Redes Neurais Artificiais

Multi-layer Perceptrons e Backpropagation

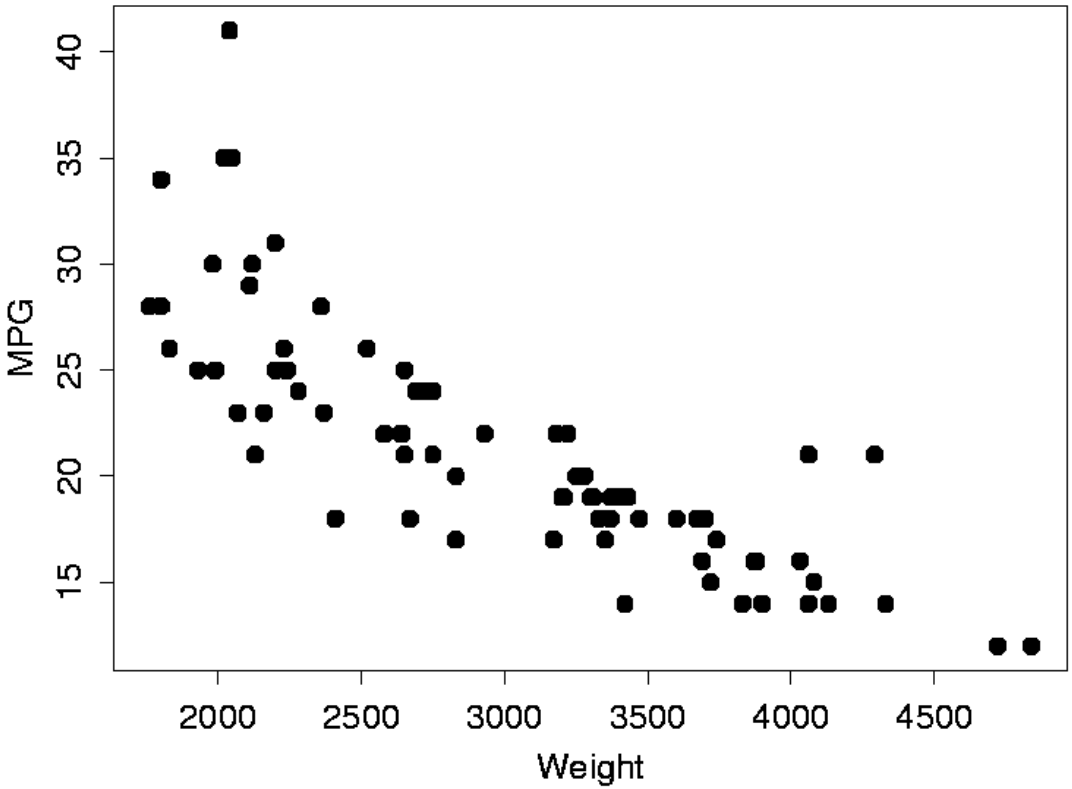
Marcílio Souto

DIMAp/UFRN

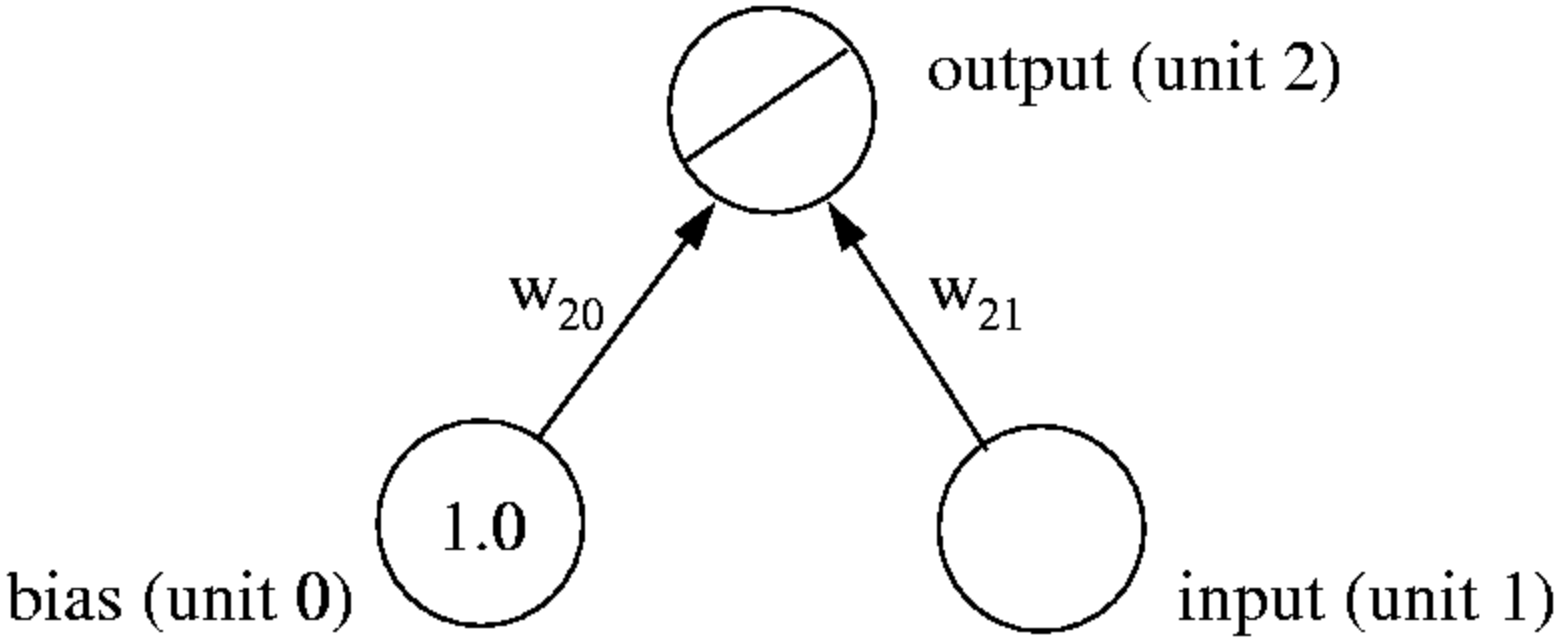
Redes de várias camadas

- **MLP - Multi-Layer Perceptron (Adaline?!)**
- **Backpropagation network**
- **Superar as limitações das redes de camada única**
  - **Regressão linear**
  - **Classificação de problemas linearmente separáveis**

