REDES NEURAIS: PERCEPTRON E MLP

Inteligência Artificial

André Câmara

UFRPE Universidade Federal Rural de Pernambuco

DEINFO Departamento de Estatística e Informática

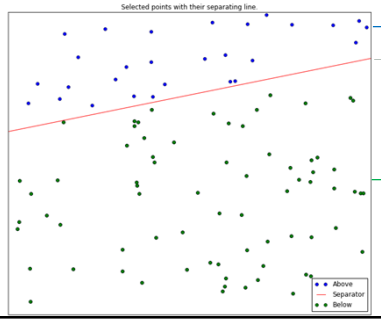
Slides baseados no material de aula da prof. Teresa Ludermir (Cin/UFPE)

Relembrando…

- Classificador Linear
  - $f(\boldsymbol{x_i}) = \boldsymbol{\phi}(\boldsymbol{x_i}) \cdot \boldsymbol{w}$
- Erro:
  - $L_2(y_i, f(\boldsymbol{x_i})) = (y_i - f(\boldsymbol{x_i}))^2$
  - Função de erro quadrática, com ponto mínimo em $\frac{\partial}{\partial w_i}[(y_i - f(\boldsymbol{x_i}))^2]$

Problemas linearmente separáveis

- Fronteira de decisão



$ax_1+bx_2+c<0$

$ax_1+bx_2+c=0$

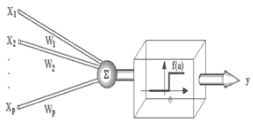
$ax_1+bx_2+c>0$

Perceptrons

- Desenvolvido por Rosemblat, 1958
- Rede mais simples para classificação de padrões linearmente separáveis
- Utiliza modelo de McCulloch-Pitts como neurônio

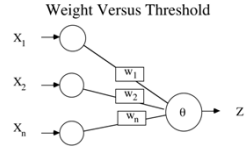
Perceptrons

- Estado de ativação
  - 1 = ativo
  - -1 = inativo
- Função de ativação
  - $a_i(t+1) = f(u_i(t))$
  - $f(u_i(t)) = \begin{cases} -1, & \text{se } u_i(t) < 0 \\ +1, & \text{se } u_i(t) \geq 0 \end{cases}$

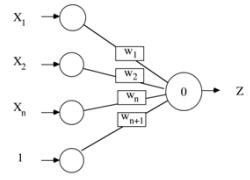


$$u = \sum_{j=1}^{N} x_j w_j - \theta$$

Weight Versus Threshold



Do you need to adjust Theta? Yes, in most cases



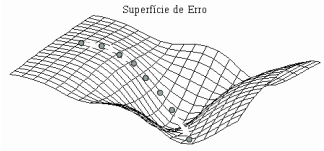
where $w_{n+1} = -\theta$

Perceptrons

- Treinamento
  - Supervisionado
  - Correção de erro
    - $\Delta w_{ij} = \eta\, x_i (d_j - y_j)$ $(d_j \neq y_j)$
    - $\Delta w_{ij} = 0$ $(d_j = y_j)$
- η = *taxa de aprendizado*
- Teorema de convergência: se é possível separar linearmente um conjunto de entradas, o Perceptron fará a classificação

Algoritmo de treinamento

- 1) Iniciar todas as conexões com $w_{ij} = 0$;
- 2) Repita até o erro ser aceitável
  - Para cada par de treinamento (***x***, *d*)
    - Calcular a saída *y*
    - Se ($d \neq y$) então
      - Atualizar pesos dos neurônios



Classificação

1) Apresentar padrão X a ser reconhecido
2) Calcular a saída y
3) Se (y = -1)
   Então
      X ∈ classe 0
   Se não
      X ∈ classe 1

Exemplo

- Dada uma rede do tipo Perceptron formada por um neurônio com três terminais de entrada, utilizando pesos iniciais
  $w_0 = 0.4$, $w_1 = -0.6$ e $w_2 = 0.6$,
  limiar $\theta = 0.5$ e uma taxa de aprendizado = 0.4,
  responda os itens abaixo:
  a) Ensinar a rede a gerar a saída -1 para o padrão 001 e a saída +1 para o padrão 110
  b) A que classe pertencem os padrões 111, 000, 100 e 011?

