



Inteligência Artificial

Profa. Patrícia R. Oliveira
EACH / USP

Parte 6 – Aprendizado Conexionista: Redes
Neurais Artificiais
(Modelos Perceptron)



Introdução

- Modelos que incorporam funções matemáticas (complexas).
- Podem ser usadas para a tarefa de classificação:
 - tomam uma instância como entrada e produzem uma saída, que é interpretada como a classe estimada pelo modelo.



Aprendizado de funções

- Aprendizado do mapeamento de instâncias em categorias (classes):
 - cada categoria é dada por um número.
 - ou por um intervalo de valores reais (por ex., 0.5 – 0.9).
- Exemplos de aprendizado de funções:
 - Entrada = 1, 2, 3, 4 Saída = 1, 4, 9, 16
 - Aqui, o conceito a ser aprendido é o quadrado dos números inteiros.
 - Entrada = [1,2,3], [2,3,4], [3,4,5], [4,5,6]
 - Saída = 1, 5, 11, 19
 - Aqui, o conceito é: $[a,b,c] \rightarrow a*c - b$

Exemplo: Classificando Veículos

- Entrada para a função: dados de pixels obtidos de imagens de veículos.
 - Saída: números: 1 para carro; 2 para ônibus; 3 para tanque

INPUT



OUTPUT = 3

INPUT



OUTPUT = 2

INPUT



OUTPUT = 1

INPUT



OUTPUT=1



Por que usar Redes Neurais?

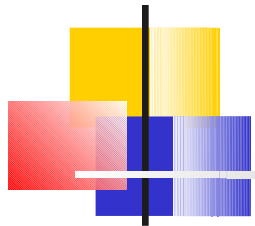
- **Motivação biológica:**

- O cérebro faz com que tarefas de classificação pareçam fáceis.
- O processamento cerebral é realizado por redes de neurônios.
- Cada neurônio é conectado a vários outros neurônios.

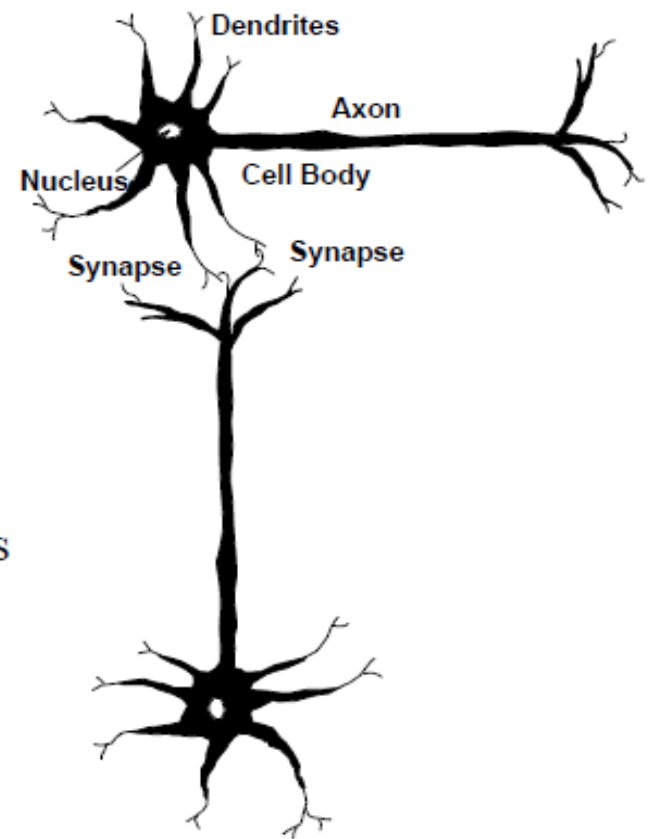
- **Redes Neurais “Naturais”:**

- A entrada de um neurônio é formada pelas saídas de vários outros neurônios.
- Um neurônio é ativado se a soma ponderada de suas entradas $>$ limiar.

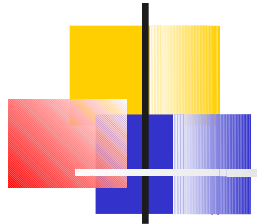
O neurônio biológico



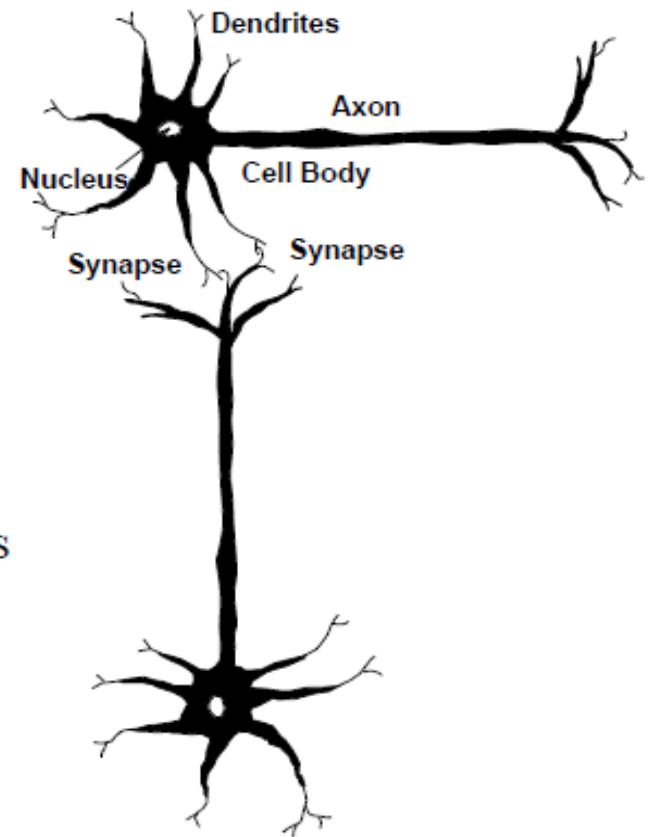
- O neurônio recebe impulsos (sinais) de outros neurônios por meio dos seus dendritos.
- O neurônio envia impulsos para outros neurônios por meio do seu axônio.
- O axônio termina num tipo de contato chamado sinapse, que conecta-o com o dendrito de outro neurônio.



O neurônio biológico



- A sinapse libera substâncias químicas, chamadas de neurotransmissores, em função do pulso elétrico disparado pelo axônio.
- O neurônio envia impulsos para outros neurônios por meio do seu axônio.
- O fluxo de neurotransmissores nas sinapses pode ter um efeito excitatório ou inibitório sobre o neurônio receptor.





Processo de aprendizado

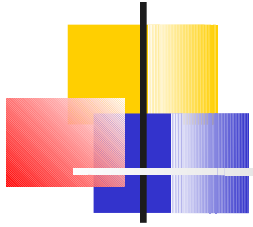
- O aprendizado ocorre por sucessivas modificações nas sinapses que interconectam os neurônios, em função da maior ou menor liberação de neurotransmissores.
- A medida que novos eventos ocorrem, determinadas ligações entre neurônios são reforçadas, enquanto outras são enfraquecidas.
- Este ajuste nas ligações entre os neurônios durante o processo de aprendizado é uma das mais importantes características das redes neurais artificiais.



Redes Neurais Artificiais (RNAs)

- Redes Neurais Artificiais (RNAs)
 - Hierarquia similar ao funcionamento do sistema biológico.
 - Neurônios que podem ser ativados por estímulos de entrada.
 - Mas essa analogia não vai muito longe...
 - Cérebro humano: aproximadamente 100.000.000.000 de neurônios.
 - RNAs: < 1000 geralmente