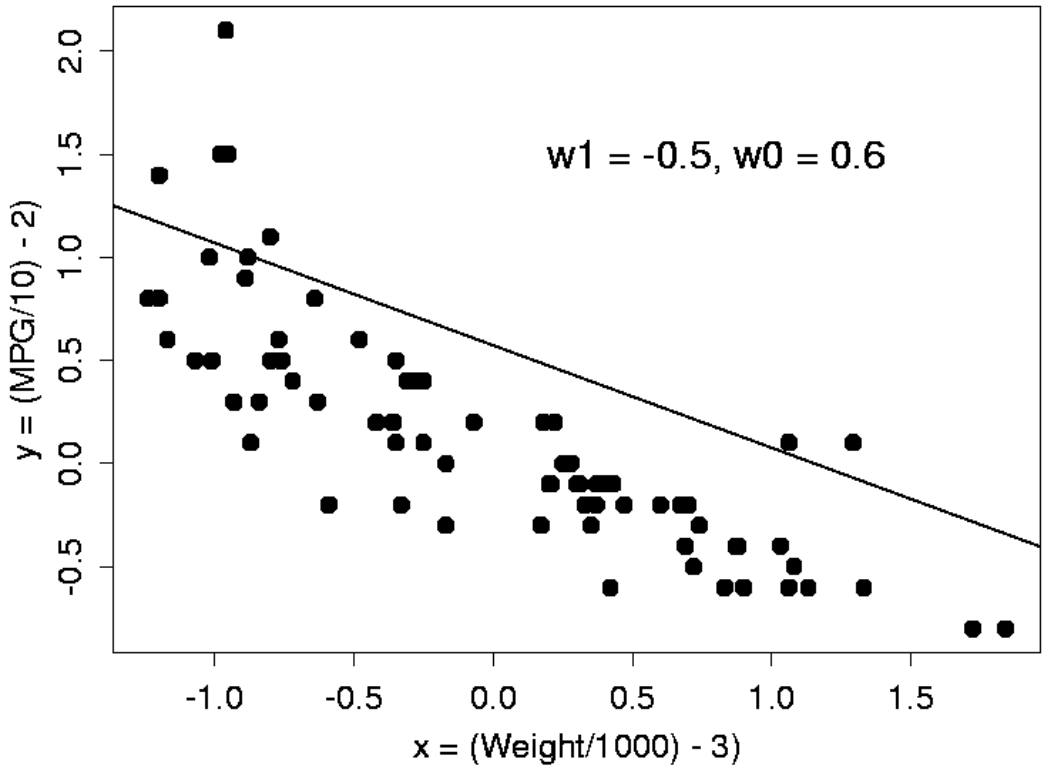
Regressão não linear



Adaline



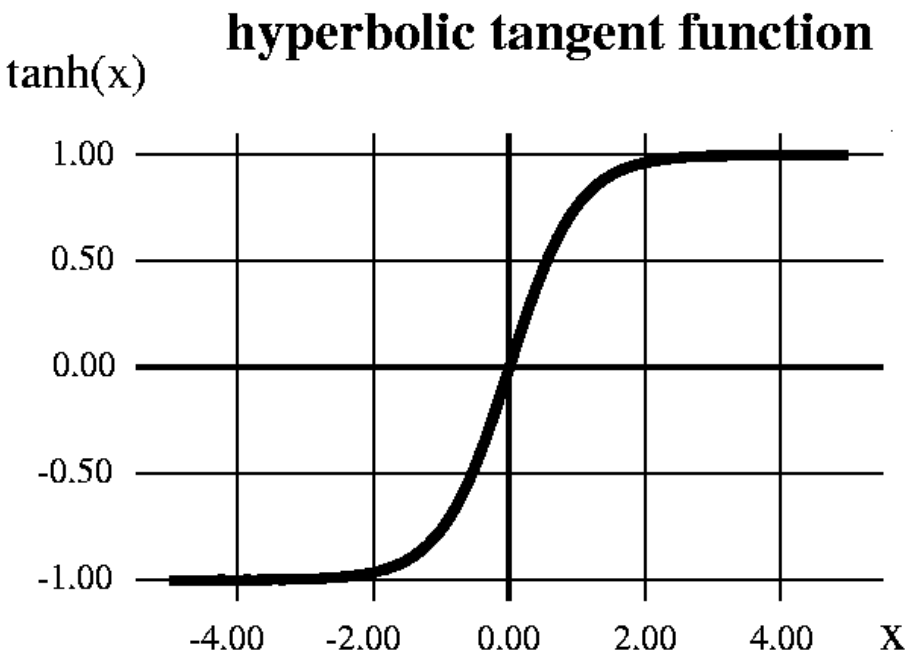
Linear x Não-linear

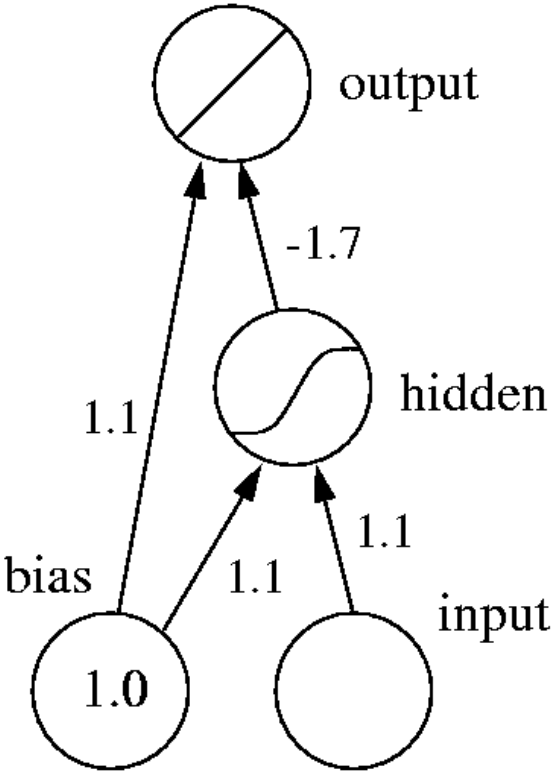


MLPs e não-linearidade (1/3)

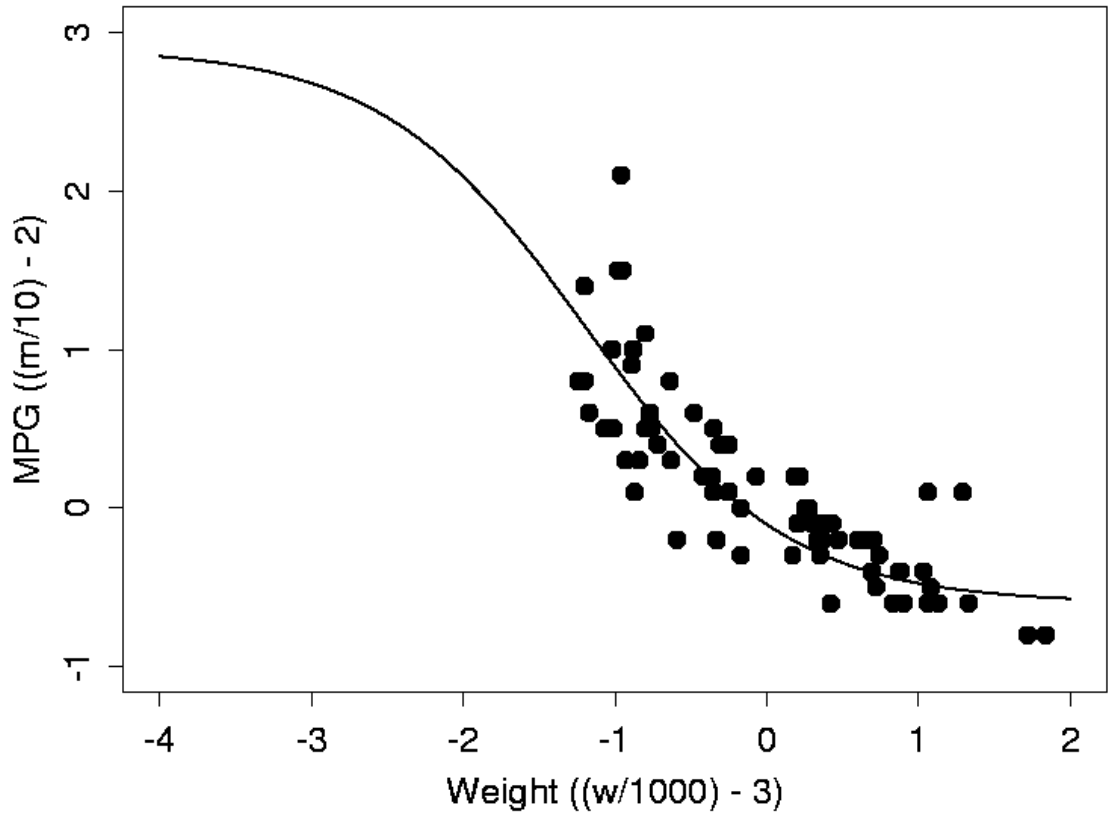
- **Adicione um nó com uma função de ativação não-linear**
  - **e.g., a tangente hiperbólica (*tanh*)**

MLPs e não-linearidade (2/3)



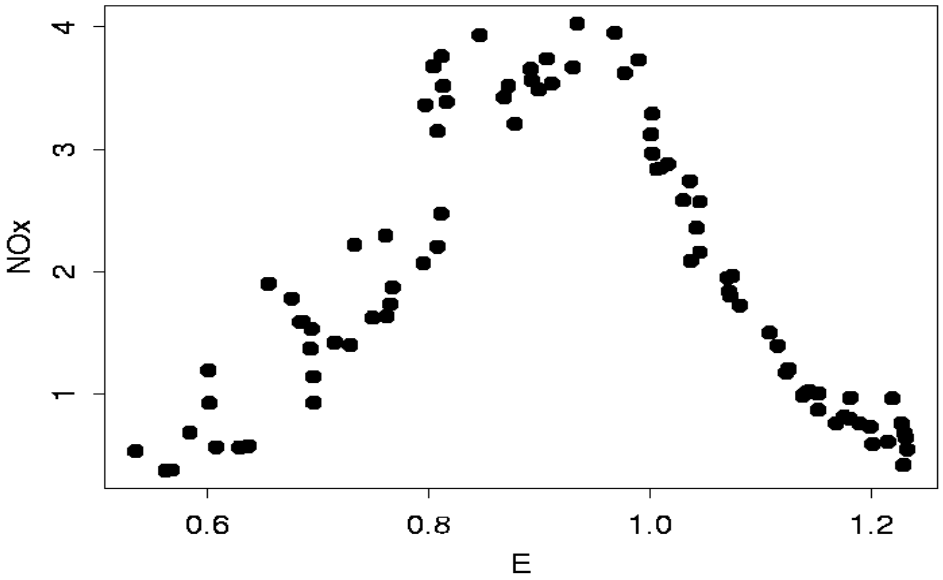


MLPs e não-linearidade (3/3)



Função contínua qualquer (1/2)

- **O que faríamos se os dados fossem como estes?**

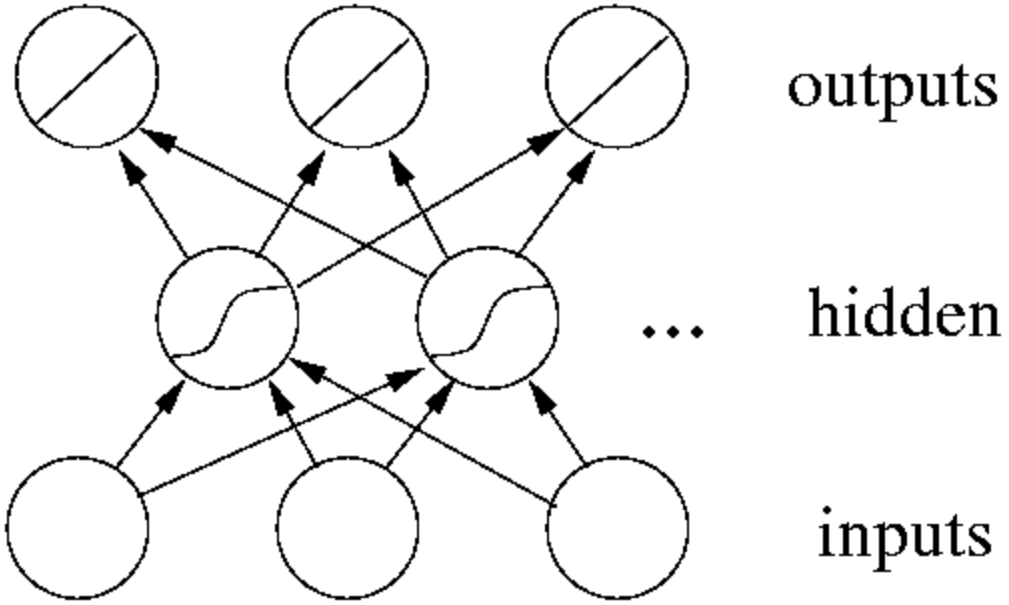


Função contínua qualquer (2/2)

- A função tangente hiperbólica não se ajusta a estes dados
- Definir uma função de ativação específica para cada conjunto de dados
  - E o aprendizado? Como fica?
- Encontrar um método geral de aproximação de função não linear que consiga se ajustar a qualquer dado
  - independente de sua aparência

Função contínua - camadas escondidas (1/2)

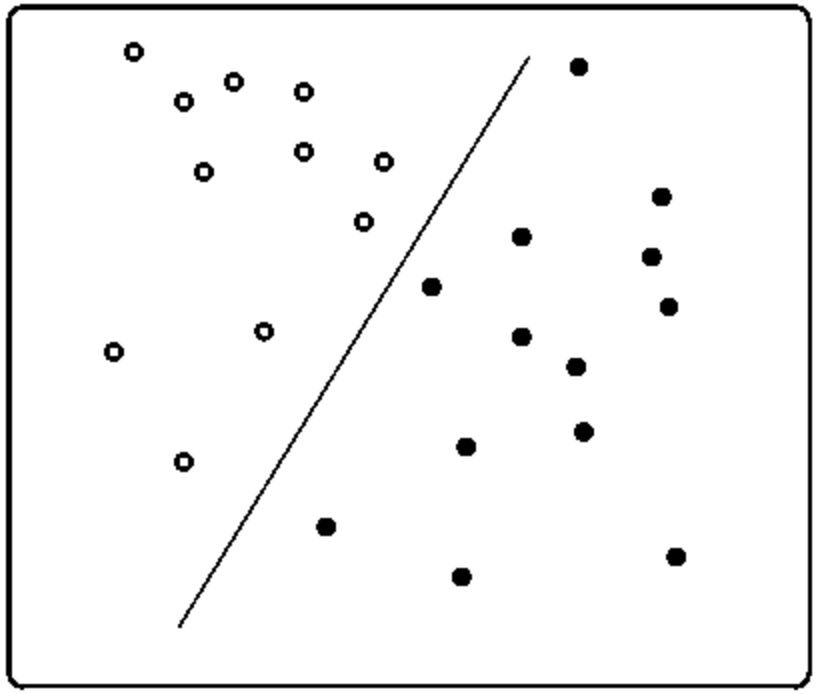
- **Solução:** **adicione mais nodos na camada escondida**