Exemplo 1: resposta a

a) Treinar a rede
   a.1) Para o padrão 001 (d = -1)
      Passo 1: definir a saída da rede
         u = 0(0.4) + 0(-0.6) + 1(0.6) -1(0.5) = 0.1
         y = u = +1 (uma vez 0.1 ≥ 0)

Exemplo 1: resposta a

a) Treinar a rede
   a.1) Para o padrão 001 (d = -1)
      Passo 1: definir a saída da rede
         u = 0(0.4) + 0(-0.6) + 1(0.6) -1(0.5) = 0.1
         y = u = +1 (uma vez 0.1 ≥ 0)
      Passo 2: atualizar os pesos
         $w_0 = 0.4 + 0.4(0)(-1 - (+1)) = 0.4$
         $w_1 = -0.6 + 0.4(0)(-1 - (+1)) = -0.6$
         $w_2 = 0.6 + 0.4(1)(-1 - (+1)) = -0.2$
         $w_3 = 0.5 + 0.4(-1)(-1 - (+1)) = 1.3$

Peso atual

Taxa de Aprendizado

$\Delta w_{ij} = \eta x_i (d_j - y_j)$

Exemplo 1: resposta a

a) Treinar a rede

a.2) Para o padrão 110 (d = 1)

Passo 1: definir a saída da rede

u = 1(0.4) + 1(-0.6) + 0(-0.2) -1(1.3) = -1.5

y = u = -1 (uma vez -1.5 < 0)

Passo 2: atualizar pesos

w_0 = 0.4 + 0.4(1)(1 - (-1)) = 1.2

w_1 = -0.6 + 0.4(1)(1 - (-1)) = 0.2

w_2 = -0.2 + 0.4(0)(1 - (-1)) = -0.2

w_2 = 1.3 + 0.4(-1)(1 - (-1)) = 0.5

Exemplo 1: resposta a

a) Treinar a rede

a.3) Para o padrão 001 (d = -1)

Passo 1: definir a saída da rede

u = 0(1.2) + 0(0.2) + 1(-0.2) -1(0.5) = -0.7

y = u = -1 (uma vez -0.7 < 0)

Passo 2: atualizar pesos

Como d = y, os pesos não precisam ser modificados

Exemplo 1: resposta a

a) Treinar a rede

a.4) Para o padrão 110 (d = 1)

Passo 1: definir a saída da rede

u = 1(1.2) + 1(0.2) + 0(-0.2) -1(0.5) = 0.9

y = u = 1 (uma vez 0.9 $\geq$ 0)

Passo 2: atualizar pesos

Como d = y, os pesos não precisam ser modificados

Exemplo 1: resposta b

b) Testar a rede

b.1) Para o padrão 111

u = 1(1.2) + 1(0.2) + 1(-0.2) -1(0.5) = 0.7

y = u = 1 (porque 0.7 $\geq$ 0)) $\Rightarrow$ classe 1

b.2) Para o padrão 000

u = 0(1.2) + 0(0.2) + 0(-0.2) -1(0.5) = -0.5

y = u = -1 (porque -0.5 < 0) $\Rightarrow$ classe 0

Exemplo 1: resposta b

b) Testar a rede

b.3) Para o padrão 100

u = 1(1.2) + 0(0.2) + 0(-0.2) +1(-0.5) = 0.7

y = u = 1 (porque 0.7 $\geq$ 0) $\Rightarrow$ classe 1

b.4) Para o padrão 011

u = 0(1.2) + 1(0.2) + 1(-0.2) -1(0.5) = -0.5

y = u = -1 (porque -0.5 < 0) $\Rightarrow$ classe 0

MULTILAYER PERCEPTRON

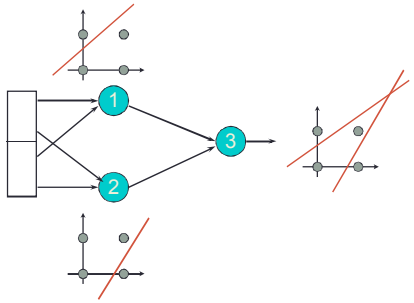
UFRPE Universidade Federal Rural de Pernambuco

DEINFO Departamento de Estatística e Informática

MLP - Introdução

- Redes de uma camada resolvem apenas problemas linearmente separáveis
- **Solução**: utilizar mais de uma camada
  - **Camada 1**: uma rede Perceptron para cada grupo de entradas linearmente separáveis
  - **Camada 2**: uma rede combina as saídas das redes da primeira camada, produzindo a classificação final
- Arquitetura de RNA mais popular
- Possuem uma ou mais camadas intermediárias de nós
  - Função de ativação mais utilizada é sigmóide logística