Ideia Geral



Valor calculado usando todos os valores das unidades de entrada

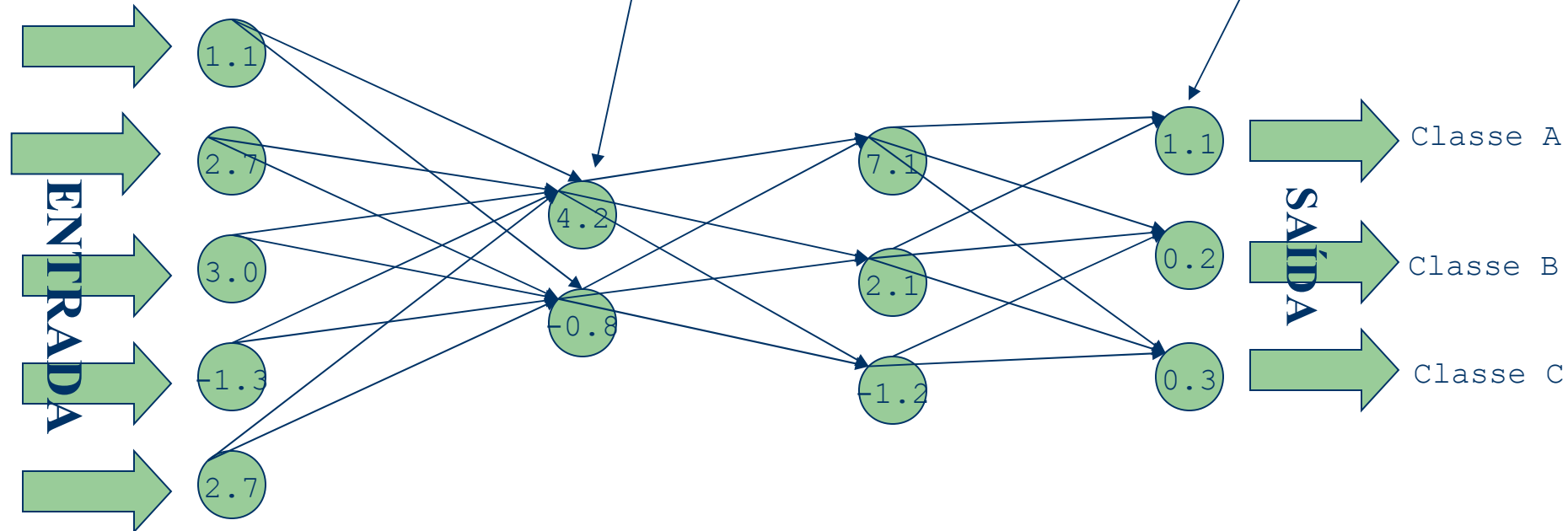
Escolhe a Classe A
(maior valor de saída)

CAMADA DE ENTRADA

CAMADA OCULTA

CAMADA DE SAÍDA

CLASSE



OS VALORES SE PROPAGAM ATRAVÉS DA REDE



Processamento das RNAs

- Cada unidade da rede realiza o mesmo cálculo.
 - geralmente baseado na soma ponderada das entradas na unidade.
- O conhecimento obtido pela rede fica armazenado nos pesos correspondentes a cada uma de suas unidades (neurônios).
- Representação “Caixa Preta”:
 - É difícil extrair o conhecimento sobre o conceito aprendido.



Aprendizado Supervisionado em RNAs

- Dados: conjunto de exemplos rotulados e representados numericamente.
- Tarefa: treinar uma rede neural usando esses exemplos.
 - O desempenho deve ser medido pela capacidade da rede em produzir saídas corretas para dados não contidos no conjunto de treinamento.



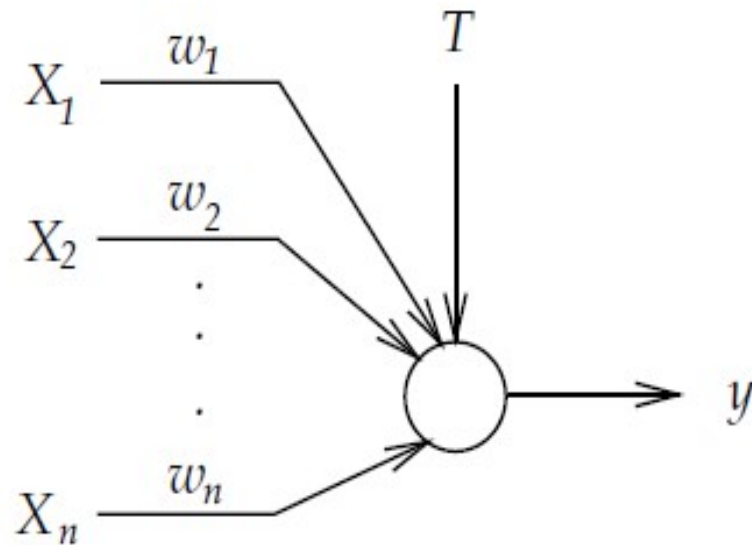
Aprendizado Supervisionado em RNAs

- Etapas preliminares ao treinamento:
 - escolha da arquitetura de rede correta.
 - número de neurônios
 - número de camadas ocultas
 - escolha da função de ativação (a mesma) para cada neurônio.
- A etapa de treinamento resume-se a:
 - ajustar os pesos das conexões entre as unidades para que a rede produza saídas corretas.

Perceptrons

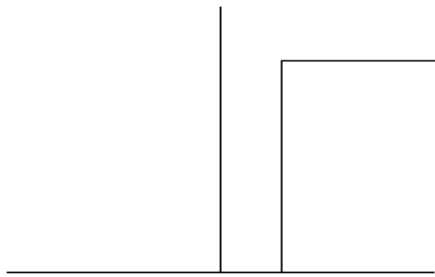
- O tipo mais simples de Rede Neural.
- Possui um único neurônio de saída.
 - Considera a soma ponderada das entradas.
 - A função de ativação da unidade calcula a saída da rede.
 - Exemplo: unidade com threshold (limiar) linear.

$$\begin{aligned} - \text{Netinput} &= \sum_{i=1}^n x_i w_i \\ - \text{if } \text{Netinput} &\geq T \text{ then } y = 1 \text{ else } y = 0 \end{aligned}$$

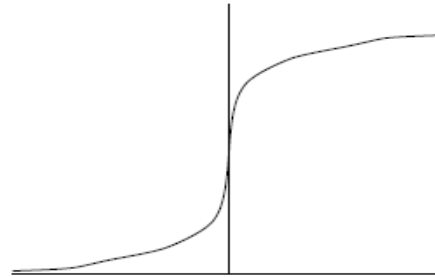


Perceptrons

- Algumas funções de transferência:

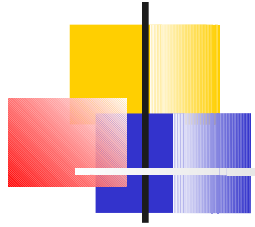


Função threshold



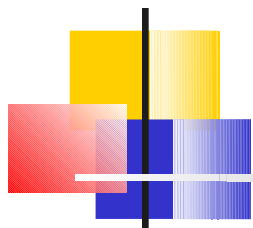
Função logística/sigmóide

- Função Step (degrau):
 - Saída +1 se $Netinput > Threshold\ T$
 - Saída -1 caso contrário
 - Aqui, os dados binários são representados por +1 e -1
- Problema principal: como aprender os valores dos pesos da rede?