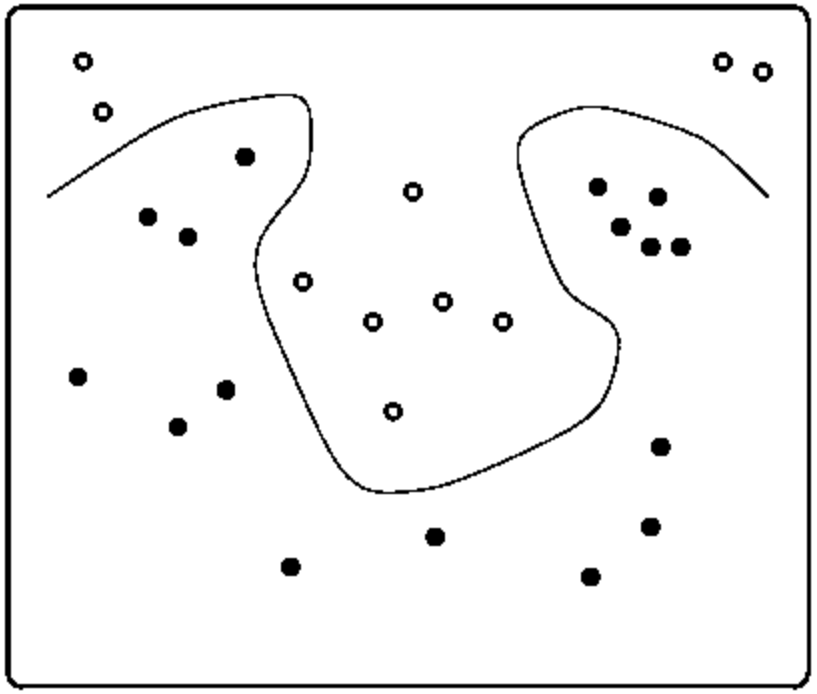


Função contínua - camadas escondidas (2/2)

- **Teorema: dado um número suficiente de nodos escondidos, uma MLP com apenas uma camada escondida pode aproximar qualquer função contínua (Cybenko, 1989)**
- **Tradução: qualquer função pode ser expressa como uma combinação linear de funcões *tanh***
  - **função de base universal**
  - **Outros exemplos: funções sigmóides, gaussiana, ...**

MLPs - classificação não linear





Backpropagation (1/2)

- **Já vimos como treinar redes lineares (Adalines) com o gradiente descendente**
- **Se tentarmos usar o mesmo algoritmo para as MLPs encontraremos uma dificuldade**
  - **Não temos as saídas desejadas para as unidades da camada escondida**
  - **Como poderíamos ajustar os pesos destas unidades? (*credit assignment problem*)**
    - **Os pesos da camada de saída podem ser ajustados com a regra das Adalines**

Backpropagation (2/2)

- **O problema não resolvido de *credit assignment* foi uma das razões para o declínio das pesquisas em RNAs**
  - **Passou-se quase 30 anos - de 1958 a 1986 - até que o algoritmo de backpropagation popularizasse uma maneira de treinar as unidades da camada escondidade**
  - **Werbos (1974), (Rumelhart, Hinton, and Williams, 1986), (LeCun,1985)**
- **Em princípio, o backpropagation propicia uma maneira de treinar redes com qualquer nr. de nodos escondidos organizados em um nr. qualquer de camadas**

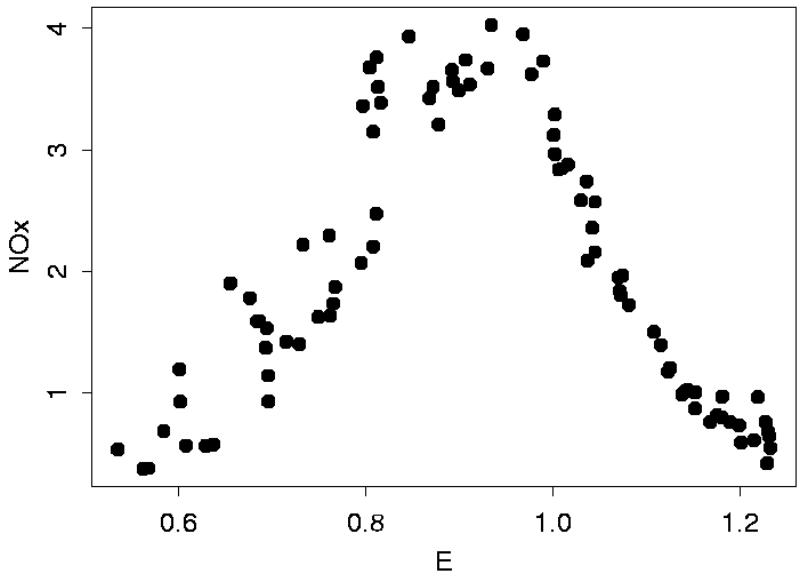
Backpropagation - o algoritmo (1/2)

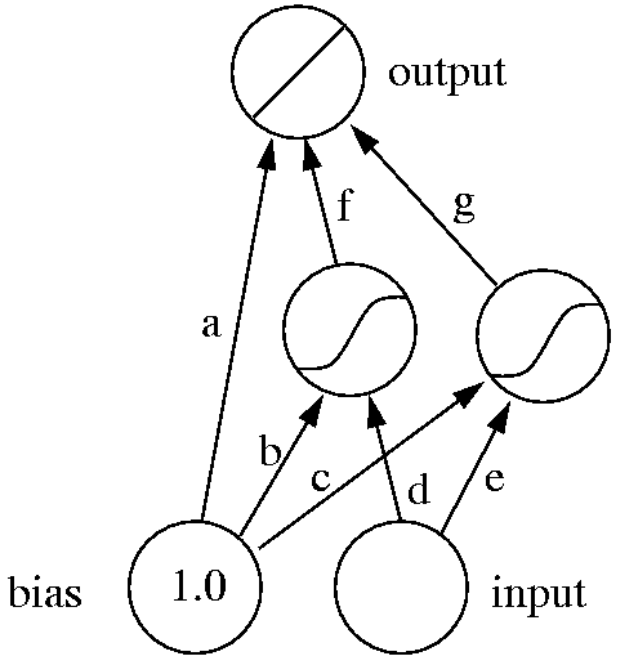
- Create a feed-forward network with n_{in} inputs; n_{hidden} hidden units, n_{out} output units
- Initialize all network weights to small random numbers
- Until the termination condition is met, Do
  - For each (x,t) in training-set, Do
    - Propagate the input forward through the network:
    - 1. Input instance x to the network and compute the output o_u of every unit u
    - Propagate the errors backwards through the network:
    - 2. For each network output k, calculate its error term δ_k

Backpropagation - o algoritmo (2/2)

- Propagate the errors backwards through the network:
- 2. For each network output k, claculate its error term δ_k
- $\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$
- 3. For each hidden unit h, calculate its error term δ_h
- $\delta_h \leftarrow o_h(1 - o_h)\sum_{k \in \text{outputs}} w_{kh}\,\delta_k$
- 4.Update each network weight w_{ji}
- $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$
- $\Delta w_{ji} \leftarrow \eta\,\delta_j\,x_{ji}$

Backpropagation - exemplo (1/9)

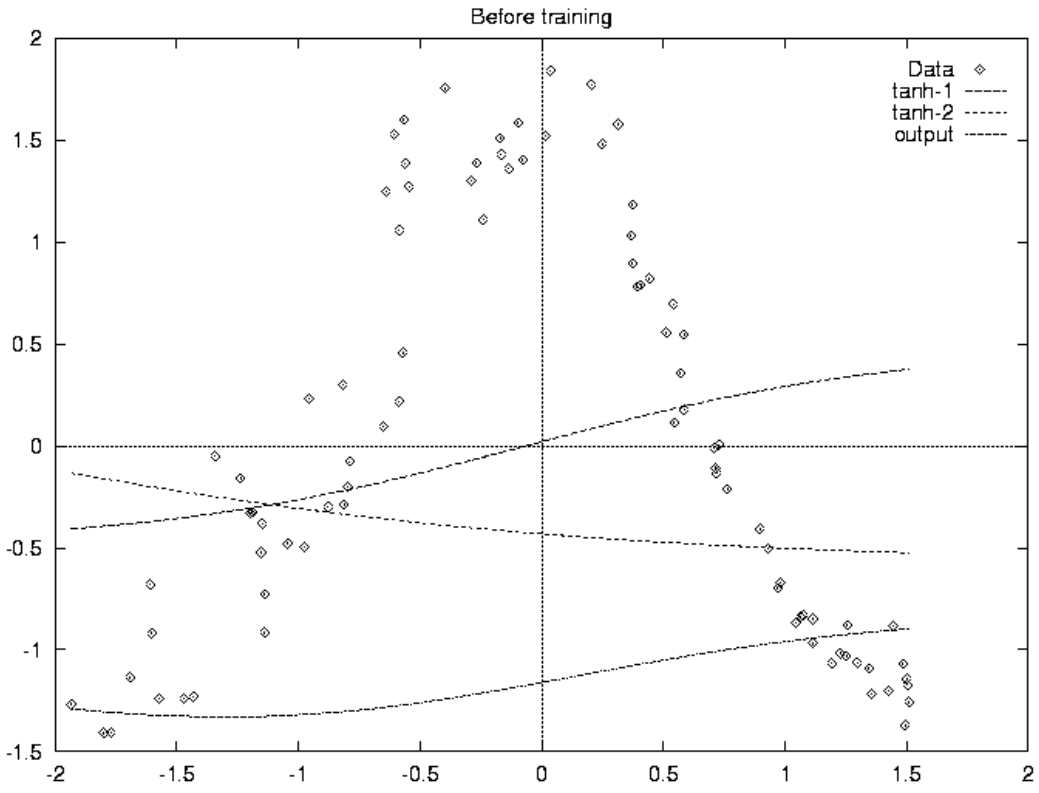




Backpropagation - exemplo (2/9)

- **O nó de saída computa uma combinação de duas funções**
  - $y_o = f * tanh1(x) + g*tanh2(x) + a$, **onde**
  - $tanh1(x) = tanh(dx + b)$
  - $tanh2(x) = tanh(ex + c)$
- **Inicializamos os pesos com valores aleatórios**
  - ***Cada nó escondido computa uma função tanh aleatória***