MLP - Introdução

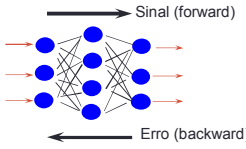


Treinamento de redes MLP

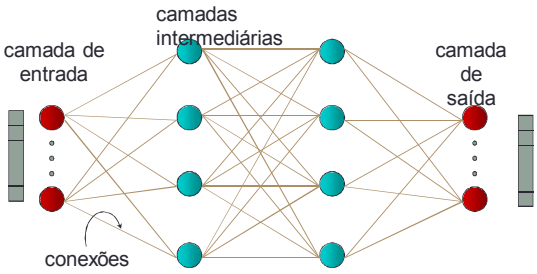
- Grande variedade de Algoritmos
  - Geralmente supervisionados
  - Estáticos
    - Não alteram estrutura da rede
    - ***Backpropagation***, Função de Base Radial
  - Construtivos
    - Alteram estrutura da rede
    - *Upstar, Cascade Correlation*

Backpropagation

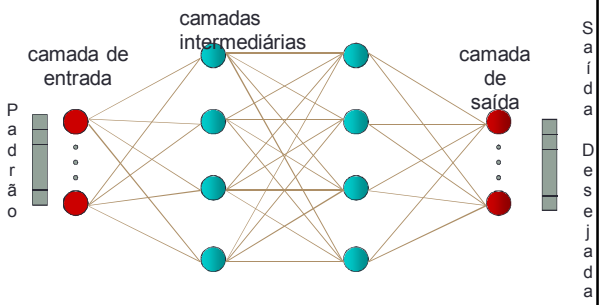
- Rede é treinada com pares entrada-saída
- Cada entrada de treinamento está associada a uma saída desejada
- Treinamento em duas fases, cada uma percorrendo a rede em um sentido
  - Fase *forward*
  - Fase *backward*