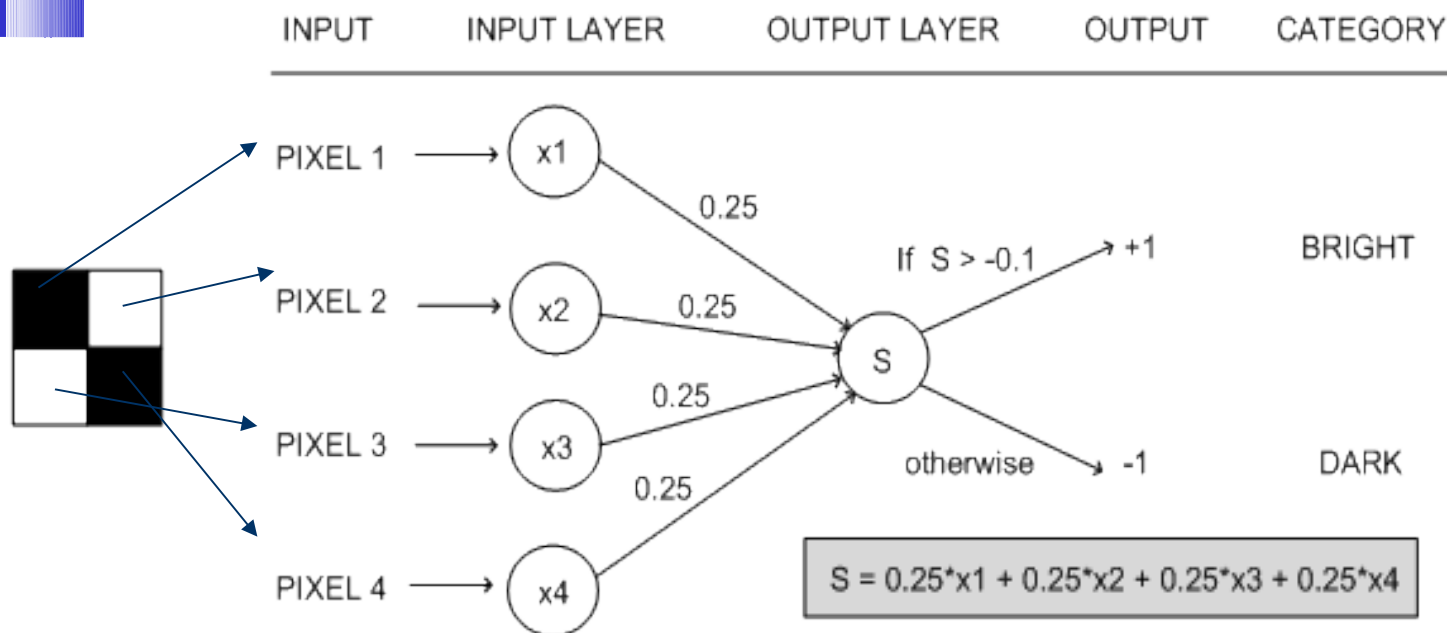


Perceptrons: Exemplo

- Classificação de imagens preto e branco representadas por uma matriz de pixels 2x2.
 - Em “clara” ou “escura”.
- Pode-se representar o problema por essa regra:
 - Se apresentar 2, 3 ou 4 pixels brancos, é “clara”.
 - Se apresentar 0 ou 1 pixels brancos, é “escura”.
- Arquitetura do Perceptron:
 - Quatro unidades de entrada, uma para cada pixel.
 - Uma unidade de saída.
 - +1 para “clara”, -1 para “escura”.



Perceptrons: Exemplo



- Exemplo de entrada: $x_1 = -1$, $x_2 = 1$, $x_3 = 1$, $x_4 = -1$
 $S = 0.25 \cdot (-1) + 0.25 \cdot (1) + 0.25 \cdot (1) + 0.25 \cdot (-1) = 0$
- $0 > -0.1$, portanto a saída para a rede é +1
 - A imagem é classificada como "clara"

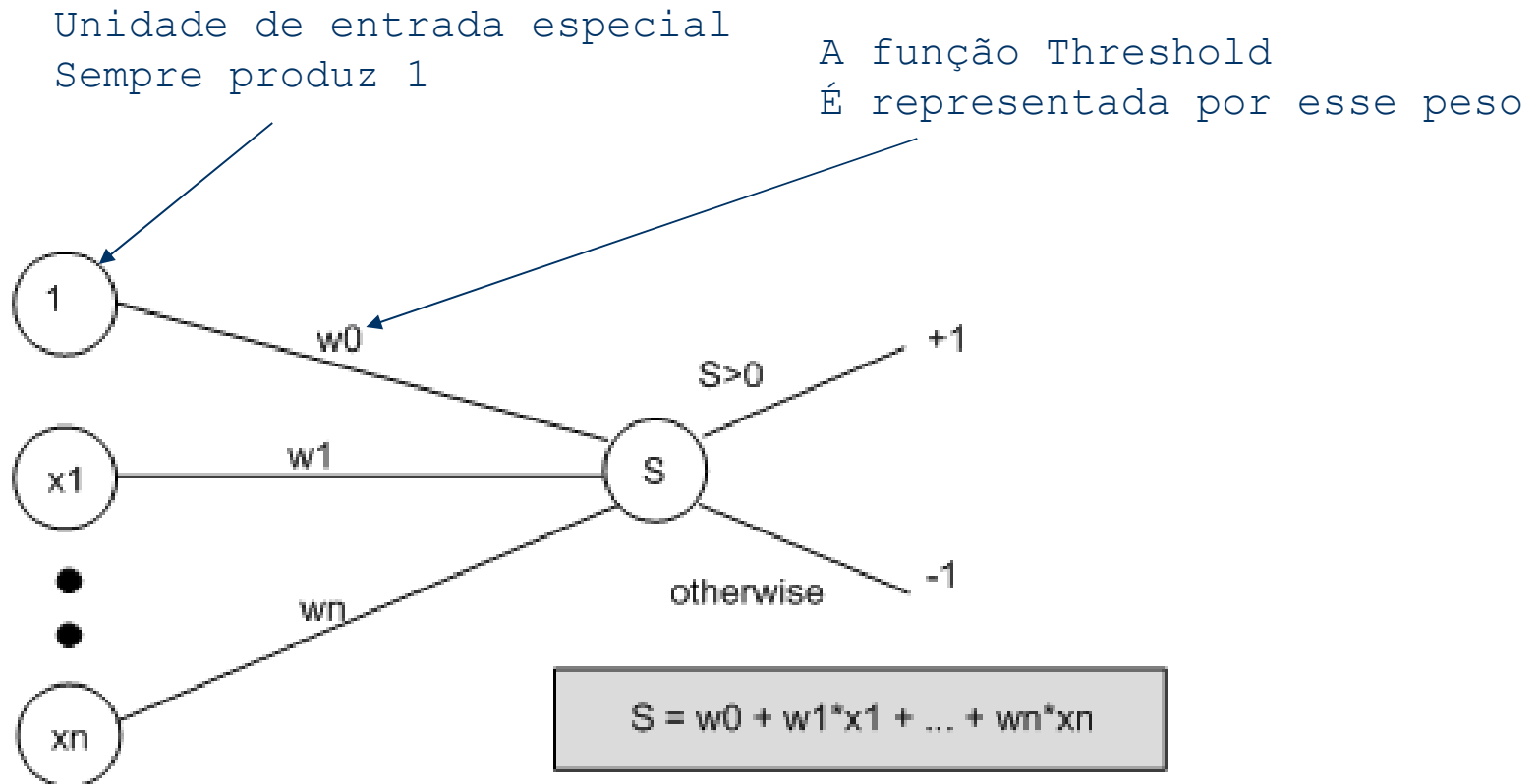


Aprendizagem em Perceptrons

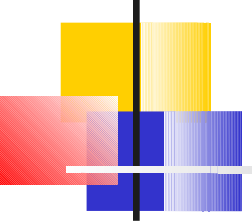
- É necessário aprender:
 - Os pesos entre as unidades de entrada e saída.
 - O valor do threshold.
- Para tornar os cálculos mais fáceis:
 - Considera-se o threshold como um peso referente a uma unidade de entrada especial, cujo sinal é sempre 1 (ou -1).
 - Agora, o único objetivo resume-se a aprender os pesos da rede.

Representação

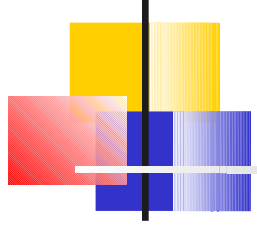
Alternativa para Perceptrons



Perceptrons: Algoritmo de Aprendizagem (1)

- 
- Os valores dos pesos são inicializados aleatoriamente, geralmente no intervalo $(-1, 1)$.
 - Para cada exemplo de treinamento E :
 - Calcule a saída observada da rede $o(E)$.
 - Se a saída desejada $t(E)$ for diferente de $o(E)$:
 - Ajuste os pesos da rede, para que $o(E)$ chegue mais próximo de $t(E)$.
 - Isso é feito aplicando-se a regra de aprendizado do Perceptron.

Perceptrons: Algoritmo de Aprendizagem (2)



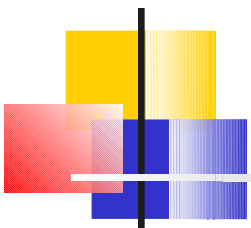
- O processo de aprendizado não para necessariamente depois de todos os exemplos terem sido apresentados.
 - Repita o ciclo novamente (uma “época”).
 - Até que a rede produza saídas corretas (ou boas o suficiente).
 - Considerando todos os exemplos no conjunto de treinamento.



Regra de Aprendizagem para Perceptrons

- Quando $t(E)$ for diferente de $o(E)$
 - Adicione Δ_i ao peso w_i
 - Em que $\Delta_i = \eta(t(E) - o(E))x_i$
 - Faça isso para todos os pesos da rede.

Regra de Aprendizagem para Perceptrons



- Interpretação:

$(t(E) - o(E))$ será igual a $+2$ ou -2

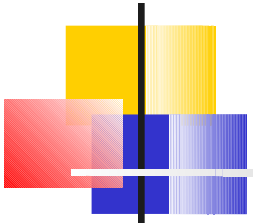
- Portanto, pode-se pensar na adição de Δ_i como uma movimentação do peso em uma determinada direção.
 - que irá melhorar o desempenho da rede com relação a E .
- Multiplicação por x_i
- O movimento aumenta proporcionalmente ao sinal de entrada.



