Técnicas de Mineração de Dados
Redes Neurais

Prof. Leandro Almeida

lma3@cin.ufpe.br

Resumo

- Objetivo:
  - Conceituar e conhecer as técnicas de Redes Neurais Artificiais.
- Conteúdo:
  - Redes Neurais Artificiais
- Referências:
  - Capítulos 1 a 4 do livro Braga, A. P.; Ponce de Leon, A. C. e Ludermir, T.B. Redes Neurais Artificiais: Teoria e Aplicações

O que é uma Rede Neural?

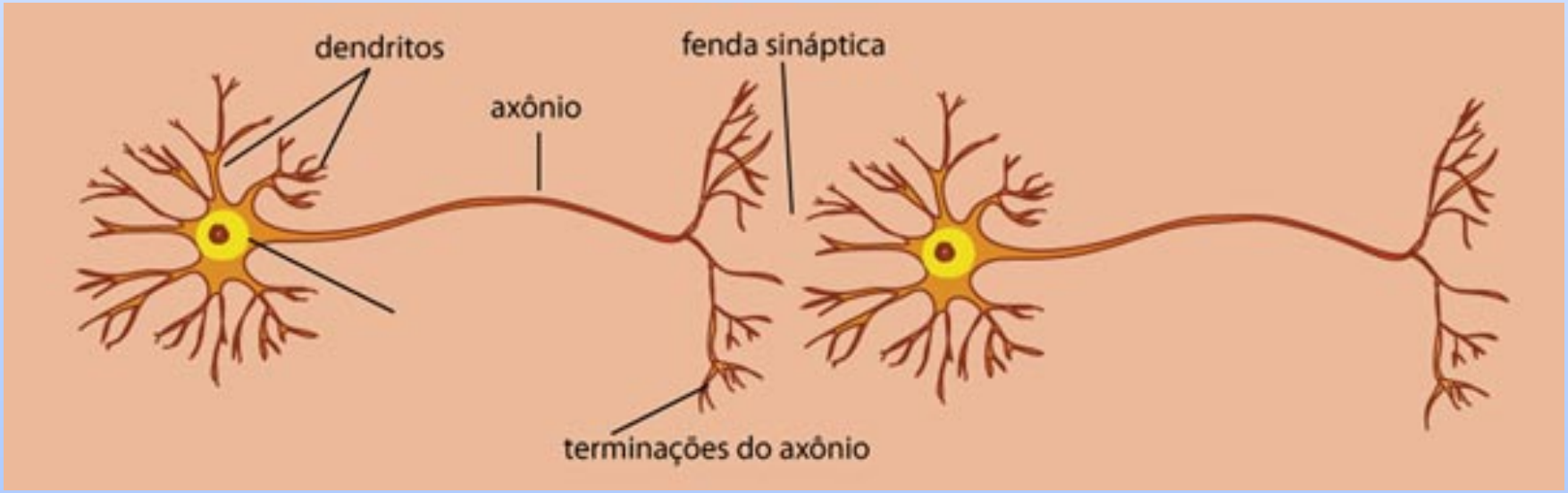
- O trabalho em Redes Neurais Artificiais (RNA), tem sido motivado pelo fato de que o cérebro humano processa informações de uma forma inteiramente diferente do computador digital convencional.
- O cérebro é um sistema de processamento de informação altamente **Complexo**, **Não-Linear** e **Paralelo**.
- Considere uma tarefa de processamento que é realizada corriqueiramente pelo cérebro: a visão humana.
  - O reconhecimento perceptivo (exemplo, reconhecer um rosto familiar em uma cena não-familiar) pode ser realizado pelo cérebro em poucos milésimos de segundo.

O que é uma Rede Neural?

- Como o **cérebro** é capaz de realizar o reconhecimento perceptivo, e outras tantas tarefas complexas, em um intervalo tão curto de tempo, ao passo que tarefas de complexidade muito menor podem levar dias para serem executadas em um **computador convencional**?
- No momento do nascimento, o cérebro de uma criança tem uma **grande estrutura** e a **habilidade** de desenvolver suas próprias regras através do que usualmente denominamos “experiência”.
- Na sua forma geral, uma RNA é uma máquina projetada para **modelar/simular** a maneira como o cérebro realiza uma tarefa particular ou uma função de interesse.