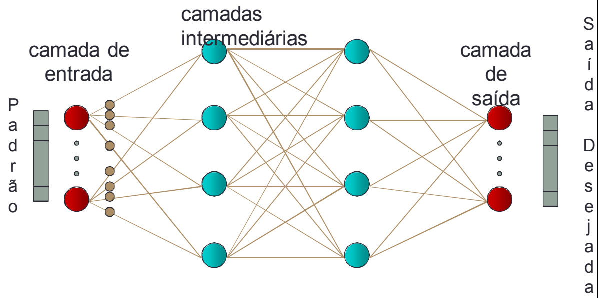
Rede MLP



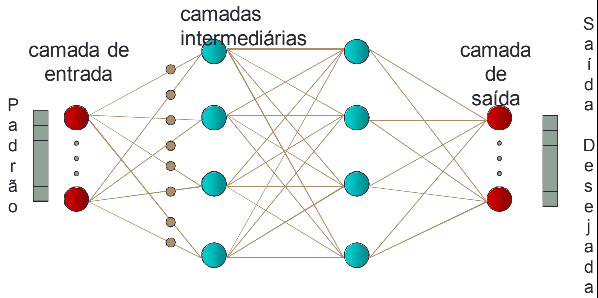
Aprendizado



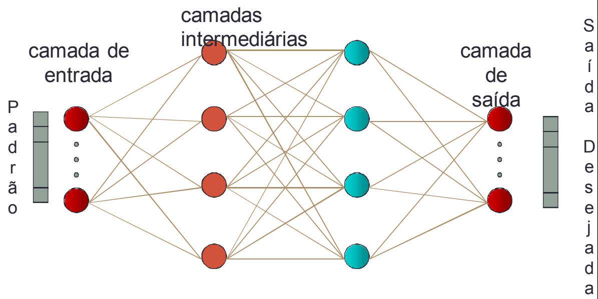
RNA - Aprendizado



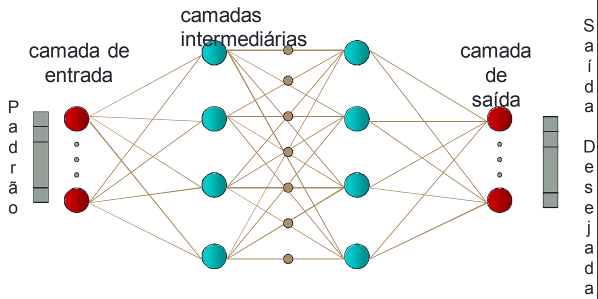
RNA - Aprendizado



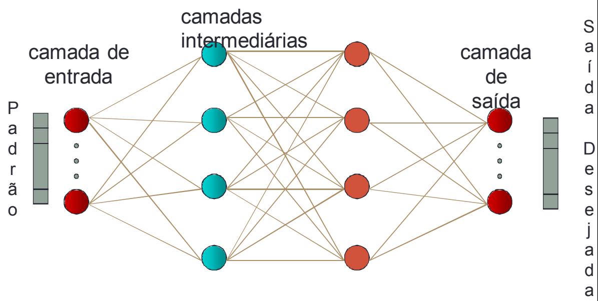
RNA - Aprendizado



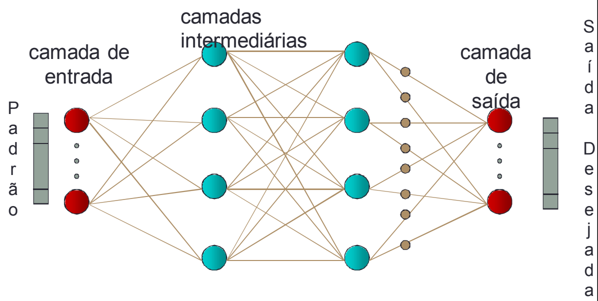
RNA - Aprendizado



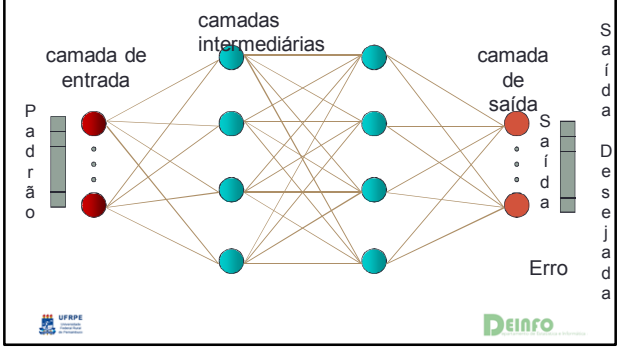
RNA - Aprendizado



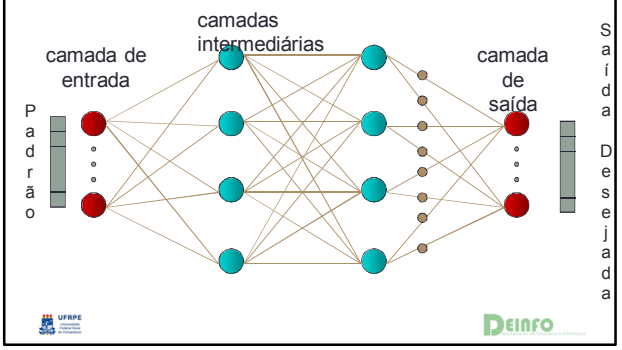
RNA - Aprendizado



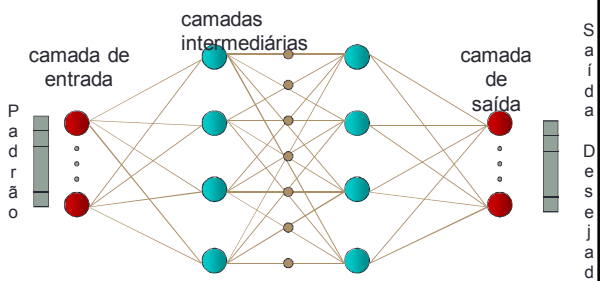
RNA - Aprendizado



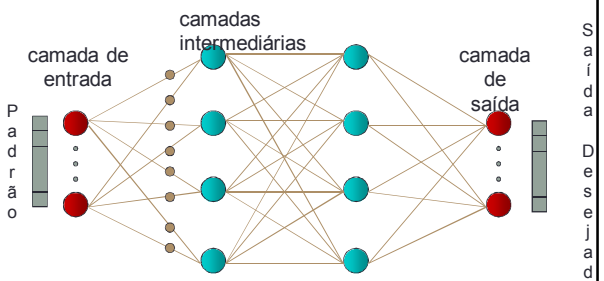
RNA - Aprendizado



RNA - Aprendizado



RNA - Aprendizado



Fase forward

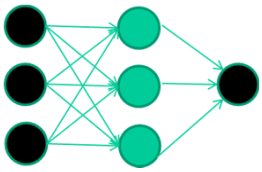
- Entrada é apresentada à primeira camada da rede
  - Após os neurônios da camada i calcularem seus sinais de saída, os neurônios da camada i + 1 calculam seus sinais de saída
- Saídas produzidas pelos neurônios da última camada são comparadas às saídas desejadas
- Erro para cada neurônio da camada de saída é calculado

Fase backward

- A partir da última camada
  - O nó ajusta seu peso de modo a reduzir o seu erro
  - Nós das camadas anteriores tem seu erro definidos por:
    - Erros dos nós da camada seguinte conectados a ele ponderados pelos pesos das conexões entre eles

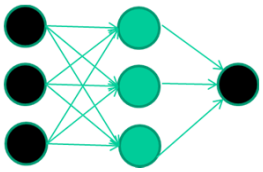
Dataset