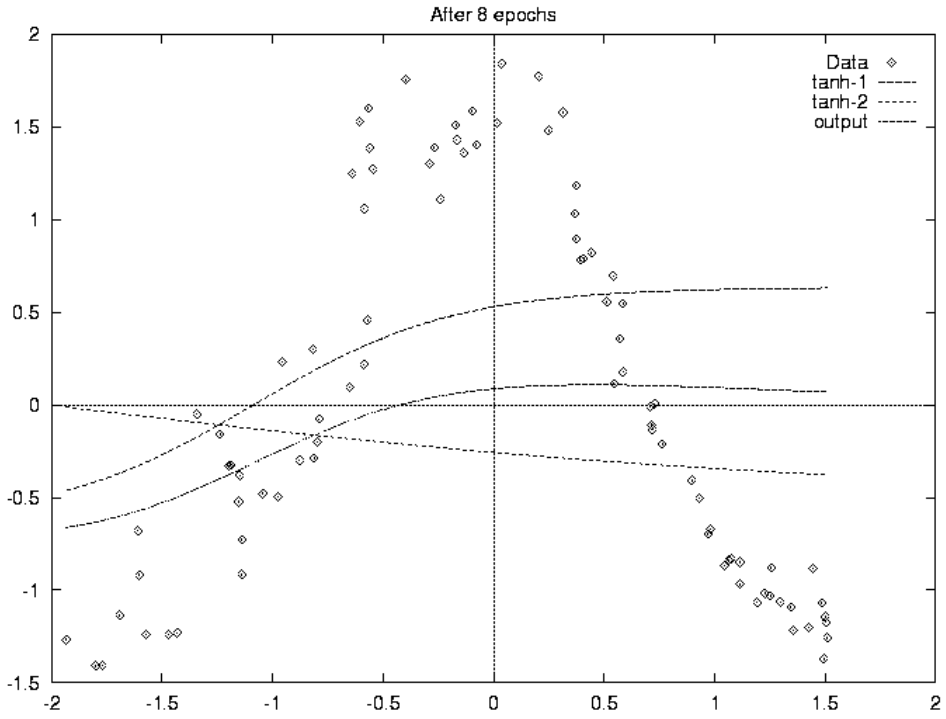
Backpropagation - exemplo (3/9)



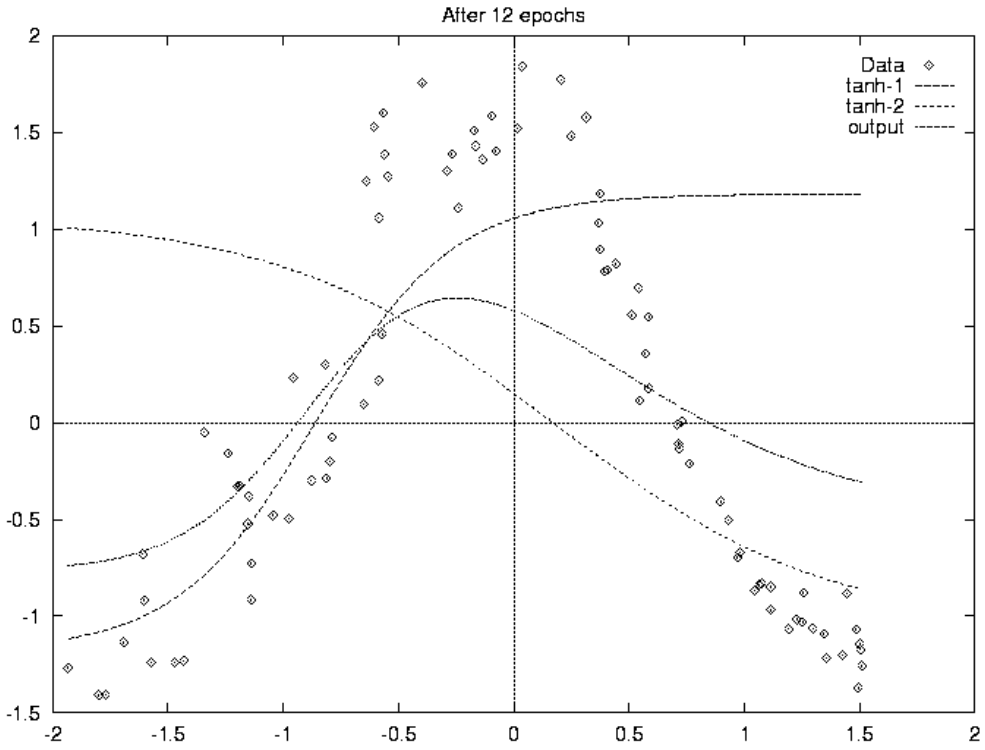
Backpropagation - exemplo (4/9)

- **Treinamento da rede**
  - **Taxa de aprendizado 0.3**
  - **Atualização dos pesos após cada padrão**
    - **Aprendizagem incremental**
  - **Depois de passarmos por todo o conjunto de treinamento 10 vezes (*10 training epochs*)**
    - **As funções computadas pela rede têm a seguinte forma**

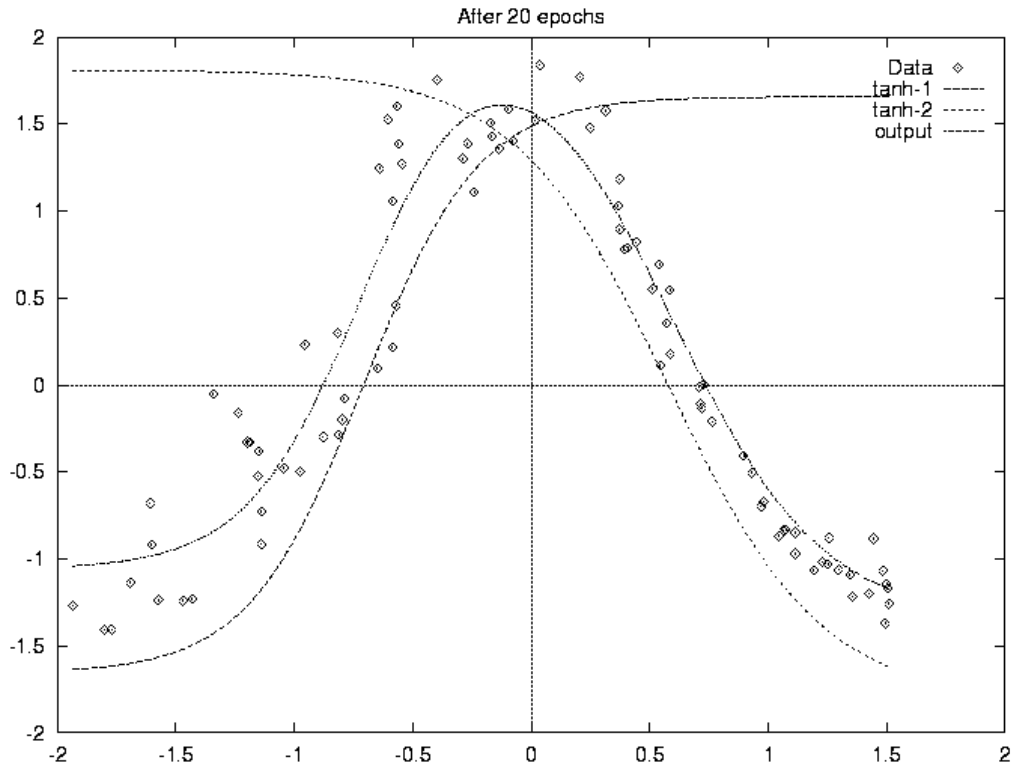
Backpropagation - exemplo (5/9)



Backpropagation - exemplo (6/9)



Backpropagation - exemplo (7/9)

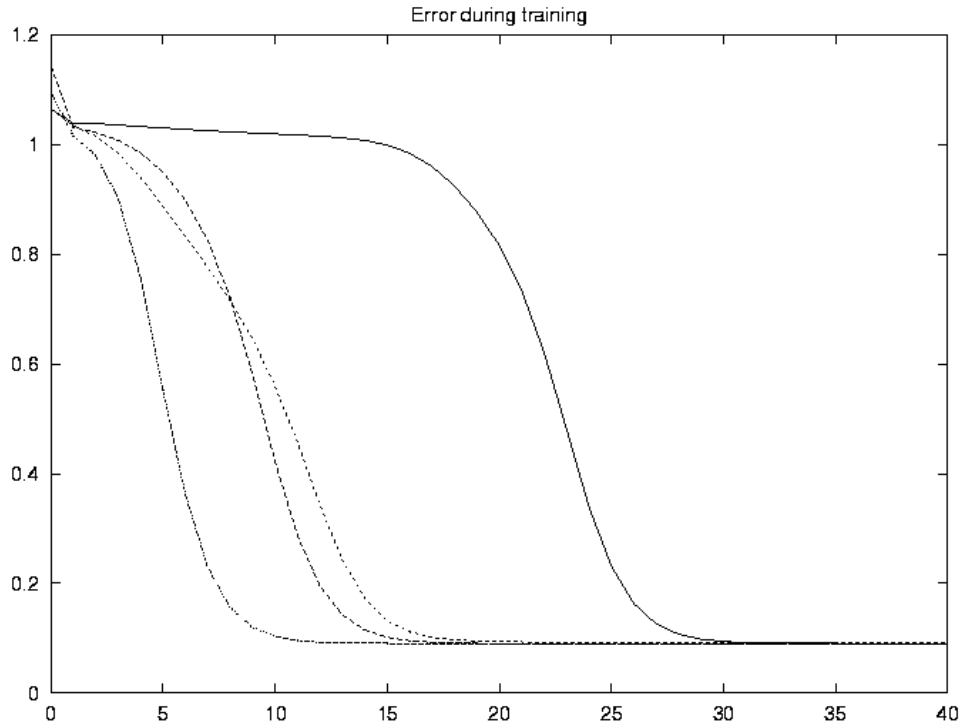


Backpropagation - exemplo (8/9)

- **As funções são:**
  - ***streched, scaled and shifted***
- **através das várias atualizações do pesos**