Inspiração Biológica

- Sinais eletroquímicos
- Limiar de disparo



Inspiração Biológica

- **Axônios** – linhas de transmissão.
- **Dendritos** – zonas receptivas
- Os neurônios ficam contidos num ambiente líquido contendo uma certa concentração de íons, que podem entrar ou sair através dos **canais iônicos**.
  - Tanto as transmissões de sinais nos axônios, como as sinapses usam esses canais iônicos.
  - Os canais iônicos podem ser modulados, permitindo ao cérebro se **adaptar** a diferentes situações.
- A **plasticidade** sináptica é a capacidade das sinapses sofrerem modificações.

Inspiração Biológica

- Numa sinapse, dependendo da carga do íon, o fluxo resulta em aumentar (**excitação**) ou diminuir (**inibição**) o potencial de membrana.
- O dendrito de um neurônio recebe íons de várias sinapses e o resultado elétrico da concentração desses íons consiste no que se chama de **potencial de membrana**.
  - O potencial de membrana gera eventualmente um pulso elétrico de disparo, denominado **potencial de ação**.
- A ativação de um neurônio ocorre apenas quando seu potencial de membrana é maior do que um valor limiar (**threshold**).
- O potencial de ação é gerado no corpo celular e percorre o axônio até a sua extremidade, que coincide com a sinapse, para atuar no neurônio pós-sinaptico seguinte.

Inspiração Biológica

- A aprendizagem é resultado de alterações locais nos neurônios.
- Fisicamente, existem diversas formas de modificações possíveis em um neurônio:
  a) dendritos podem **nascer**, bem como serem **removidos**;
  b) alguns dendritos se **esticam** ou se **encolhem**, permitindo ou eliminando, respectivamente, a conexão com outras células;
  c) novas sinapses podem ser **criadas** ou sofrer **alterações**;
  d) sinapses também podem ser **removidas**; e
  e) todo neurônio pode **morrer** e também se **regenerar**.
- A **aprendizagem** via modulação sináptica é o mecanismo mais importante para as redes neurais, sejam elas biológicas ou artificiais.

Inspiração Biológica

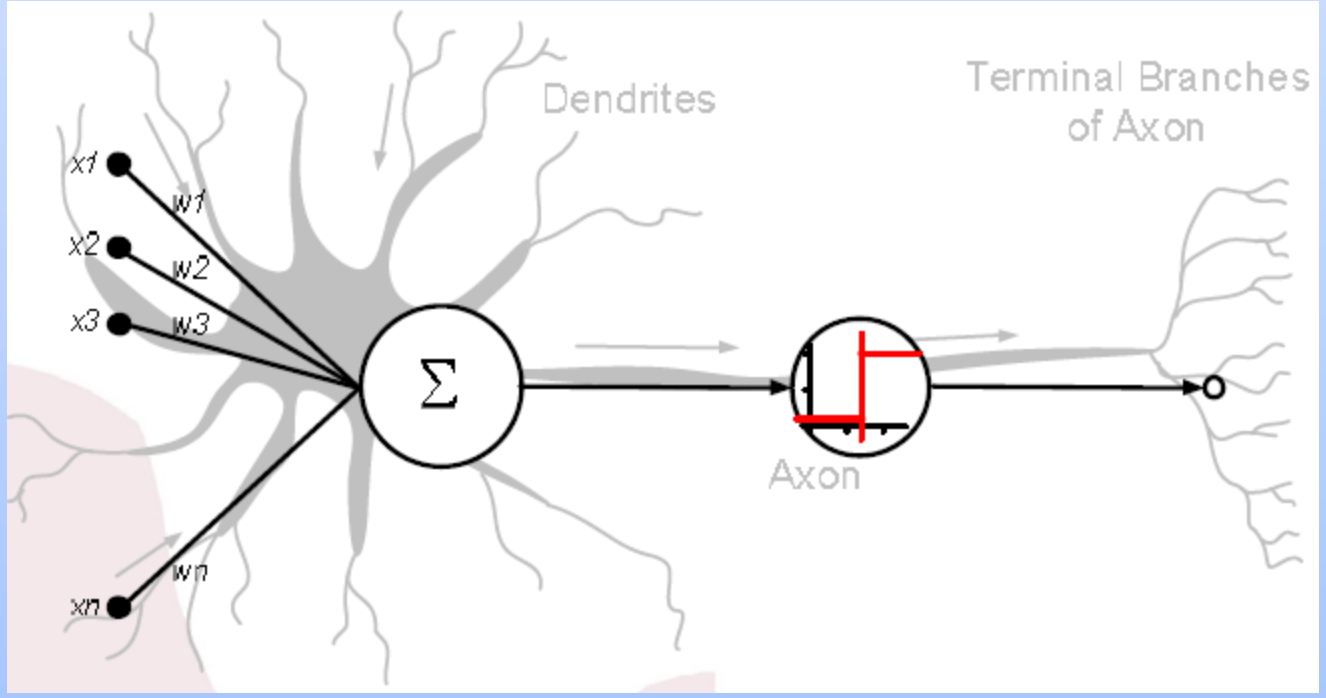
- A **memória** também é resultado de um processo adaptativo das sinapses.