Atrib.			*classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



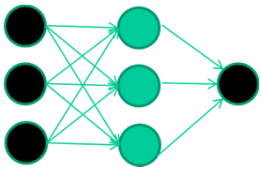
Treinando a rede neural

Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



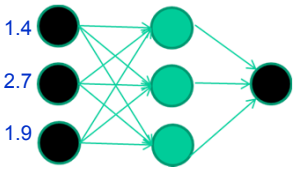
Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Initializar com pesos aleatórios



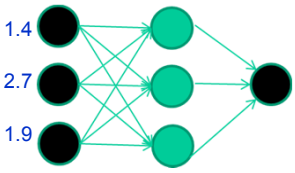
Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Apresentar um padrão de treinamento



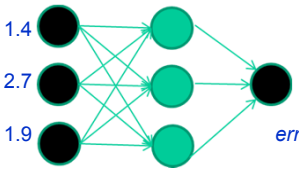
Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Calcular a saída dada pela rede



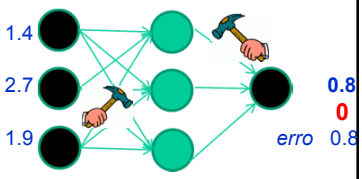
Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Comparar com a saída desejada



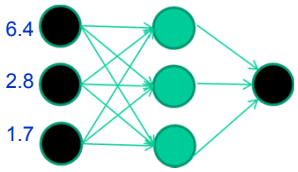
Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Ajustar pesos baseado no erro



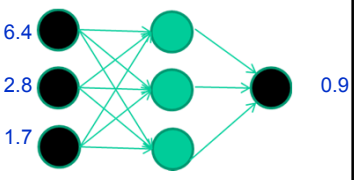
Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Apresentar um padrão de treinamento



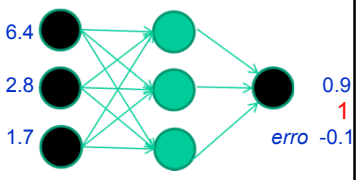
Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Calcular a saída dada pela rede



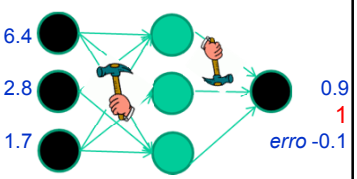
Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Comparar com a saída desejada



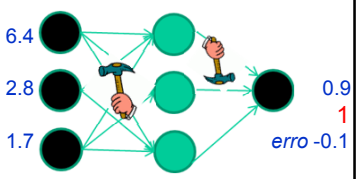
Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Ajustar pesos baseado no erro