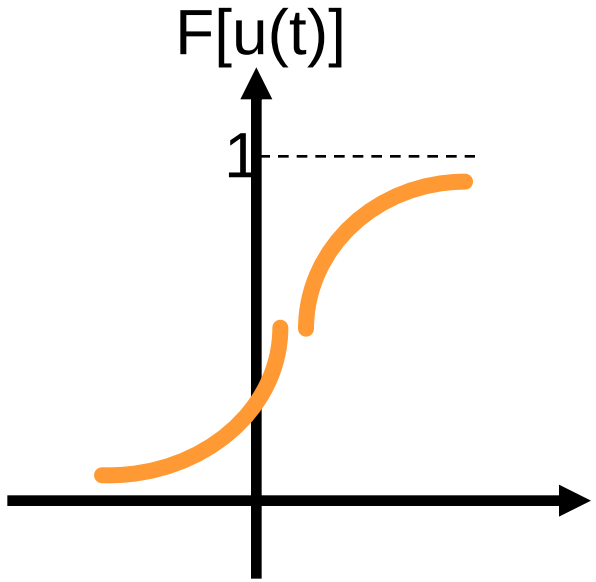
- **Na próxima figura, plotamos o gráfico do erro sobre todos os 88 padrões como uma função do ciclo de treinamento**
  - **4 treinamentos são mostrados, com inicializações de pesos diferentes**

Backpropagation - exemplo (9/9)



Função Sigmoide (1/3)

- **Uma função sigmoide é capaz de "imitar" uma série de comportamentos dependendo de sua inclinação**



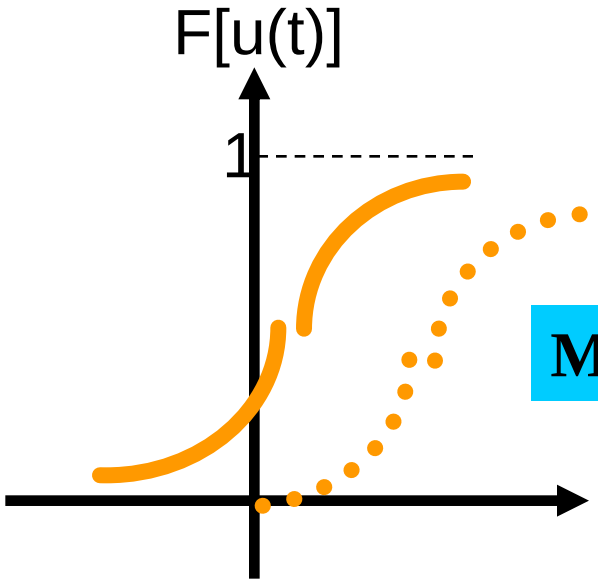
$$F[u(t)] = 1/(1 + e^{-u(t)})$$

$$u = \sum_{j=1}^{N} x_j w_j$$

A inclinação é ajustada de acordo com os pesos

Função Sigmoide (2/3)

- **Uma função sigmoide também pode ser transladada pelo termo constante na Equação 1 ($x_0 = -1$)**



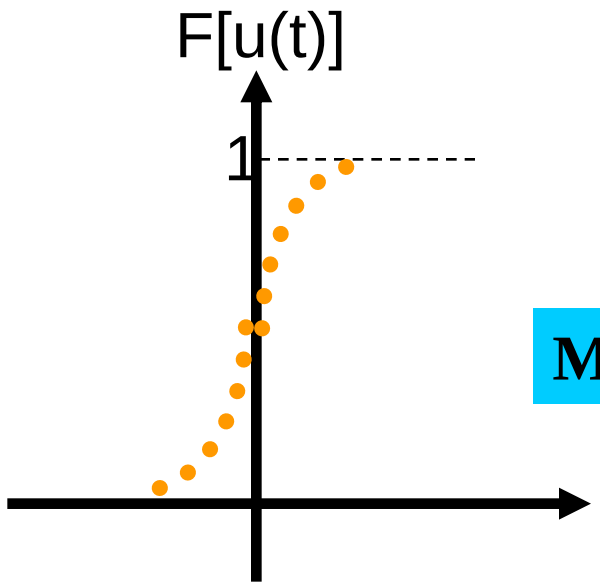
$$F[u(t)] = 1/(1 + e^{-u(t)})$$

$$u = \sum_{j=0}^{N} x_j w_j \qquad \textbf{Eq. 1}$$

Mudança em w_0

Função Sigmoide (3/3)

- **Uma função sigmoide também pode ser *streched***



$$F[u(t)] = 1/(1 + e^{-u(t)})$$

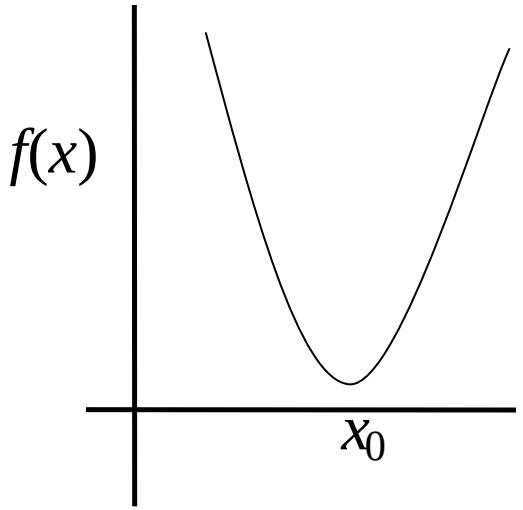
$$u = \sum_{j=0}^{N} x_j w_j \qquad \textbf{Eq. 1}$$

Mudança de todos w_j $j \neq 0$

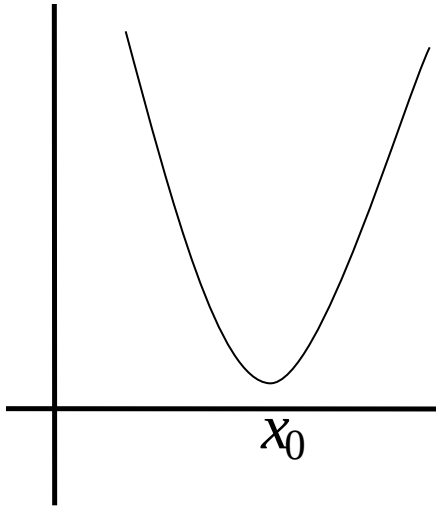
O problema da otimização

- **Achar o ponto de máximo ou mínimo de uma função *f*(*x*,*y*,*z*,*u*,...) de *n* parâmetros, *x*,*y*,*z*,*u*,...**

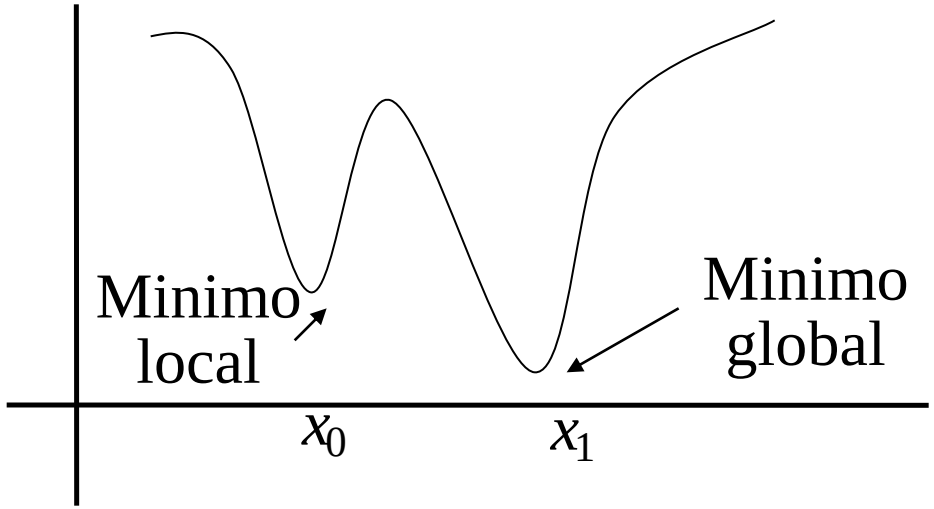
minimizar *f*(*x*) = maximizar -*f*(*x*)