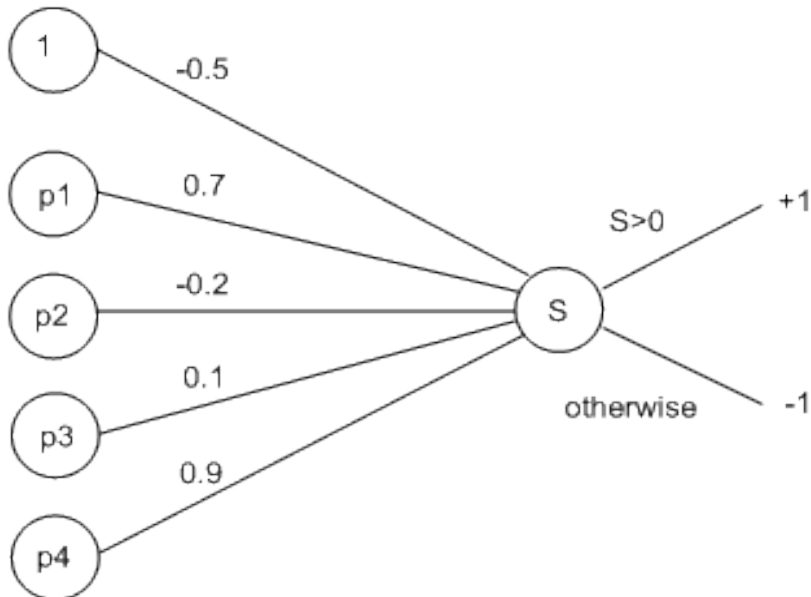
Taxa de Aprendizado

- O parâmetro η é chamado de taxa de aprendizagem.
 - Geralmente escolhido como uma pequena constante entre 0 e 1 (por exemplo, 0.1).
- Controla o movimento dos pesos.
 - não deixa haver uma mudança grande para um único exemplo.
- Se uma mudança grande for mesmo necessária para que os pesos classifiquem corretamente um determinado exemplo:
 - Essa deve ocorrer gradualmente, em várias épocas.

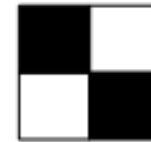
Exemplo anterior: classificar uma imagem em “clara” ou “escura”



- 1) Suponha que a rede Perceptron em treinamento presente, em um dado instante de tempo, o seguinte conjunto de pesos:

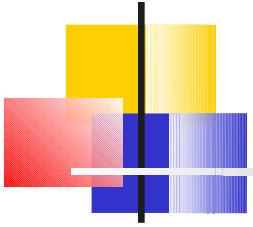


- 2) Use o exemplo de treinamento, e_1 , abaixo, para atualizar os pesos da rede:



- Use a taxa de aprendizado $\eta = 0.1$

Exemplo anterior: classificar uma imagem em “clara” ou “escura”

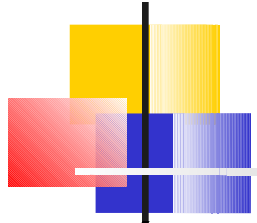


■ Solução:



- Aqui, $x_1 = -1$, $x_2 = 1$, $x_3 = 1$, $x_4 = -1$
- Propagando essa informação através da rede:
 - $S = (-0.5 * 1) + (0.7 * -1) + (-0.2 * +1) + (0.1 * +1) + (0.9 * -1) = -2.2$
- Portanto, a saída da rede é $o(e_1) = -1$ (“escura”)
- Mas deveria ter sido $+1$ (“clara”)
 - Portanto $t(e_1) = +1$

Exemplo anterior: classificar uma imagem em “clara” ou “escura”



- Cálculo dos valores de erro:

- $\Delta_0 = \eta(t(E)-o(E))x_0$

- $= 0.1 * (1 - (-1)) * (1) = 0.1 * (2) = 0.2$

- $\Delta_1 = \eta(t(E)-o(E))x_1$

- $= 0.1 * (1 - (-1)) * (-1) = 0.1 * (-2) = -0.2$

- $\Delta_2 = \eta(t(E)-o(E))x_2$

- $= 0.1 * (1 - (-1)) * (1) = 0.1 * (2) = 0.2$

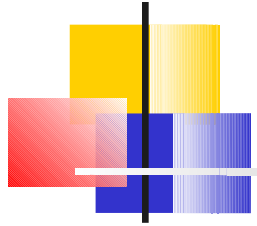
- $\Delta_3 = \eta(t(E)-o(E))x_3$

- $= 0.1 * (1 - (-1)) * (1) = 0.1 * (2) = 0.2$

- $\Delta_4 = \eta(t(E)-o(E))x_4$

- $= 0.1 * (1 - (-1)) * (-1) = 0.1 * (-2) = -0.2$

Exemplo anterior: classificar uma imagem em “clara” ou “escura”



- Ajuste dos pesos:

- $w'_0 = -0.5 + \Delta_0 = -0.5 + 0.2 = -0.3$

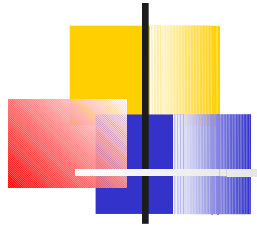
- $w'_1 = 0.7 + \Delta_1 = 0.7 + -0.2 = 0.5$

- $w'_2 = -0.2 + \Delta_2 = -0.2 + 0.2 = 0$

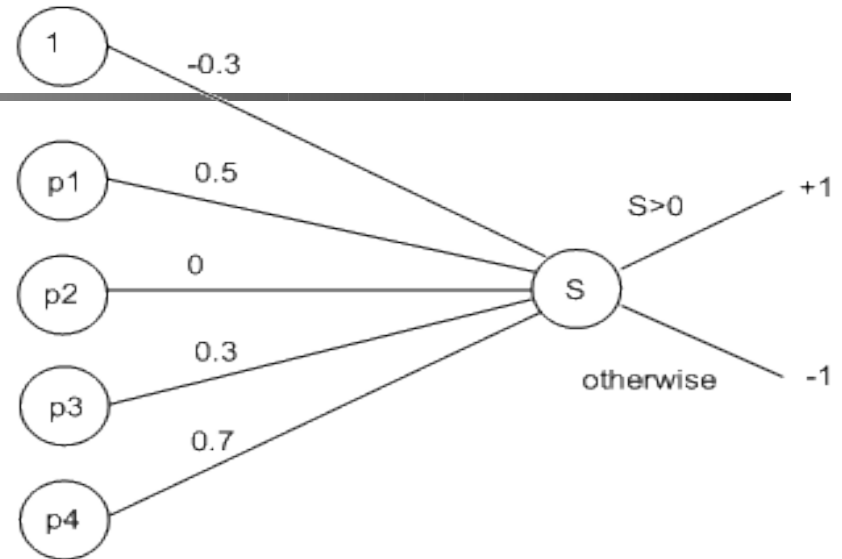
- $w'_3 = 0.1 + \Delta_3 = 0.1 + 0.2 = 0.3$

- $w'_4 = 0.9 + \Delta_4 = 0.9 - 0.2 = 0.7$

Exemplo anterior: classificar uma imagem em “clara” ou “escura”



- Nova configuração da rede:



- Calcule a saída para o exemplo, e_1 , novamente:

$$S = (-0.3 * 1) + (0.5 * -1) + (0 * +1) + (0.3 * +1) + (0.7 * -1) = -1.2$$

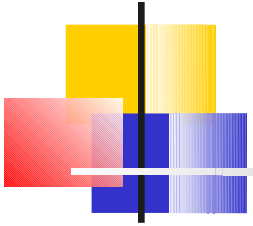
- Portanto, a nova saída da rede é $o(e_1) = -1$ (“escura”)

- Ainda resulta em classificação errada.