- Um dos resultados de um **processo de aprendizagem** é a criação de um **padrão de conexões sinápticas** duradouro, que resulta na memorização de uma determinada experiência.

- A aprendizagem pode ser vista como o **processo adaptativo** da estrutura sináptica, enquanto a memória é o resultado deste processo adaptativo.

Perceptron

- Funções de classificação binária
- Função de ativação



Motivações

- No processo de aprendizado através de exemplos, as redes neurais artificiais exibem uma outra característica muito interessante: **GENERALIZAÇÃO**

- Isso significa que se a rede **aprende** a lidar com um certo problema, e lhe é apresentado um **similar**, mas não exatamente o mesmo, ela tende a **reconhecer** esse novo problema, oferecendo **solução semelhante**.

- De forma análoga, os seres humanos tendem a aplicar os conhecimentos anteriores para lidar com **novos problemas**.

Motivações

- Alguns Benefícios das Redes Neurais Artificiais
  - **Adaptabilidade** por intermédio de aprendizado.
  - Capacidade de operar com **conhecimento parcial**.
  - **Tolerância** a falhas.
  - **Generalização**.
  - Informação **contextual**.
  - **Mapeamento** entrada-saída.

Processo de Aprendizagem

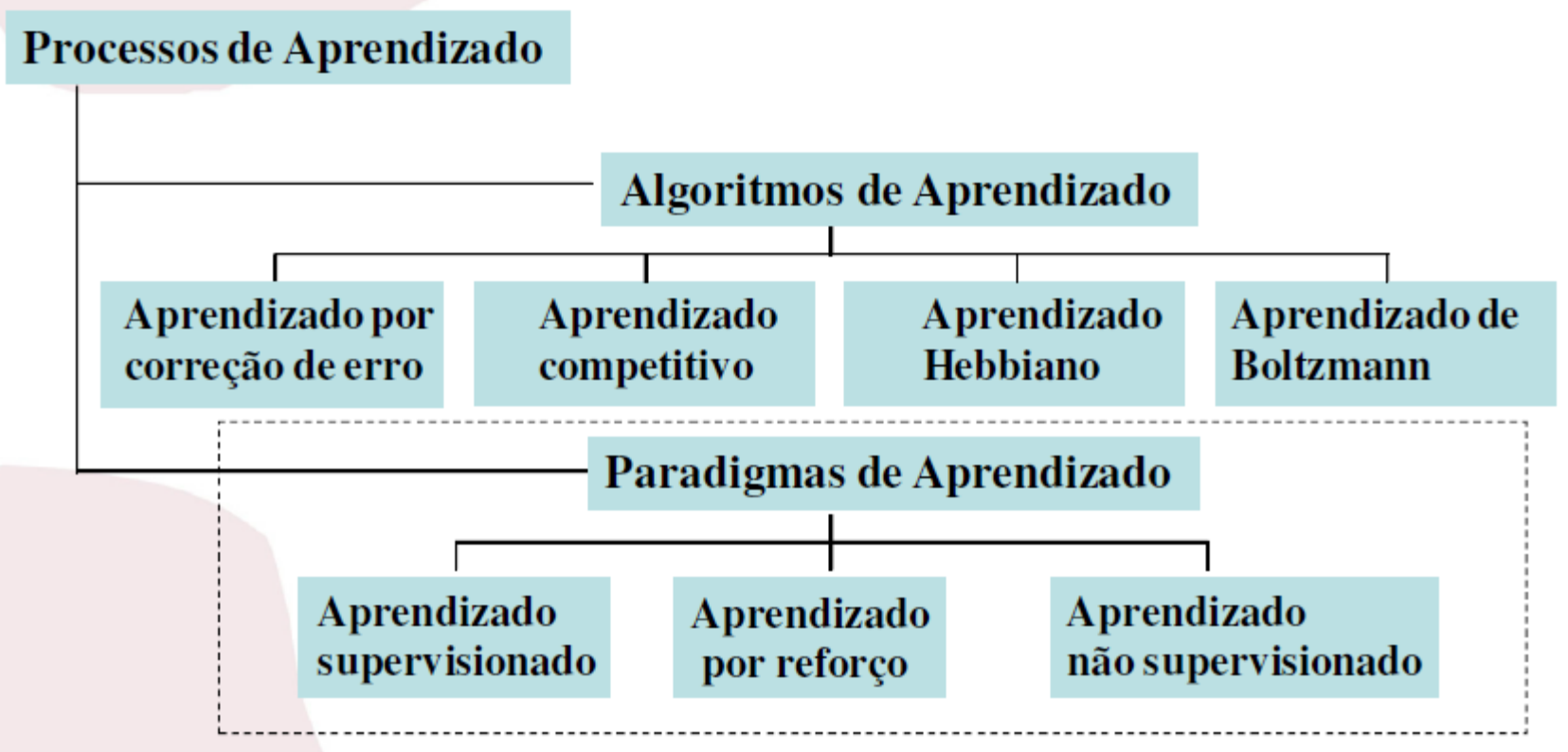
- Uma rede neural artificial pode se encontrar em duas fases:
  - A primeira fase é a de **aprendizagem**, ou **treinamento**, em que a rede se encontra no processo de aprendizado, **ajustando os parâmetros livres**, para poder posteriormente desempenhar a função destinada; e
  - A segunda fase é a de **aplicação** propriamente dita, na função para a qual ela foi destinada, como de classificação de padrões de vozes, imagens, etc.

Processo de Aprendizagem

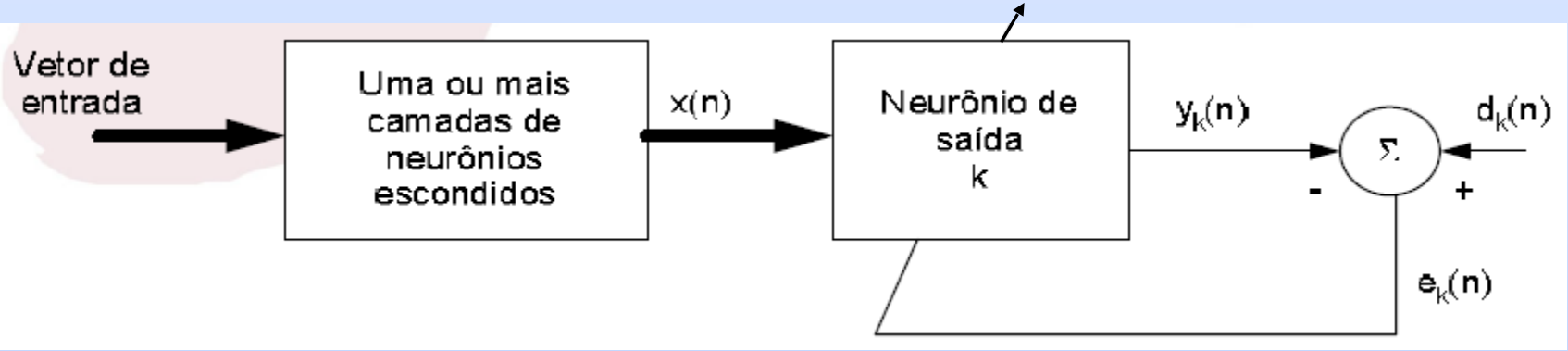
- O processo de aprendizagem implica na seguinte sequência de eventos:
  1. A rede neural é **estimulada** por um ambiente;
  2. A rede neural sofre **modificações** nos seus parâmetros livres como resultado desta estimulação;
  3. A rede neural **responde de uma maneira nova** ao ambiente, devido as modificações ocorridas na sua estrutura interna.
- O **problema de aprendizagem** é solucionado por um **conjunto pré-estabelecido de regras** – o algoritmo de aprendizagem