Atrib.			*Classe*
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

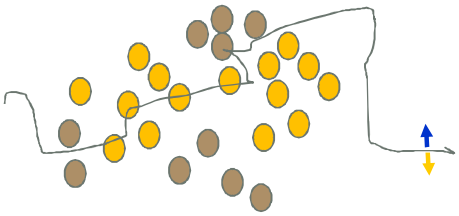
…



Repetir o processo milhares, talvez milhões de vezes

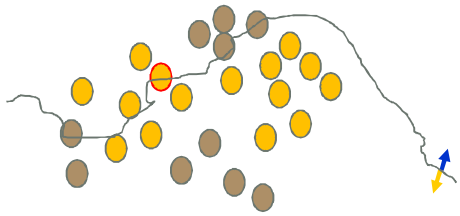
Superfície de decisão…

Pesos aleatórios



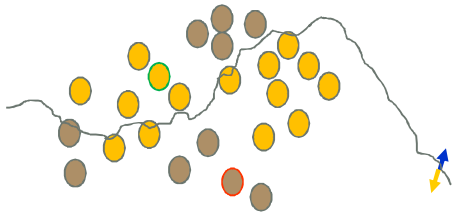
Superfície de decisão

Apresentar um padrão de treinamento / ajustar os pesos



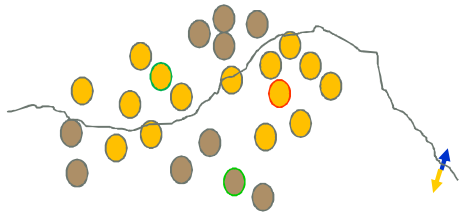
Superfície de decisão

Apresentar um padrão de treinamento / ajustar os pesos



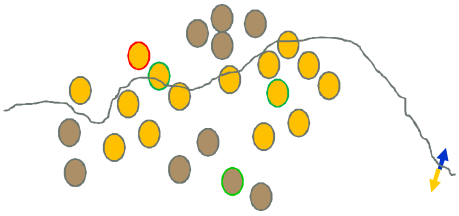
Superfície de decisão

Apresentar um padrão de treinamento / ajustar os pesos



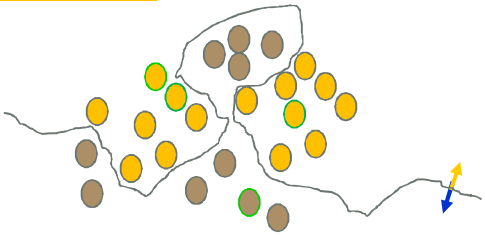
Superfície de decisão

Apresentar um padrão de treinamento / ajustar os pesos



Superfície de decisão

Eventualmente ….

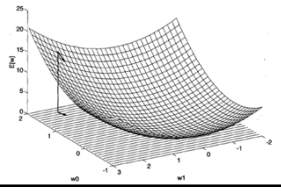


Backpropagation

- Supor que cada combinação de pesos corresponda a um ponto em uma superfície de solução
  - Solução = pontos mais baixos da superfície
  - Procura minimizar erro ajustando pesos e *thresholds* para que eles correspondam aos pontos mais baixos da superfície
    - método do gradiente descendente

Backpropagation

- **Gradiente** de uma função está na direção e sentido onde a função tem taxa de variação máxima
  - Garantido de achar uma solução para superfícies simples
- Backpropagation fornece aproximação da trajetória no espaço de peso computado pelo método do gradiente descendente



Backpropagation

- Processamento
  - Forward (teste)
  - Backward (treinamento)
- Estados de ativação
  - 1 (+1) = ativo
  - 0 (-1) = inativo

Backpropagation

- Função de ativação
  - Não linear
  - Diferenciável , contínua e, geralmente, não decrescente

(sigmoidal logística)

$$a(t+1) = \frac{1}{1+e^{-\lambda u(t)}}$$

(tang. hiperbólica)

$$a(t+1) = \frac{1-e^{-\lambda u(t)}}{1+e^{-\lambda u(t)}}$$

Backpropagation

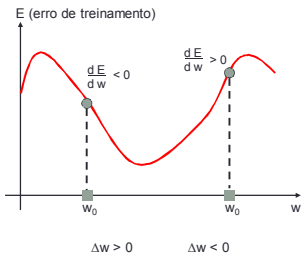
- Funcionamento do algoritmo:
  - Ponto de partida para obter a expressão de ajuste de pesos:

$$E = \frac{1}{2}\sum_j (d_j - y_j)^2$$

Erro para um padrão, considerando todos os nós de saída.

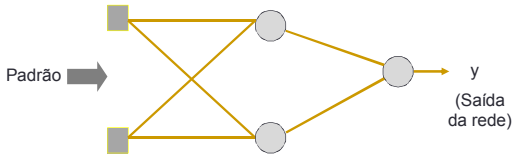
Backpropagation

- Interpretação gráfica: Busca do mínimo global.