Função unimodal x multimodal

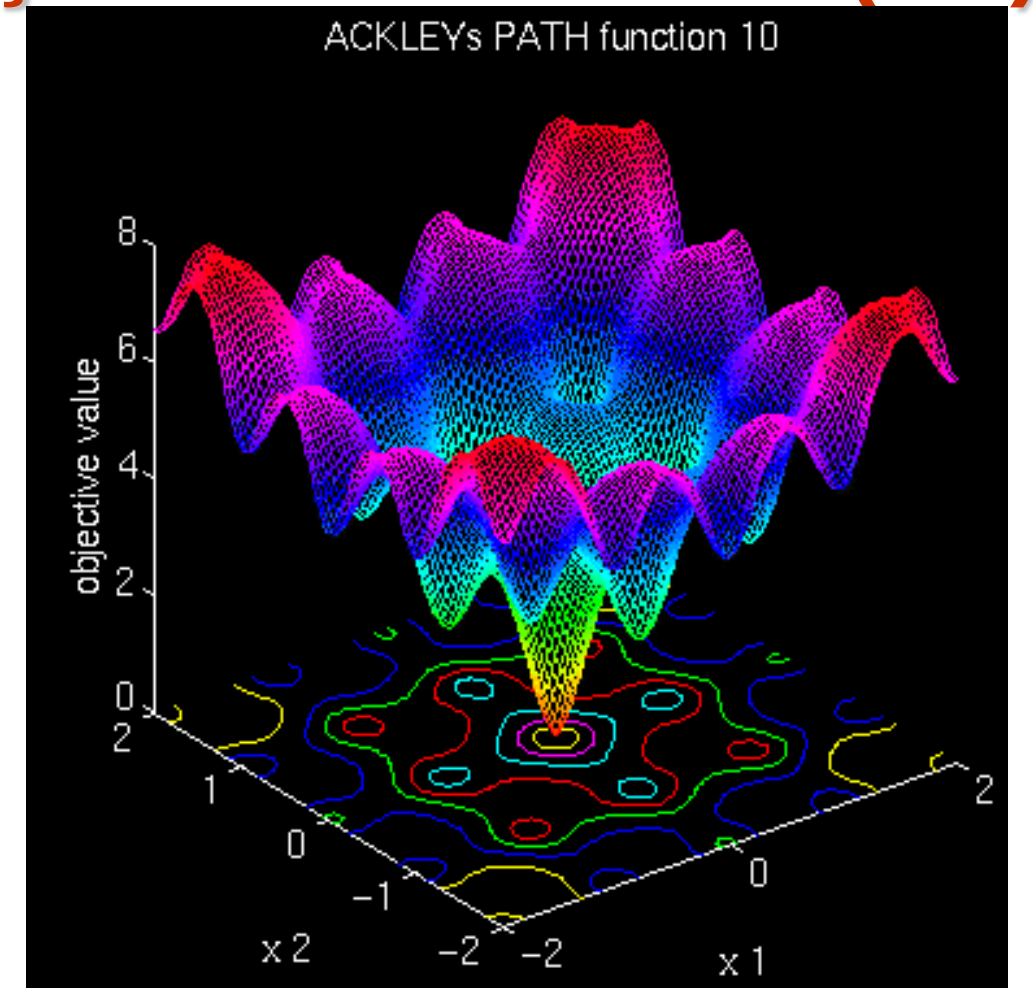


Função unimodal

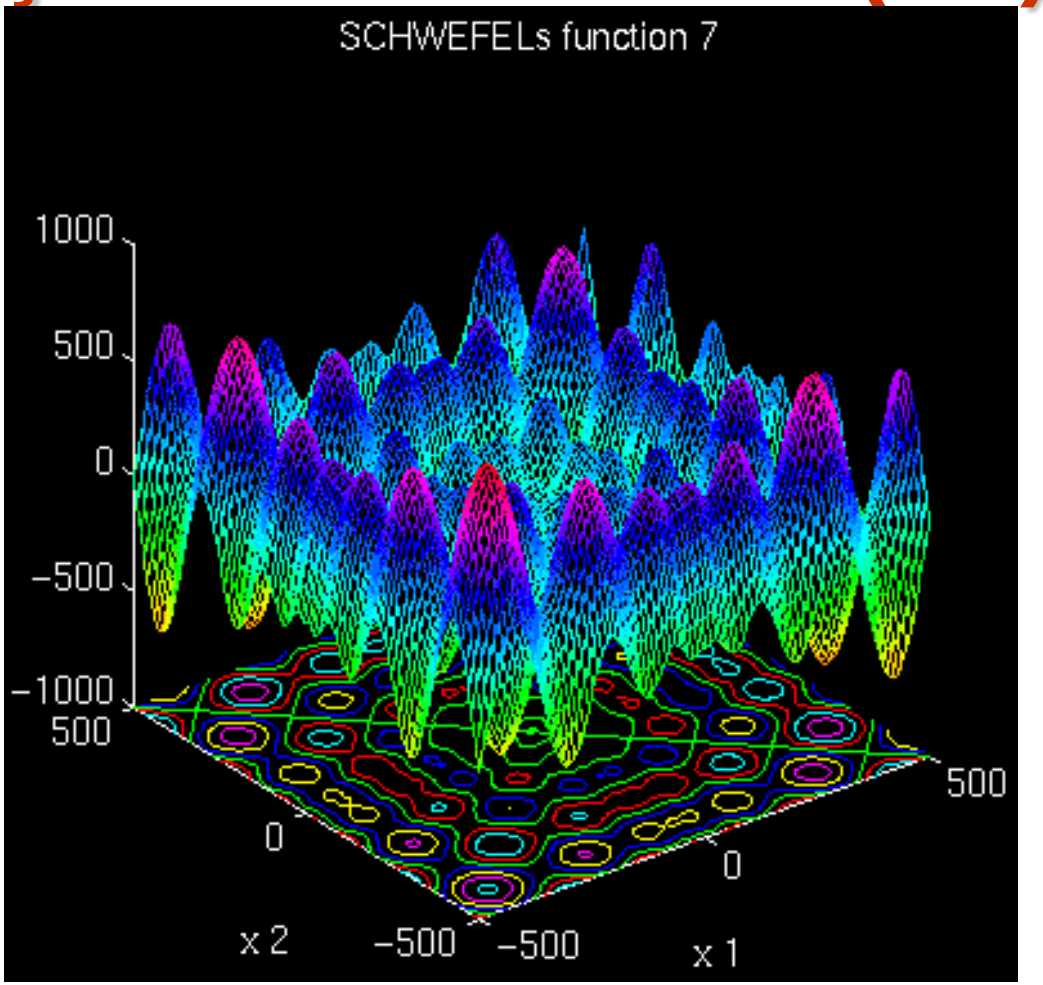


Função multimodal

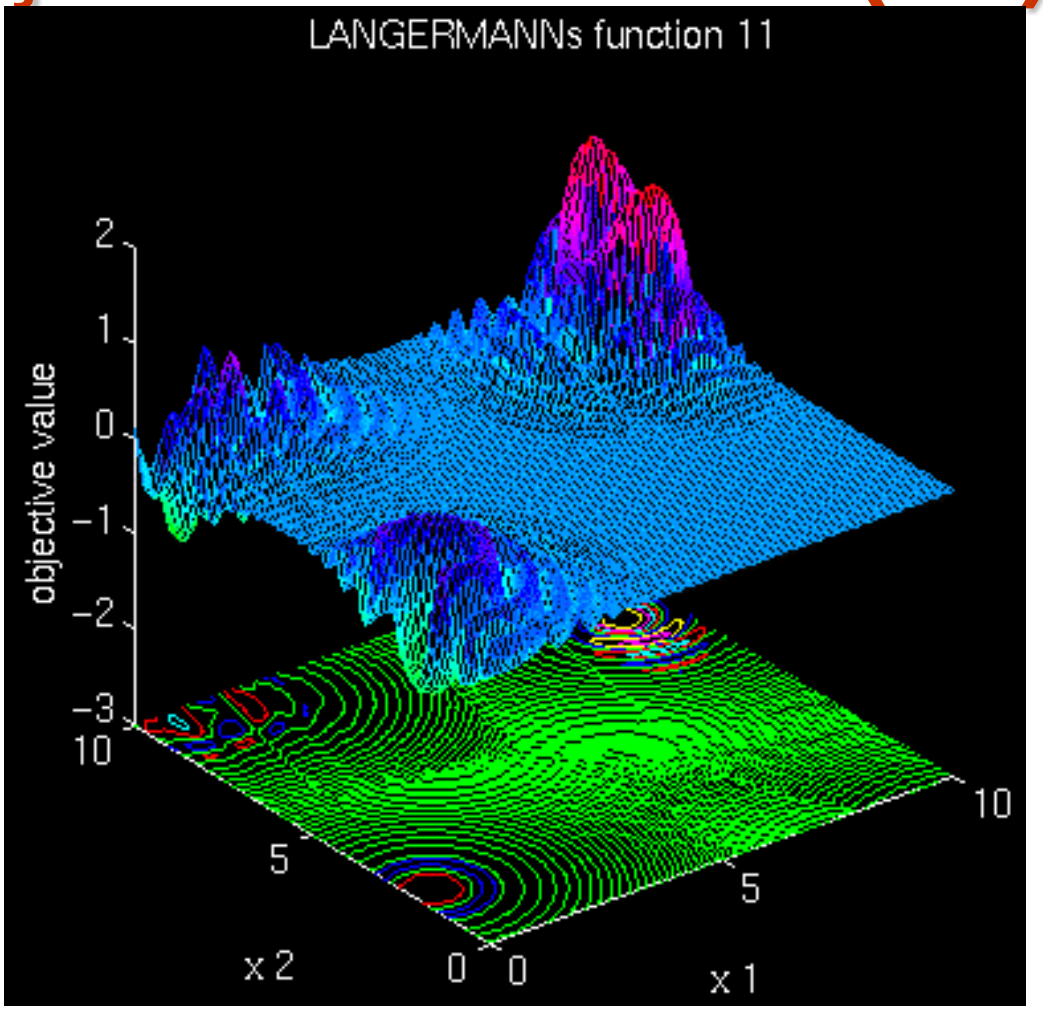
Exemplos de Funções Multimodais (1/3)



Exemplos de Funções Multimodais (2/3)

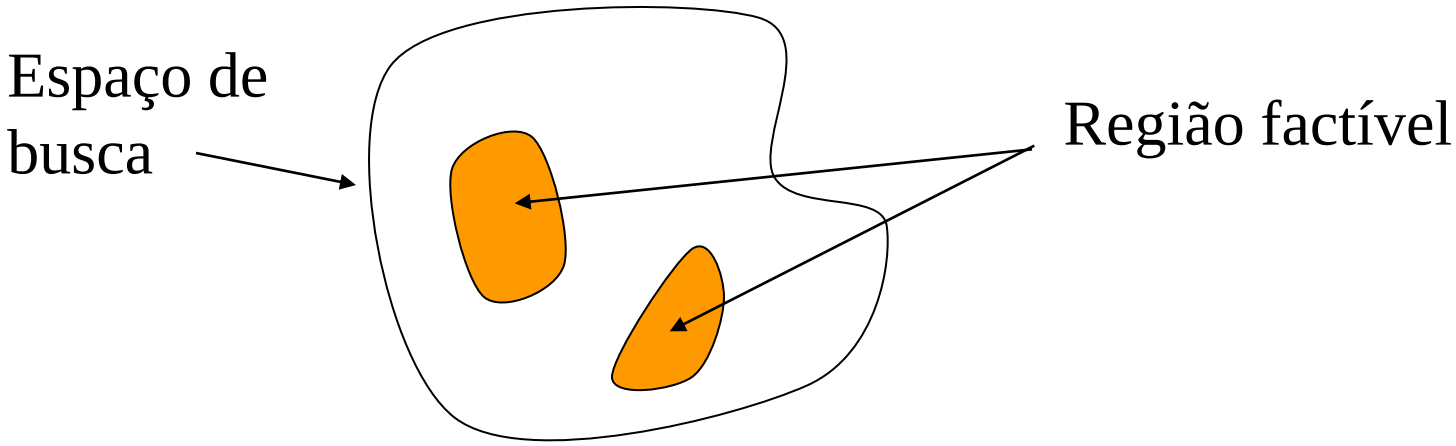


Exemplos de Funções Multimodais (3/3)



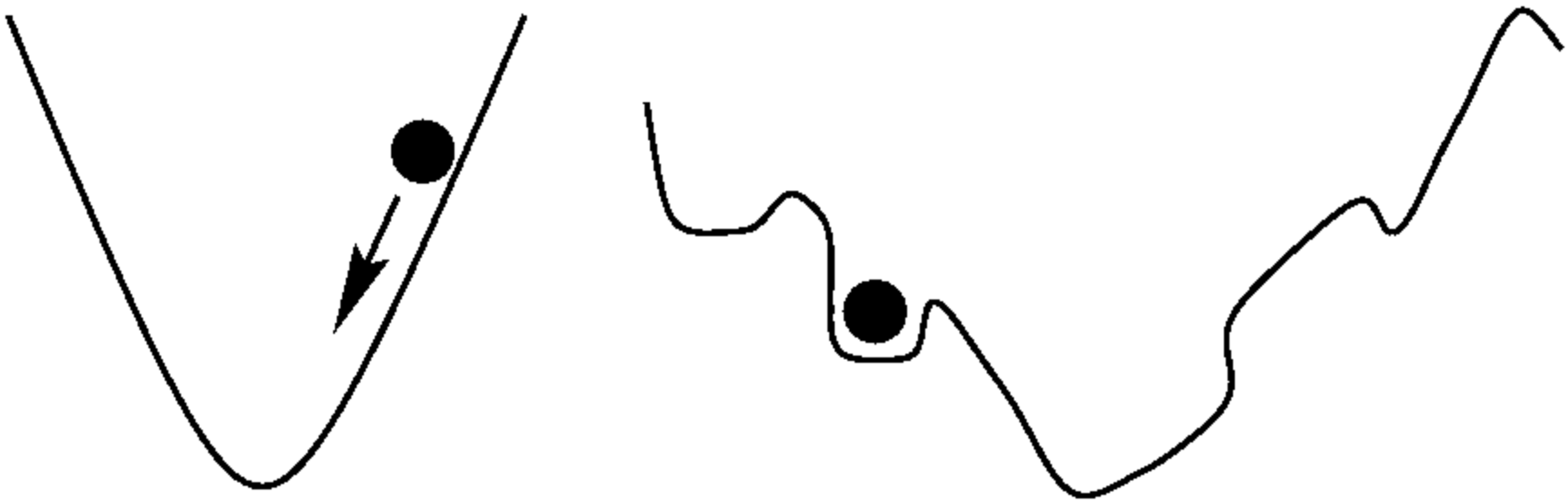
Região factível x não factível

- **O espaço de busca se divide em região factível e não factível**
- **As soluções da região factível são aquelas que satisfazem a todas as restrições**