- Outro fator a ser considerado na solução do problema de aprendizagem é a maneira pela qual uma rede neural se relaciona com seu ambiente – **o paradigma (modelo) de aprendizagem**

Aprendizagem de Máquina



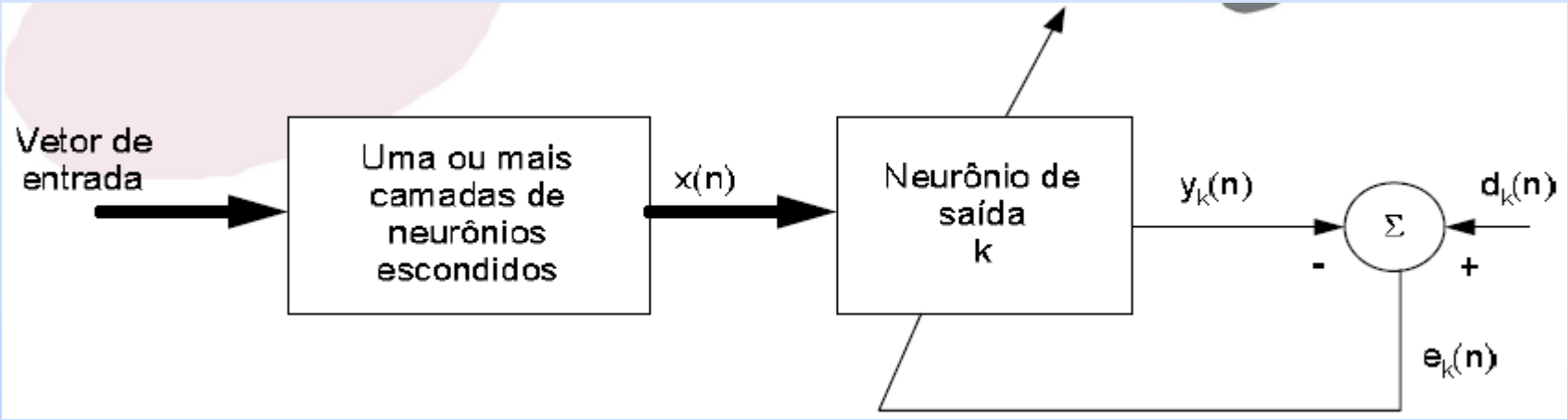
Aprendizagem por Correção de Erro



- O **sinal de saída** do neurônio **k** é representado por $y_k(n)$, e a **resposta desejada** por $d_k(n)$, produzindo um **sinal de erro**:
  - $e_k(n) = d_k(n) - y_k(n)$
- O sinal de erro $e_k(n)$ aciona um mecanismo de controle, cujo propósito é aplicar uma **sequência de ajustes** corretivos aos **pesos sinápticos** do neurônio **k**. Os ajustes corretivos são projetados para **aproximar**, passo a passo, o sinal de saída $y_k(n)$ da resposta desejada $d_k(n)$.
- Este objetivo é alcançado **minimizando-se** uma **função de custo** ou índice de desempenho, **E(n)**, definido em termos do sinal de erro como:

$$E(n) = \tfrac{1}{2} e_k^2(n)$$

Aprendizagem por Correção de Erro



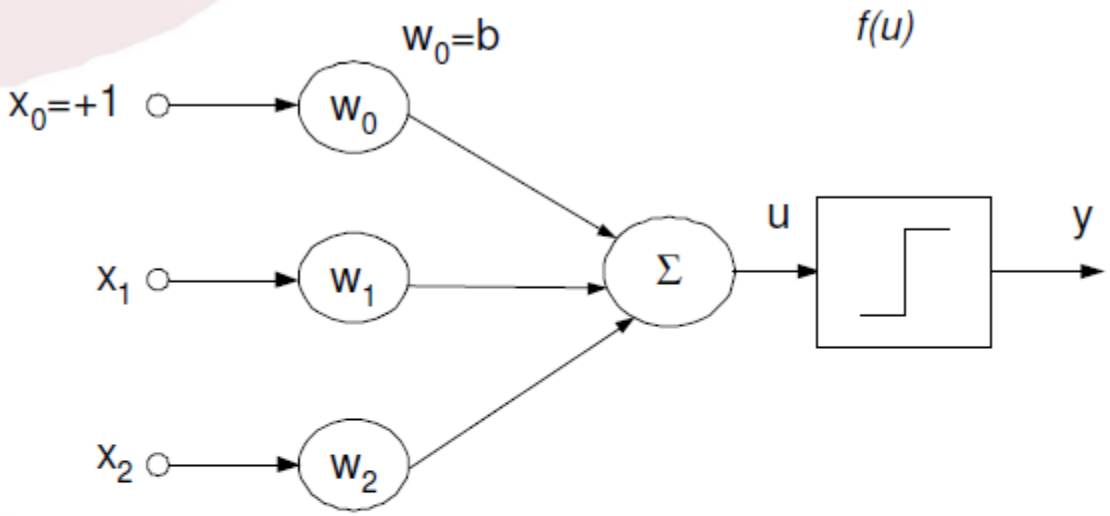
- Nota-se que o sinal de erro deve ser diretamente mensurável, ou seja, a **resposta desejada deve ser fornecida por alguma fonte externa**, e o neurônio k deve ser visível ao mundo externo.

- Tendo calculado o ajuste sináptico, o **valor atualizado** do peso sináptico é determinado por:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n).$$

O Perceptron

- **Perceptron de duas entradas e um bias**



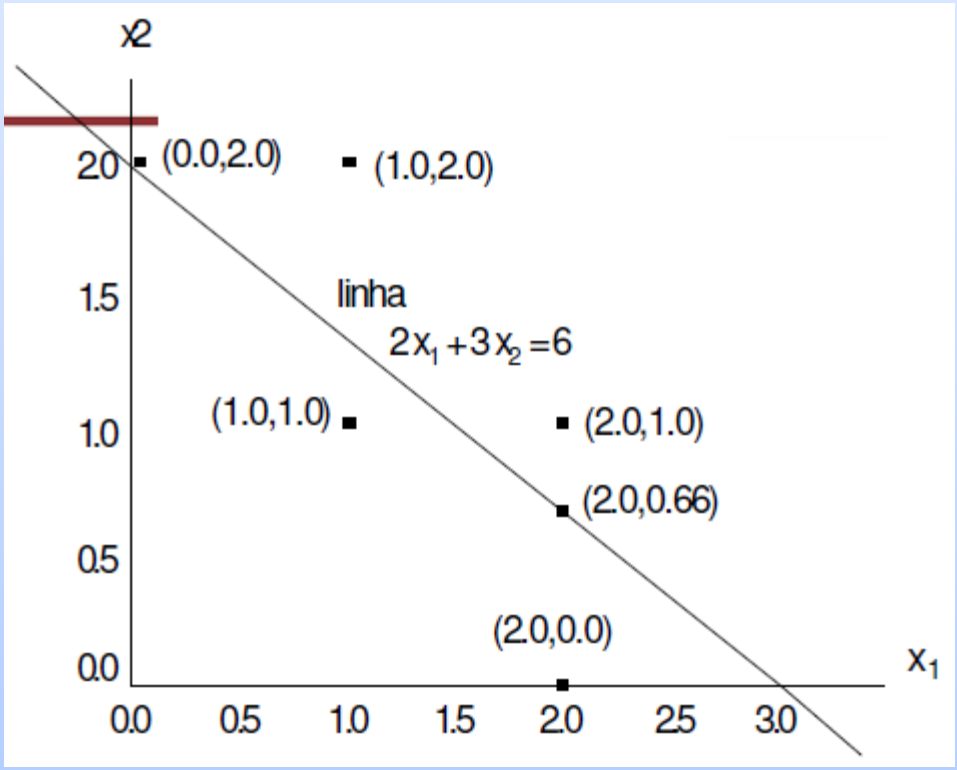
$$y = f(w_1 x_1 + w_2 x_2 + w_0), \quad \text{sendo} \quad \begin{cases} f(u) = 1 & \text{se } u \geq 0 \\ f(u) = 0 & \text{se } u < 0 \end{cases}$$