Backpropagation

Fase Forward: Apresenta-se o padrão à rede, que gera uma saída.

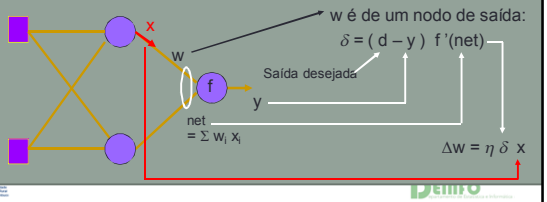


Backpropagation

Fase Backward: Ajusta os pesos da rede a partir da camada de saída.

$$\Delta w_{ji} = \eta\, \delta_j(t)\, x_i(t)$$

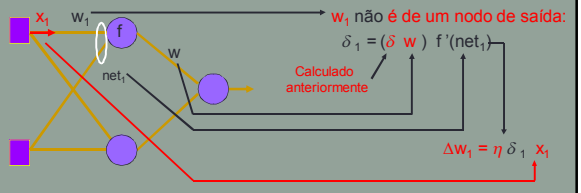
$$onde \quad \delta_j(t) = \begin{cases} (d_j - y_j)\, f'(net_j), & se\ for\ nodo\ de\ saída \\ (\sum_l \delta_l\, w_{lj})\, f'(net_j), & caso\ contrário \end{cases}$$



Backpropagation

$$\Delta w_{ji} = \eta\, \delta_j(t)\, x_i(t)$$

$$onde \quad \delta_j(t) = \begin{cases} (d_j - y_j)\, f'(net_j), & se\ for\ nodo\ de\ saída \\ (\sum_l \delta_l\, w_{lj})\, f'(net_j), & caso\ contrário \end{cases}$$



Treinamento

1) Iniciar todas as conexões com valores aleatórios
2) Repita
   erro = 0
   Para cada par de treinamento (X, d)
      Para cada camada k := 1 a N
         Para cada neurônio J := 1 a M_k
            Calcular a saída y_{jk}
   Se erro > ε
   Então Para cada camada k := N a 1
         Para cada neurônio J := 1 a M_k
            Atualizar pesos
   Até erro < ε

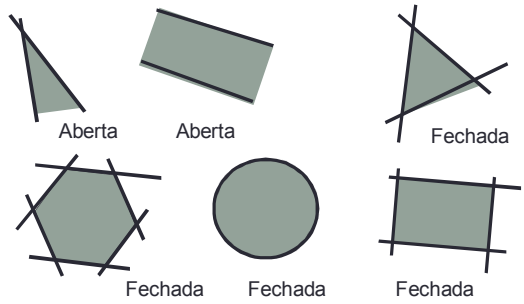
Teste

1) Apresentar padrão X a ser reconhecido
2) Para cada camada k := 1 a N
   Para cada neurônio J := 1 a M_k
      Calcular a saída y_{jk}
      Comparar saída y_{Nj} com d_{cj} para cada classe c
      Classificar padrão como pertencente a classe cuja saída desejada é mais próxima da saída produzida

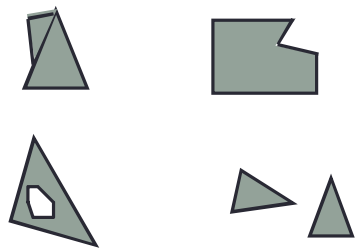
MLPs como classificadores

- Função implementada por cada neurônio é formada pela combinação das funções implementadas por neurônios da camada anterior
  - Camada 1: linhas retas no espaço de decisão
  - Camada 2: regiões convexas
    - Número de lados = número de unidades na camada 1
  - Camada 3: Combinações de figuras convexas, produzindo formatos abstratos

Regiões convexas



Combinações de regiões convexas



REDES NEURAIS
PARTE III

Unidades intermediárias

- Número de camadas intermediárias necessárias
  - **1 camada**: suficiente para aproximar qualquer função contínua ou Booleana
  - **2 camadas**: suficiente para aproximar qualquer função
  - **3 ou mais camadas**: pode facilitar o treinamento da rede
    - Cada vez que o erro é propagado para a camada anterior, ele se torna menos útil

Unidades intermediárias

- Número de neurônios nas camadas intermediárias
  - Em geral não é conhecido
  - Utilizar função do número de entradas e saídas
    - Não funciona
  - Número de pesos vezes dez é menor que o número de exemplos
    - Apenas reduz overfitting
    - Se o número de exemplos for muito maior que o número de pesos, overfitting é improvável, mas pode ocorrer underfitting