- Mas o valor de S já está mais próximo de zero (de -2.2 para -1.2)

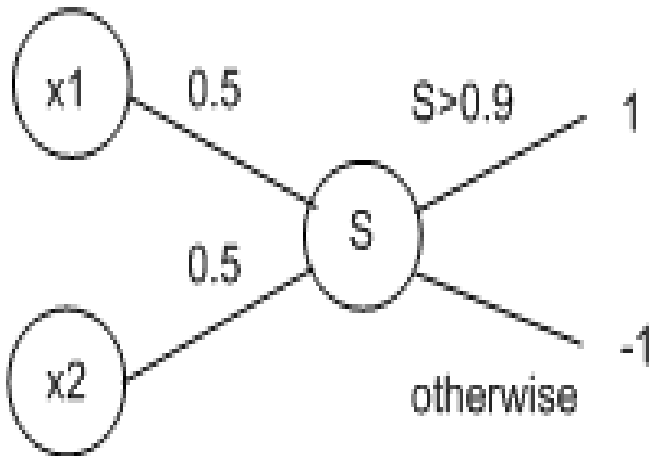
- Em poucas épocas, esse exemplo será classificado corretamente.

Exemplo: Aprendizado de Funções Booleanas

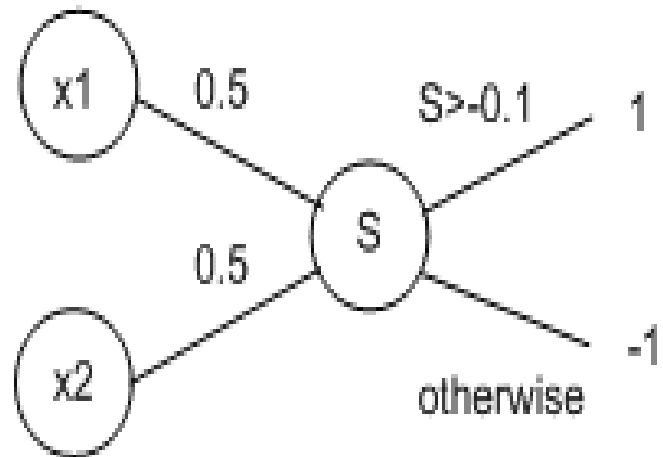


- Entradas assumem dois valores possíveis (+1 ou -1).
- Produz um valor como saída (+1 ou -1).
 - Exemplo 1: Função AND
 - Produz +1 somente se ambas as entradas forem iguais a +1.
 - Exemplo 2: Função OR
 - Produz +1 se pelo menos uma das entradas for igual a +1.

Exemplo: Aprendizado de Funções Booleanas

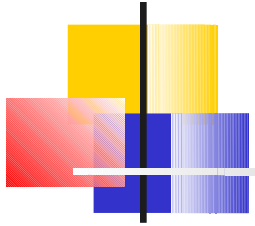


An ANN for AND

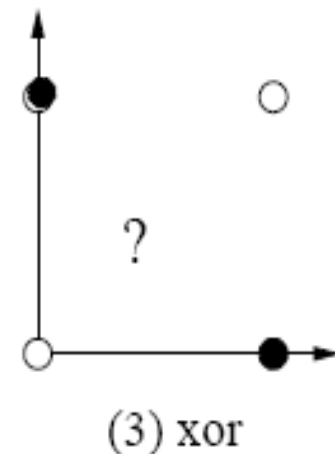
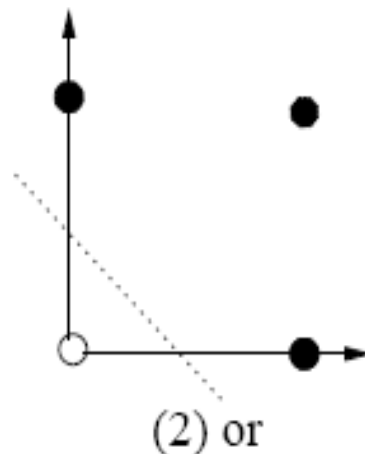
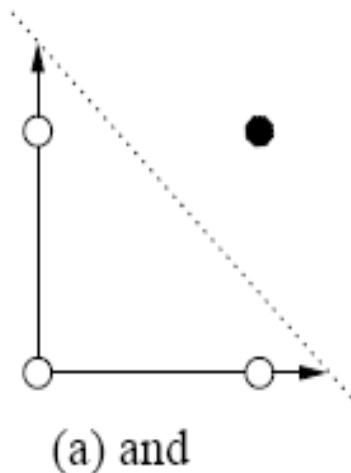


An ANN for OR

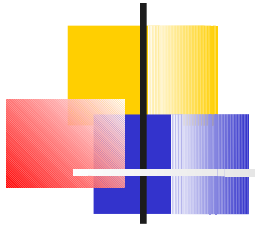
Capacidade de Aprendizado da Rede Perceptron



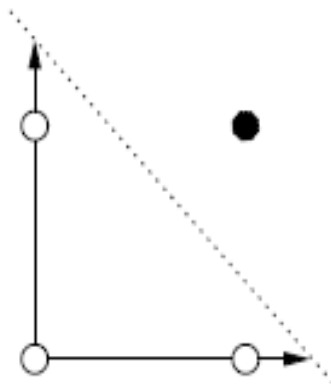
- O que a rede neural Perceptron é capaz de aprender?
 - somente a discriminação de classes que sejam linearmente separáveis.
 - Por exemplo, as funções booleanas AND e OR.



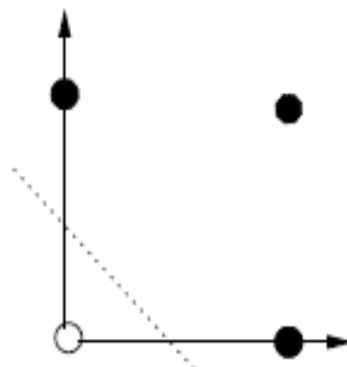
Capacidade de Aprendizado da Rede Perceptron



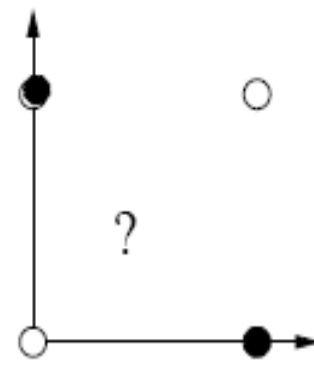
- Redes Perceptron não conseguem aprender a função XOR.
 - provado em 1969 por Minsky e Papert.
- A função XOR não é linearmente separável.
 - Não é possível traçar uma linha divisória que classifique corretamente todos os pontos.



(a) and

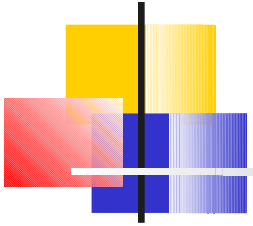


(2) or



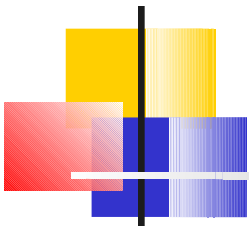
(3) xor

Redes Perceptron Multicamadas



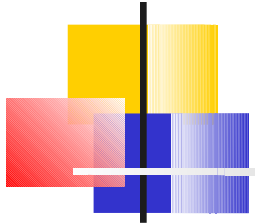
- Redes Perceptron não são capazes de aprender conceitos complexos.
- Porém, os perceptrons formam a base para a construção de um tipo de rede que pode aprender conceitos mais sofisticados.
 - Redes Perceptron Multicamadas (Multilayer Perceptron – MLP).
 - Pode-se pensar nesse modelo como sendo uma rede formada por vários neurônios similares ao “tipo perceptron”.