Com os parâmetros w_0, w_1 e w_2, a função *f(u)* separa o espaço de entradas em **duas regiões**, usando uma linha reta dada por:

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

Exemplo

Pontos (x_1, x_2)	$2x_1+3x_2$	posição
(0.0, 2.0)	6	linha
(1.0, 1.0)	5	abaixo
(1.0, 2.0)	8	acima
(2.0, 0.0)	4	abaixo
(2.0, 0.66)	6	linha
(2.0, 1.0)	7	acima



Posição dos pontos em função da linha $2 x_1 + 3 x_2 = 6$ de delimitação

Linha: $2 x_1 + 3 x_2 = 6$

Acima: $2 x_1 + 3 x_2 > 6$

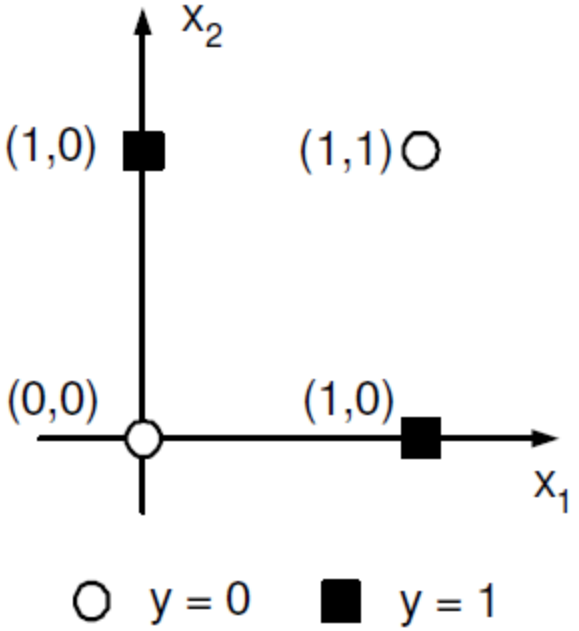
Abaixo: $2 x_1 + 3 x_2 < 6$

$\theta = 6$

Problema do XOR (ou-exclusivo)

No caso do XOR, não existe uma única reta que divide os pontos (0,0) e (1,1) para um lado, e (0,1) e (1,0) do outro lado.

Função XOR



Conclui-se que um **neurônio do tipo Perceptron** não implementa uma **função ou-exclusivo** (constatado por Minsky & Papert, em 1969).

Introdução

- Não linearidades são inerentes à maioria das situações e problemas reais.
- Não linearidades são incorporadas através:
  - De funções de ativação não lineares.
  - Da composição de sucessivas camadas de neurônios.
- MLP (*MultiLayer Perceptron*):
  - RNA composta por neurônios com funções de ativação sigmoidais nas camadas intermediárias.

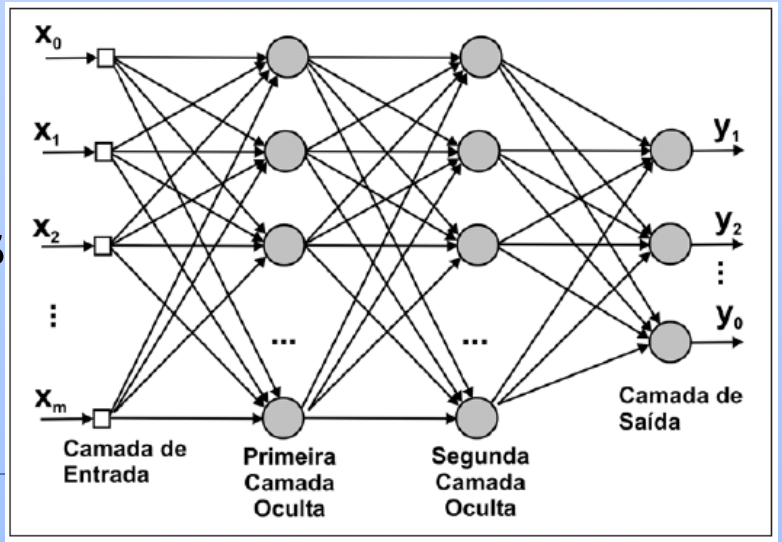


Figura 2 - Rede multilayer feedforward.