Adição do coeficiente de momento (1/3)

- **Mínimo local**

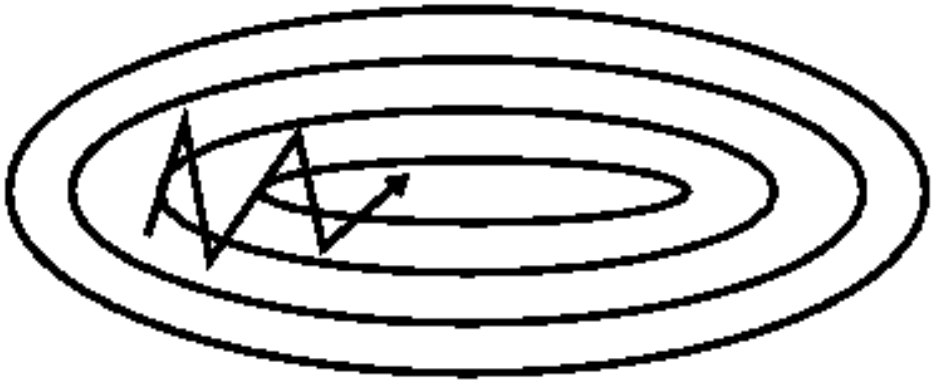


Adição do coeficiente de momento (2/3)

- **Adiciona uma fração α do valor anterior de atualização dos pesos ao atual**
  - $\Delta w_{ji}(t+1) \leftarrow \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(t)$
- **Qdo. o gradiente se mantém apontando na mesma direção, o tamanho dos passos na direção do mínimo crescerá**
  - **Atenção: se ambos η e α forem muito grandes, há o risco de passar pelo mínimo**

Adição do coeficiente de momento (3/3)

- **Qdo. o gradiente se mantém mudando de direção, o coef. de momento suavizará está variação**
  - **Vales longos e estreitos**
  - **Plateaus**



Backpropagation - Overfitting

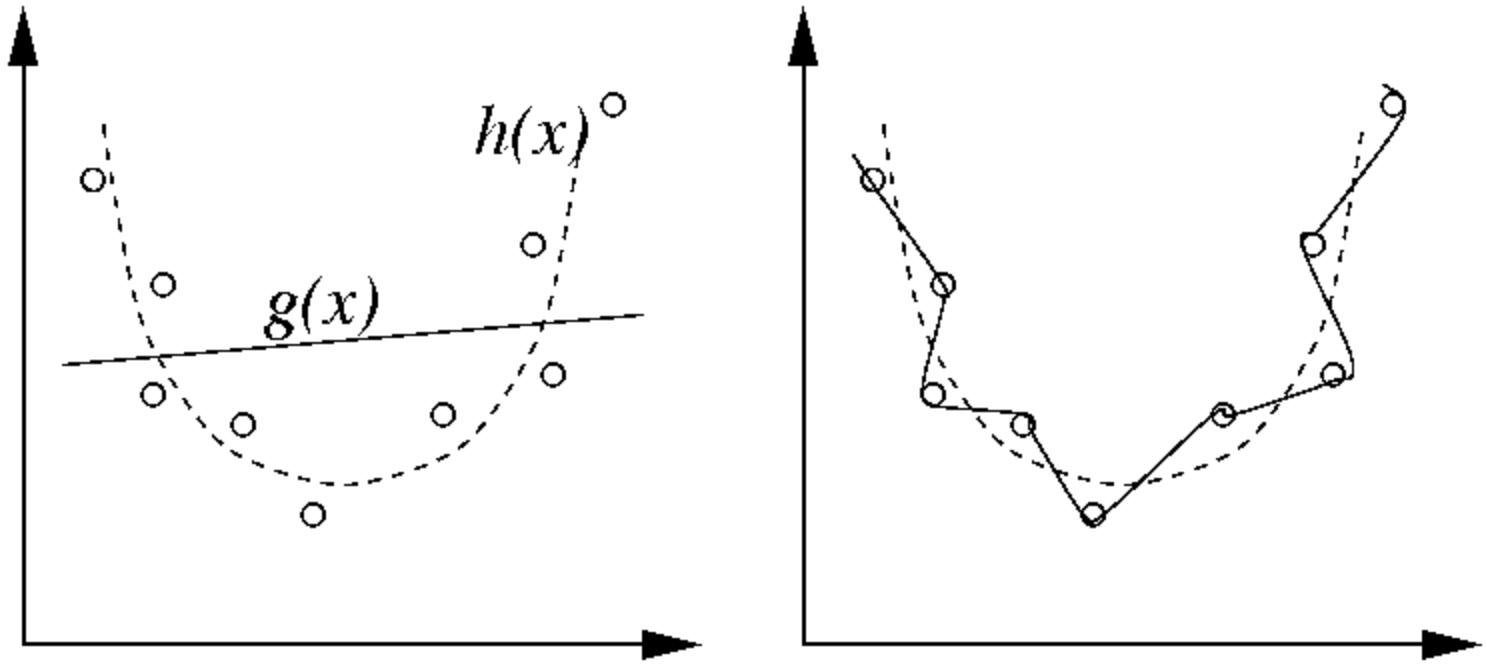
- No exemplo anterior, usamos uma rede com 2 unidades escondidas
  - Apenas olhando os dados já dava pra ter uma idéia que 2 funções tanh resolveriam o problema
- Em geral não temos como saber qtos. nodos escondidos e pesos serão necessários
- Procuramos um modelo que produza o melhor resultado para novos dados (generalização)
  - Pode haver conflito com a tarefa de modelar o cj. de treinamento (Memorização)

Bias-Variance trade-off (1/5)

- **O conjunto de treinamento foi gerado a partir de uma função suave, *h(x)*, com ruído adicionado**

- **Obviamente, queremos encontrar um modelo que aproxime *h(x), dado um conjunto específico de dados y(x) gerado como:***
  - $y(x) = h(x) + \varepsilon$

Bias-Variance trade-off (2/5)



Bias-Variance trade-off (3/5)

- **No gráfico da esquerda, tentamos ajustar os pontos usando uma função *g(x)* que tem poucos parâmetros: uma reta**
  - **O Modelo é muito rígido e não consegue modelar nem o cj. de treinamento nem novos dados**
  - **O modelo têm um *bias* alto**
  - **Underfitting**

Bias-Variance trade-off (4/5)

- **O gráfico da direita mostra um modelo que foi ajustado usando muitos parâmetros livres**
  - **Ele se ajusta perfeitamente ao dados do cj. de treinamento**
  - **Porém, este modelo não seria um bom "previsor" de *h(x)* para novos valores de x**
  - **Dizemos que o modelo tem uma variância (*variance*) alta**
  - **Overfitting (super-ajustamento)**

Bias-Variance trade-off (5/5)

- **Claramente, o que queremos é um compromisso entre:**
  - **um modelo que seja poderoso o bastante para representar a estrura básica do dado - *h(x)* -,**
  - **mas não tão poderoso a ponto de modelar fielmente o ruído associado a um cj. de dados particular**

Prevensão do overfitting (1/3)

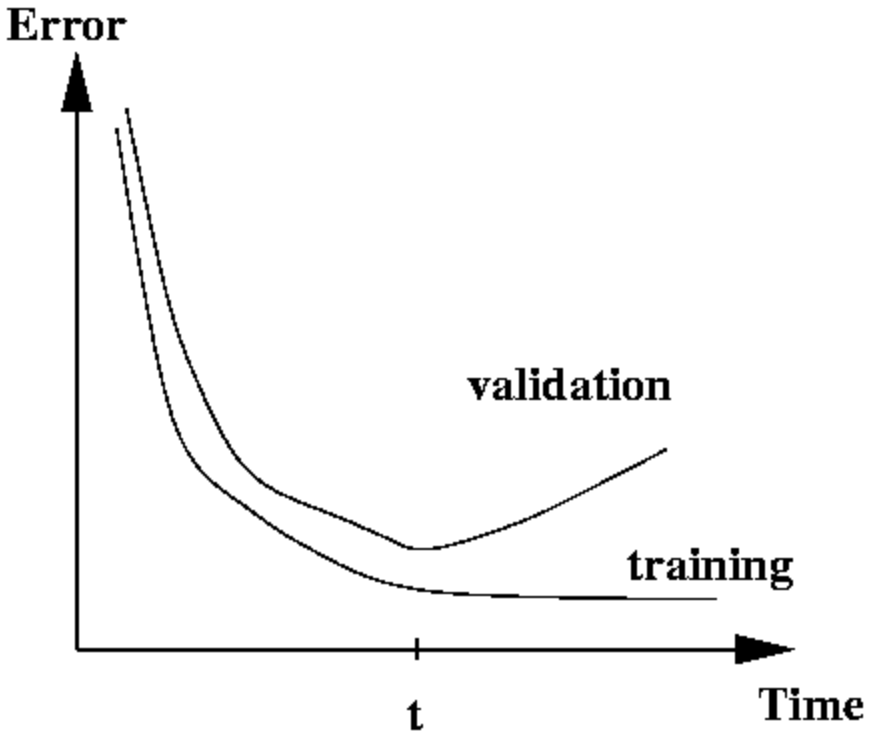
- **O bias-variance trade-off é um problema principalmente qdo. o cj. de treinamento é pequeno**
- **Em geral, qdo. temos um número “infinito” de dados para o treinamento - on-line learning contínuo - não temos o problema de overfitting**
- **As técnicas seguintes são usadas para prevenir overfitting em off-line learning**

