  - Se a derivada é zero, o valor da atualização permanece o mesmo.

Variações do Backpropagation

- Usando Métodos de Otimização Numérica
  - Algoritmos de Segunda Ordem
    - Utilizam informações sobre a derivada segunda do erro em função dos pesos.
    - Procuram aproximar a superfície do erro por uma aproximação quadrática em torno do valor atual de W.

FIM DO MÓDULO REDE MLP E APRENDIZADO BACKPROPAGATION