Unidades intermediárias

- Número de neurônios nas camadas intermediárias (cont.)
  - Depende de:
    - Número de exemplos de treinamento
    - Quantidade de ruído
    - Complexidade da função a ser aprendida
      - Distribuição estatística

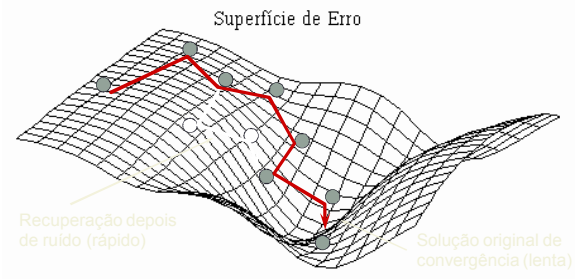
Unidades intermediárias

- Número de neurônios nas camadas intermediárias (cont.)
  - Existem problemas com uma entrada e uma saída que precisam de milhares de unidades e vice-versa
  - Pode crescer exponencialmente com o número de entradas
  - Solução neural eficiente: aquela onde o número de unidades cresce apenas polinomialmente com o número de entradas

Generalização

- Classificação correta de padrões não utilizados no treinamento ou com ruído
- Ocorre através da detecção de características relevantes do padrão de entrada
- Padrões desconhecidos são atribuídos a classes cujos padrões apresentam características semelhantes
- Tolerância a falhas

Generalização



Dificuldades de aprendizado

- Backpropagation é muito lento em superfícies complexas
  - Considerar efeitos de segunda ordem para gradiente descendente
- Mínimos locais: solução estável que não fornece saída correta
  - Taxa de aprendizado decrescente
  - Adicionar nós intermediários
  - Utilizar momentum
  - Adicionar ruído

Dificuldades de aprendizado

- Overfitting
  - Depois de um certo ponto do treinamento, a rede piora ao invés de melhorar
  - Memoriza padrões de treinamento, incluindo suas peculiaridades (piora generalização)
  - Alternativas
    - Encerrar treinamento cedo
    - Reduzir pesos

Atualização dos pesos

- Ciclo
  - Apresentação de todos os exemplos de treinamento durante o aprendizado
  - Exemplos devem ser apresentados em ordem aleatória
- Abordagens para atualização dos pesos
  - Por padrão (online)
  - Por ciclo (batch)

Atualização dos pesos

- Por padrão
  - Pesos atualizados após apresentação de cada padrão
  - Estável se taxa de aprendizado e momentum forem pequenos (reduzir progressivamente as taxas)
  - Altas taxas rede instável
  - Mais rápida, principalmente se o conjunto de treinamento for grande e redundante
  - Requer menos memória

Atualização dos pesos

- Por ciclo
  - Pesos atualizados depois que todos os padrões de treinamento forem apresentados
  - Geralmente mais estável
  - Pode ser lento se o conjunto de treinamento for grande e redundante
  - Estimativa mais precisa do vetor gradiente
- Método depende da aplicação

Atualização dos pesos

- Momentum
  - $\Delta w_{ij}(t + 1) = \eta x_i y_j (1 - y_j)\delta_j + \alpha(w_{ij}(t) - w_{ij}(t - 1))$
  - Aumenta velocidade de aprendizado evitando perigo de instabilidade
  - Pode acelerar treinamento em regiões muito planas da superfície de erro
  - Suprime oscilação de pesos em vales e ravinas

Dicas para melhorias

- Projeto de uma RNA utilizando backpropagation é mais uma arte que uma ciência
  - Envolve inúmeros fatores
  - Resultado da experiência do projetista
- Utilizar função sigmoidal assimétrica (tangente hiperbólica)
  - Aprendizado mais rápido (em geral)

Dicas para melhorias

- Resposta desejada deve estar $[-a + \varepsilon, a - \varepsilon]$
  - a = valor máximo da função de ativação
- Inicialização dos pesos e thresholds deve ser uniformemente distribuído dentro de um intervalo pequeno
  - Reduz probabilidade dos neurônios saturarem
  - Intervalos muito pequenos podem tornar treinamento lento
  - Geralmente utiliza-se (-2.4/fan_in, + 2.4/fan_in)