Prevensão do overfitting (2/3)

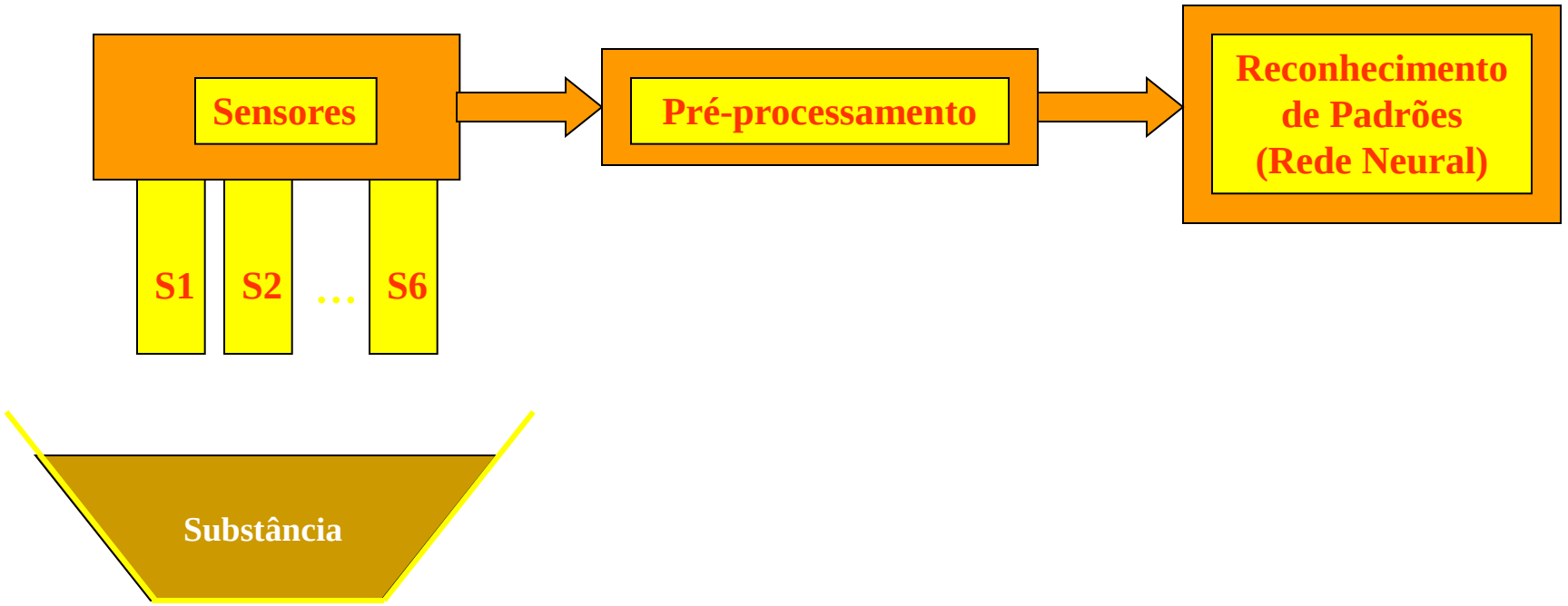
- **Early stopping**
  - **Grande número de dados disponível**
  - **Conjunto de treinamento**
  - **Conjunto de validação**
    - **usado para testar a generalização da rede durante o treinamento**
  - **Conjunto de teste**

Prevensão do overfitting (3/3)

- **Early stopping**



Nariz Artificial

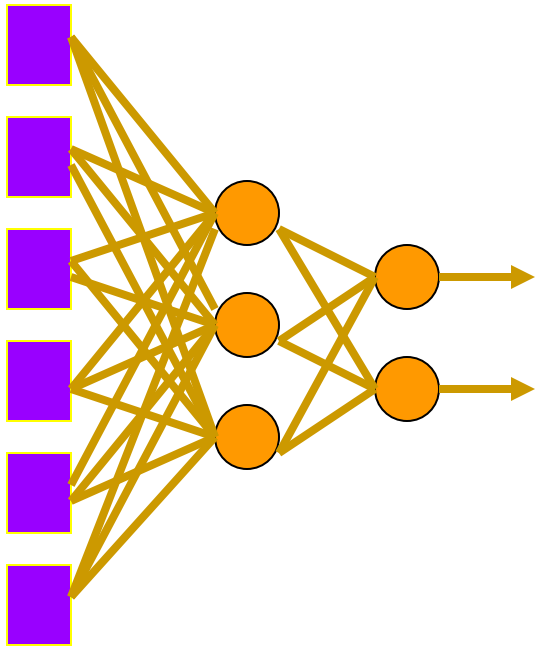


Problema Abordado

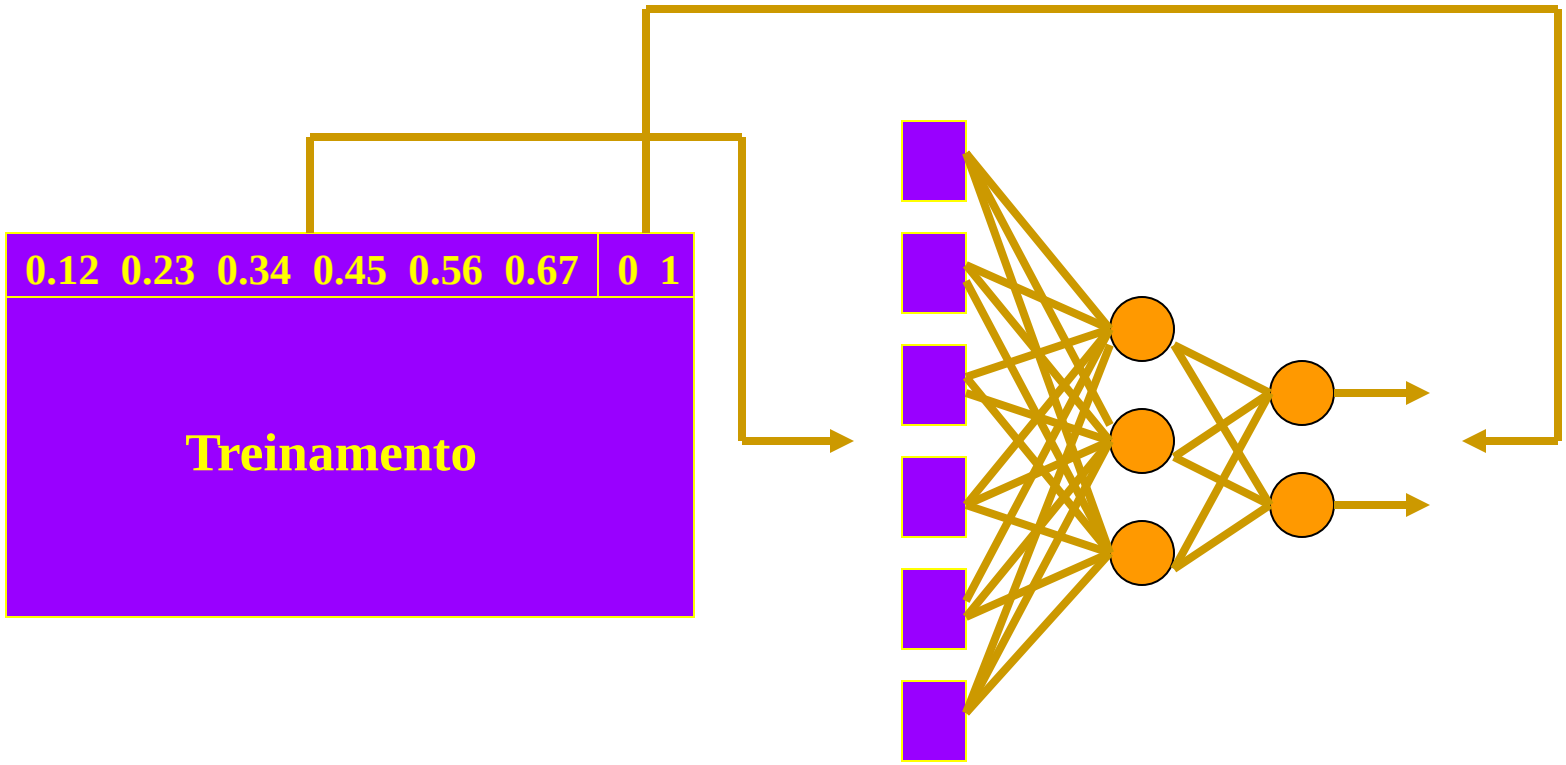
- **Base de Dados:**
  - **Classificação entre odores de duas safras de vinho (A e B)**
  - **Para cada safra, as resistências dos sensores foram registradas a cada 0.5s.**
  - **Cada conjunto de seis valores registrados no mesmo instante de tempo é um *padrão* (total de 200 padrões, sendo 100 da safra A e 100 da safra B).**
- **Divisão do Conjunto de Padrões (Proben1):**
  - **50% dos padrões de cada safra escolhidos aleatoriamente para *treinamento*,**
  - **25% para *validação*,**
  - **e 25% para *teste*.**

Rede MLP

- **Rede MLP:**
  - **Uma camada intermediária,**
  - **6 unidades de entrada (uma para cada sensor),**
  - **2 unidades de saída (uma para cada safra de vinho),**
  - **Função de ativação sigmóide logística,**
  - **Todas as possíveis conexões entre camadas adjacentes, sem conexões entre camadas não-adjacentes,**



Treinamento



Soma dos Erros Quadráticos (SSE)

- **Saídas da rede:**

  0.98 0.12 ... 0.16

  0.02 0.96 ... 0.88

- **Saídas desejadas:**

  1.00 0.00 ... 0.00

  0.00 1.00 ... 1.00

- **Soma dos erros quadráticos (SSE):**

  $$SSE = (0.98 - 1)^2 + (0.12 - 0)^2 + \ldots + (0.16 - 0)^2 +$$
  $$(0.02 - 0)^2 + (0.96 - 1)^2 + \ldots + (0.88 - 1)^2$$

Bibliografia

- Braga, A.;Carvalho, A. C. P. L. F. & Ludermir, T. : Redes Neurais Artificiais: Teoria e Aplicações. Publicado pela Editora Livro Técnico e Científico, Rio de Janeiro, Brasil, 2000 (livro texto)
- Rezende, S. O. (Coordenadora): Sistemas Inteligentes: Fundamentos e Aplicações. Publicado pela Editora Manole LTDA, São Paulo, Brasil, 2002 (livro texto)
- Mithcell, T.: Machine Learning, McGraw-Hill, 1997.
- Haykin, S.: Neural Networks: A Comprehensive Foundation, Prentice Hall, 1999