Redes Perceptron Multicamadas

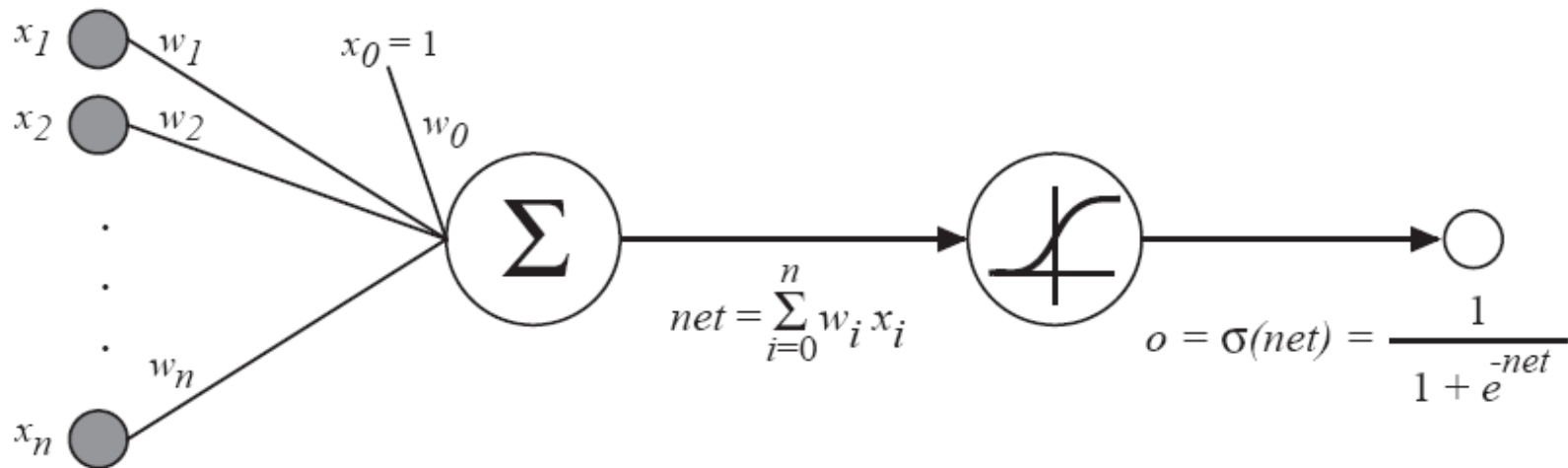


- Limitações das unidades Perceptron
 - A regra de aprendizado na MLP baseia-se em cálculo diferencial.
 - Funções do tipo degrau não são diferenciáveis.
 - Não são contínuas no valor do threshold.
 - Uma função de ativação alternativa deve ser considerada.
 - Tem que ser diferenciável.

Unidades com função sigmóide



- Unidades com função de ativação sigmóide podem ser usadas em redes MLP.

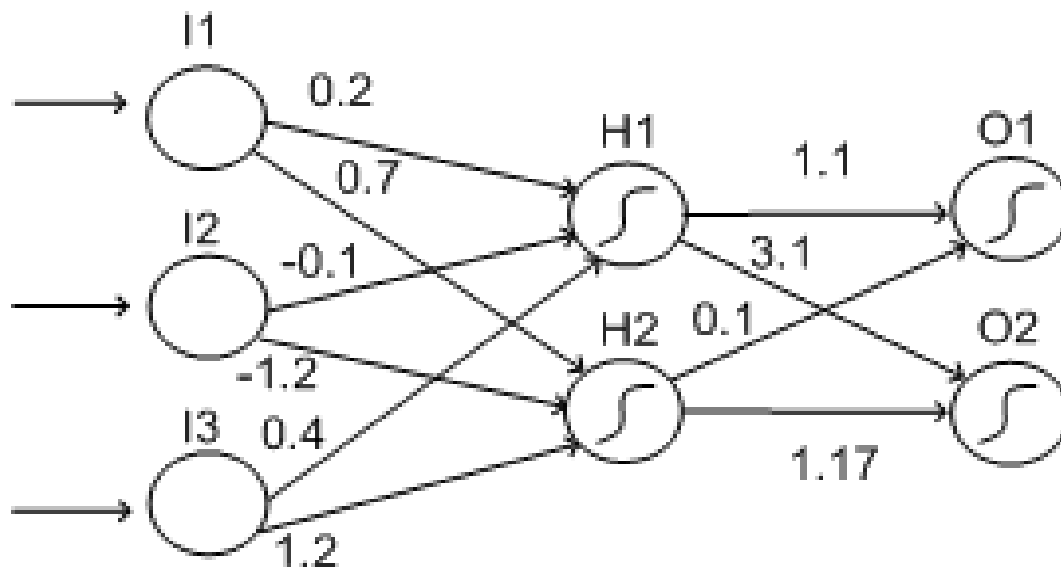


- Função sigmóide: $\sigma(x) = \frac{1}{1 + e^{-x}}$

- Derivada da função sigmóide: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$

Exemplo de MLP

- Considere a seguinte MLP já treinada e que classifica um exemplo como sendo da classe 1 se $O1 > O2$ e da classe 2, caso contrário.
- Qual a classe estimada pela rede para o exemplo: $[10, 30, 20]$?





Exemplo de MLP

- Primeiro, calcule as somas ponderadas para a camada oculta:

$$S_{H1} = (0.2*10) + (-0.1*30) + (0.4*20) = 2-3+8 = 7$$

$$S_{H2} = (0.7*10) + (-1.2*30) + (1.2*20) = 7-6+24 = -5$$

- A seguir, calcule a saída da camada oculta:

- Usando: $\sigma(S) = 1/(1 + e^{-S})$

- $\sigma(S_{H1}) = 1/(1 + e^{-7}) = 1/(1+0.000912) = 0.999$

- $\sigma(S_{H2}) = 1/(1 + e^5) = 1/(1+148.4) = 0.0067$



Exemplo de MLP

- A seguir, calcule as somas ponderadas para a camada de saída:

$$S_{O1} = (1.1 * 0.999) + (0.1 * 0.0067) = 1.0996$$

$$S_{O2} = (3.1 * 0.999) + (1.17 * 0.0067) = 3.1047$$

- Finalmente, calcule a saída da rede:

- Usando: $\sigma(S) = 1/(1 + e^{-S})$

- $\sigma(S_{O1}) = 1/(1 + e^{-1.0996}) = 1/(1+0.333) = 0.750$

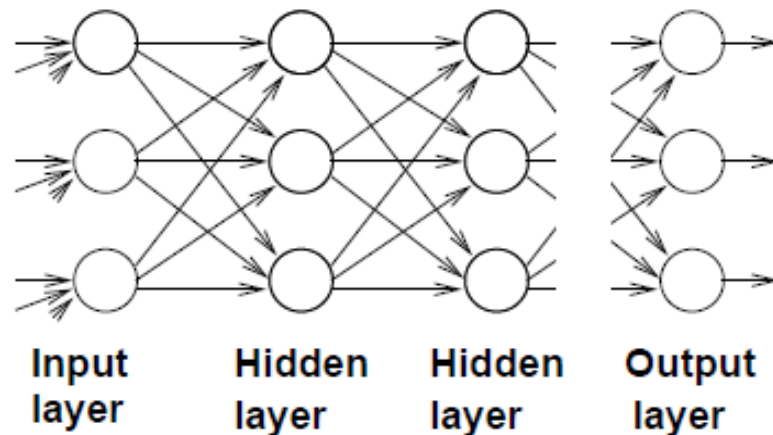
- $\sigma(S_{O2}) = 1/(1 + e^{-3.1047}) = 1/(1+0.045) = 0.957$

- Como a saída do neurônio O2 > saída do neurônio O1

- a classe estimada para o exemplo é a classe 2.

Características da MLP

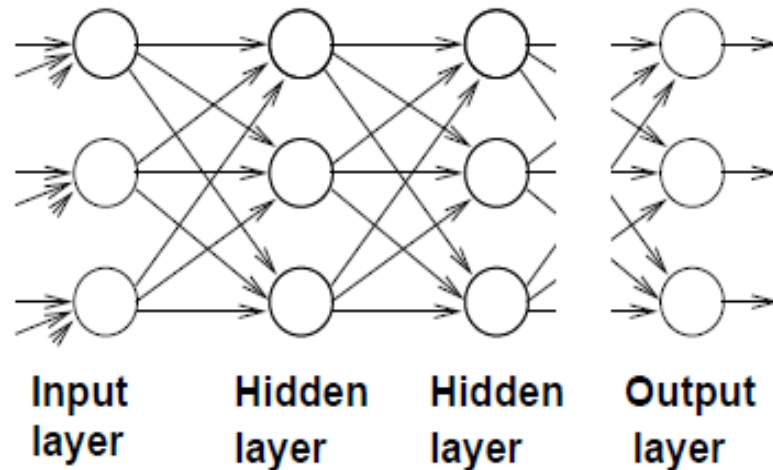
- Rede Neural do tipo “feedforward”:
 - Alimentação de entradas pela camada mais à esquerda;
 - Propagação dos sinais para frente através da rede;
 - Neurônios entre camadas vizinhas estão completamente conectados.





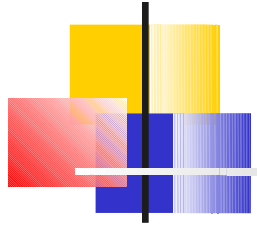
Características da MLP

- Camada de entrada: exemplos (sinais) de entrada.
- Camada(s) oculta(s): necessária(s) para o aprendizado de funções complexas.
- Camada de saída: saídas da rede.



Esquema de Aprendizagem

para a rede MLP



1) Obtenha um conjunto de dados rotulados.

a saída desejada para cada exemplo deve ser conhecida!

2) Gere um conjunto de pesos com valores aleatórios para a rede (por exemplo entre -1 e 1).

Enquanto o critério de convergência não for alcançado, faça:

Para todos os exemplos do conjunto de treinamento:

3) Apresente um exemplo para a rede e calcule as suas saídas. A diferença entre a saída da rede e a saída desejada é considerada como o valor de erro para essa iteração.

4) Ajuste os pesos da rede.

5) Volte para o passo 3.

Algoritmo Backpropagation

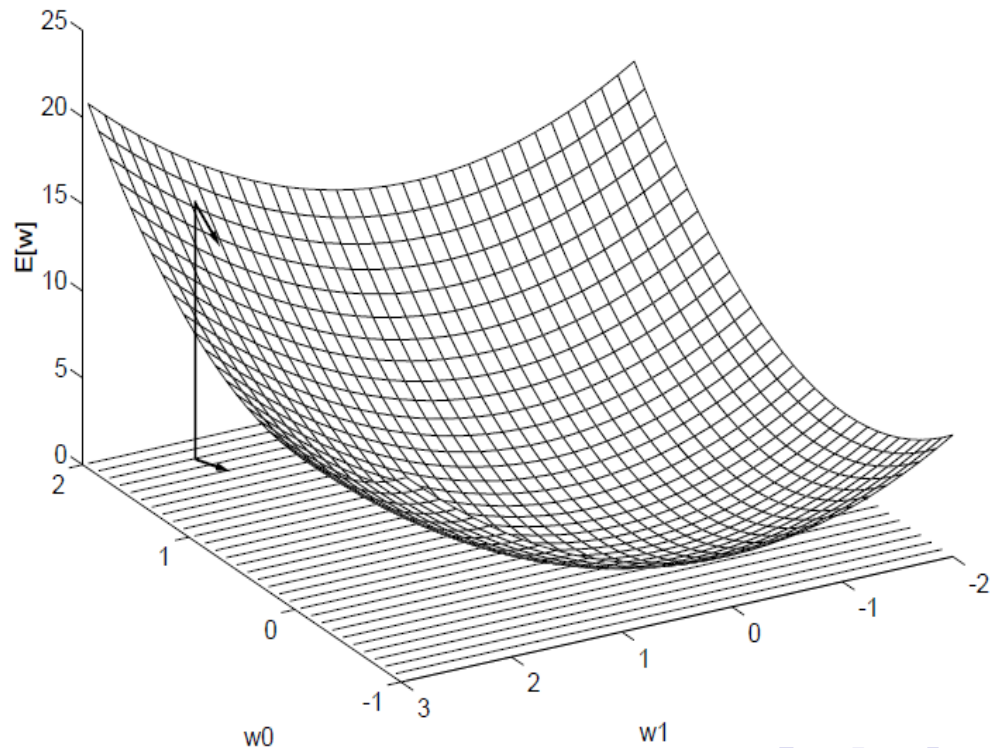
– Idéia Geral

- O ajuste de pesos em uma rede MLP se dá por meio da aplicação do algoritmo de aprendizagem Backpropagation.

- Idéia geral: $Erro = f(w_{ij})$

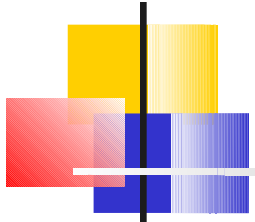
- Deseja-se minimizar o valor de Erro.

- Problema de otimização multidimensional.



Algoritmo Backpropagation

– versão estocástica



- Obs: aqui considera-se uma MLP com apenas uma camada oculta.
- Inicialize todos os pesos da rede com pequenos valores aleatórios.
- Enquanto o critério de convergência não for alcançado, faça:
 - Para todos os exemplos no conjunto de treinamento, faça:

1) Apresente um exemplo para a rede e calcule as suas saídas.

2) Para cada neurônio k , da camada de saída faça:

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

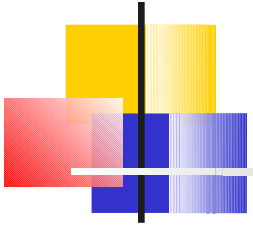
3) Para cada neurônio h , da camada oculta faça:

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in \text{outputs}} w_{hk} \delta_k$$

4) Ajuste cada peso w_{ij} da rede

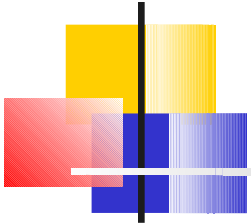
$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \quad \text{where} \quad \Delta w_{ij} = \eta \delta_j x_{ij}$$

Ajuste dos pesos via Backpropagation



- A notação w_{ij} é usada para:
 - especificar o peso da conexão entre o neurônio i e o neurônio j .
- Para cada exemplo, calcule o ajuste Δw_{ij} para cada peso w_{ij} da rede.
 - e depois adicione Δw_{ij} à w_{ij}
- Para calcular os ajustes, é necessário calcular os termos de erro δ_i para cada i -ésimo neurônio.
 - Primeiro, calcula-se o termo de erro para as unidades na camada de saída;
 - Depois, essas informações são usadas para calcular os termos de erro para as unidades da camada oculta.
- Dessa forma, os erros são propagados de volta através da rede.

Algoritmo Backpropagation - Critério de Parada

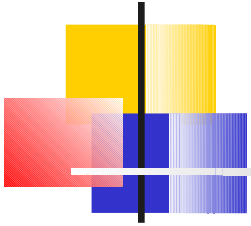


- Várias condições podem ser usadas como critério de parada.
 - Pode-se optar por parar depois de um número fixo de iterações.
 - em que uma iteração é definida pela apresentação completa do conjunto de treinamento (“época”).
 - Ou até que o erro sobre o conjunto de treinamento esteja abaixo de um determinado limiar (“threshold”).
 - Em que o erro é definido como:

$$\frac{1}{2} \sum_{E \in \text{examples}} \left(\sum_{k \in \text{outputs}} (t_k(E) - o_k(E))^2 \right)$$

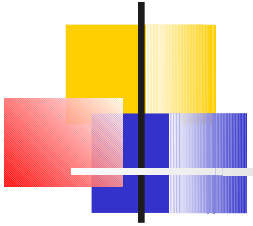
- Ou até que o erro sobre um conjunto de validação esteja abaixo de um determinado limiar (“threshold”).

Redes MLP para Tarefas de Classificação



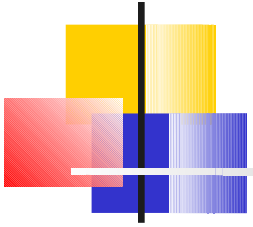
- Objetivo: treinar uma rede MLP, utilizando um conjunto de exemplos de treinamento, para classificar corretamente exemplos pertencentes a um conjunto de teste.
- Consideraremos aqui, redes MLP com apenas uma camada oculta e neurônios com função de ativação sigmóide.
- Para modelar a tarefa de classificação usando uma MLP, devem, inicialmente ser escolhidos:
 - O número de neurônios na camada oculta;
 - A maneira com que as unidades de entrada representarão os exemplos;
 - A maneira como os neurônios de saída representarão as classes estimadas.
 - O valor da taxa de aprendizagem (por exemplo, 0.1).

Redes MLP para Tarefas de Classificação



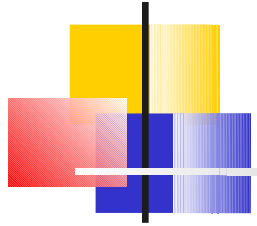
- Para treinar a rede MLP a classificar corretamente um determinado exemplo E , os seguintes passos devem ser executados.
 - Passo 1: Propagar o exemplo E (para frente) através da rede.
 - Passo 2: Calcular os termos de erro δ_i para cada neurônio da rede.
 - Passo 3: Ajustar os pesos da rede.
- O treinamento completo para um problema de classificação consiste em apresentar todo o conjunto de exemplos, várias vezes se necessário, até que o algoritmo de aprendizagem atinja o critério de parada.

Passo 1: propagando um exemplo através da rede



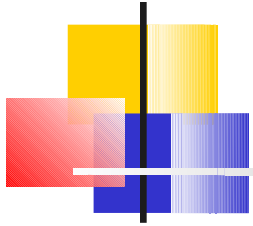
- Passo 1.1) Dado o exemplo E como entrada da rede, determine:
 - as saídas dos neurônios ocultos (que serão os sinais de entrada para as unidades de saída).
 - as saídas dos neurônios de saída (que indicam a estimativa de classe para o exemplo E).

Passo 1: propagando um exemplo através da rede



- Passo 1.2) Registre os valores desejado e obtido para o exemplo E.
 - considere $t_i(E)$ o valor desejado para a unidade de saída i , para o exemplo E.
 - considere $o_i(E)$ o valor obtido para a unidade de saída i , para o exemplo E.
- Para tarefas de classificação, por exemplo:
 - Cada $t_i(E)$ será igual a 0, exceto para um único $t_j(E)$, que será igual a 1.
 - Mas, $o_i(E)$ será, na verdade, um valor real.
- Os valores de saída dos neurônios ocultos h_i também devem ser registrados.

Passo 2: calculando os termos de erro para cada unidade



- Passo 2.1) Calcule os termos de erro para cada neurônio de saída k :

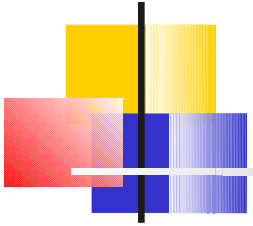
$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

- Passo 2.2) Calcule os termos de erro para cada neurônio oculto h :

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in \text{outputs}} w_{hk} \delta_k$$

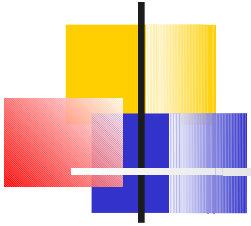
- Ou seja, para cada unidade oculta h :
 - Some todos os erros das unidades de saída que recebem sinais de h , ponderando esses valores de erro pelos pesos apropriados.
 - Multiplique o resultado dessa somatória por $o_h(1 - o_h)$.
 - Informação da primeira derivada da função de ativação.

Passo 3: ajustando os pesos da rede

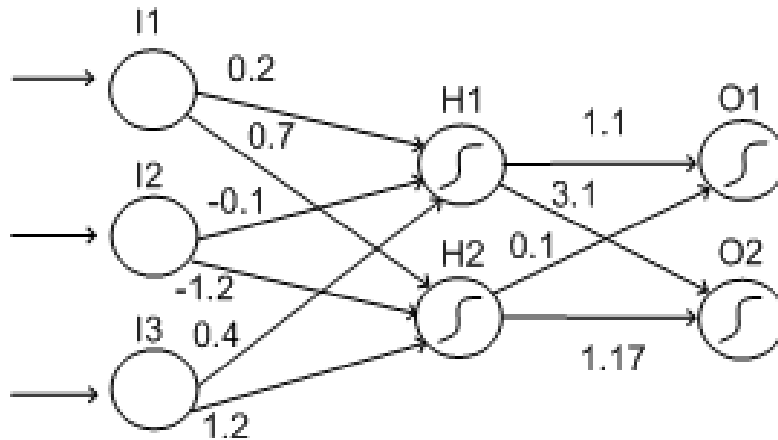


- Passo 3.1) Para cada peso de conexão w_{ij} entre uma unidade de entrada i e uma unidade oculta j :
 - Calcule: $\Delta w_{ij} = \eta \delta_j x_{ij}$
 - em que x_{ij} é a i -ésima entrada para a unidade h_j .
- Passo 3.2) Para cada peso de conexão w_{ij} entre uma unidade oculta i e uma unidade de saída j :
 - Calcule: $\Delta w_{ij} = \eta \delta_j x_{ij}$
 - em que x_{ij} , nesse caso, equivale à saída da unidade oculta h_i .
- Finalmente, some Δw_{ij} a todos os pesos w_{ij} .

Algoritmo Backpropagation - Exemplo



- Considere uma MLP com a seguinte configuração:



- Atualize os pesos dessa rede, supondo que o exemplo $E = (10, 30, 20)$ seja dado como entrada para o modelo acima.
- Considere ainda que:
 - \mathbf{e} deva ser classificado como sendo da classe 1.
 - a taxa de aprendizagem seja $\eta = 0.1$.

Algoritmo Backpropagation -

Exemplo

- A propagação do exemplo E através da rede é resumida pelos seguintes cálculos:

Input units		Hidden units			Output units		
Unit	Output	Unit	Weighted Sum Input	Output	Unit	Weighted Sum Input	Output
I1	10	H1	7	0.999	O1	1.0996	0.750
I2	30	H2	-5	0.0067	O2	3.1047	0.957
I3	20						

- Ainda:

- $t_1(E) = 1$ e $t_2(E) = 0$
- $o_1(E) = 0.750$ e $o_2(E) = 0.957$

Algoritmo Backpropagation -

Exemplo

- Dado que:

- $t_1(E) = 1$ e $t_2(E) = 0$

- $o_1(E) = 0.750$ e $o_2(E) = 0.957$

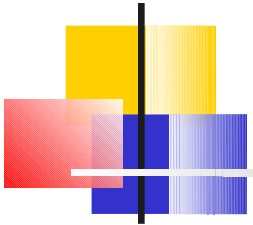
- Os termos de erro para os neurônios de saída, são calculados da seguinte forma:

$$\delta_{O1} = o_1(E)(1 - o_1(E))(t_1(E) - o_1(E)) = 0.750(1-0.750)(1-0.750) = 0.0469$$

$$\delta_{O2} = o_2(E)(1 - o_2(E))(t_2(E) - o_2(E)) = 0.957(1-0.957)(0-0.957) = -0.0394$$

Algoritmo Backpropagation -

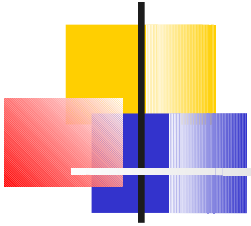
Exemplo



- Dado que:
 - $\delta_{o1} = 0.0469$ e $\delta_{o2} = -0.0394$
 - $h_1(E) = 0.999$ e $h_2(E) = 0.0067$
- Os termos de erro para os neurônios ocultos, são calculados da seguinte forma:
 - Para H1, realiza-se a somatória:
$$(w_{11} * \delta_{o1}) + (w_{12} * \delta_{o2}) = (1.1 * 0.0469) + (3.1 * -0.0394) = -0.0706$$
 - E depois multiplica-se o resultado acima por $h_1(E)(1-h_1(E))$:
$$-0.0706 * (0.999 * (1-0.999)) = -0.0000705 = \delta_{H1}$$

Algoritmo Backpropagation -

Exemplo



- Para H2, realiza-se a somatória:

$$(w_{21} * \delta_{01}) + (w_{22} * \delta_{02}) = (0.1 * 0.0469) + (1.17 * -0.0394) = -0.0414$$

- E depois multiplica-se o resultado acima por $h_2(E)(1-h_2(E))$:

$$-0.0414 * (0.067 * (1-0.067)) = -0.00259 = \delta_{H2}$$

Algoritmo Backpropagation -

Exemplo

- Os cálculos das mudanças de pesos para as conexões entre a camada de entrada e a camada oculta estão resumidos na tabela:

Input unit	Hidden unit	η	δ_H	x_i	$\Delta = \eta * \delta_H * x_i$	Old weight	New weight
I1	H1	0.1	-0.0000705	10	-0.0000705	0.2	0.1999295
I1	H2	0.1	-0.00259	10	-0.00259	0.7	0.69741
I2	H1	0.1	-0.0000705	30	-0.0002115	-0.1	-0.1002115
I2	H2	0.1	-0.00259	30	-0.00777	-1.2	-1.20777
I3	H1	0.1	-0.0000705	20	-0.000141	0.4	0.39999
I3	H2	0.1	-0.00259	20	-0.00518	1.2	1.1948

Algoritmo Backpropagation -

Exemplo

- Os cálculos das mudanças de pesos para as conexões entre a camada oculta e a camada de saída estão resumidos na tabela:

Hidden unit	Output unit	η	δ_O	$h_i(E)$	$\Delta = \eta * \delta_O * h_i(E)$	Old weight	New weight
H1	O1	0.1	0.0469	0.999	0.000469	1.1	1.100469
H1	O2	0.1	-0.0394	0.999	-0.00394	3.1	3.0961
H2	O1	0.1	0.0469	0.0067	0.00314	0.1	0.10314
H2	O2	0.1	-0.0394	0.0067	-0.0000264	1.17	1.16998



Leituras

- MITCHELL, T. Machine Learning. McGraw Hill, 1997
 - Capítulo 4: Artificial Neural Networks.