Introdução

- Perceptron:
  - Aprendizado supervisionado e correção de erros
  - Ajustes no vetor de pesos
  - Saída desejada -> saída obtida

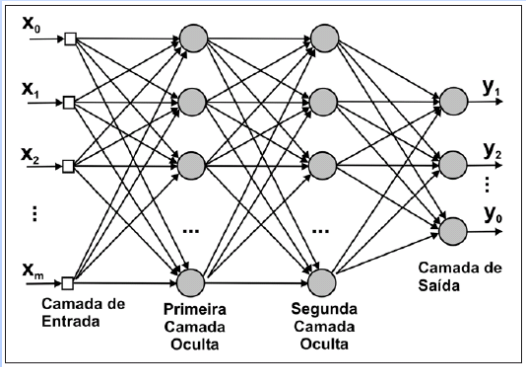


Figura 2 - Rede multilayer feedforward.

- MLP:
  - Aplicado somente à última camada.
  - Não há saídas desejadas para camadas intermediárias.
  - Como calcular ou estimar o erro das camadas intermediárias?

Introdução

- Algoritmo Back-propagation
  - Década de 80. Novo “gás” para área de redes neurais.

- Gradiente descendente

  - Estimação do erro das camadas intermediárias pelo **efeito** que estas causam no erro da camada de saída.
  - Erro da camada de saída **retroalimentado** para camadas intermediárias.
  - Ajustes dos pesos **proporcional** aos valores das conexões entre as camadas.

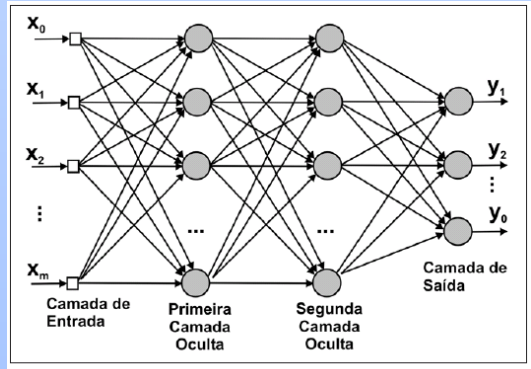


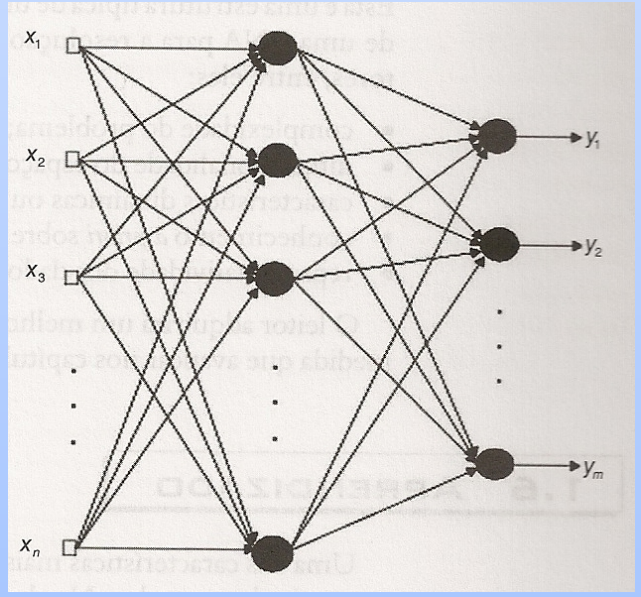
Figura 2 - Rede multilayer feedforward.

Arquitetura

- Redes com duas camadas podem implementar qualquer função, seja ela linearmente separável ou não [Cybenko, 1989].

- A qualidade da aproximação obtida depende da complexidade da rede.

- Número de camadas, número de neurônios, funções de ativação

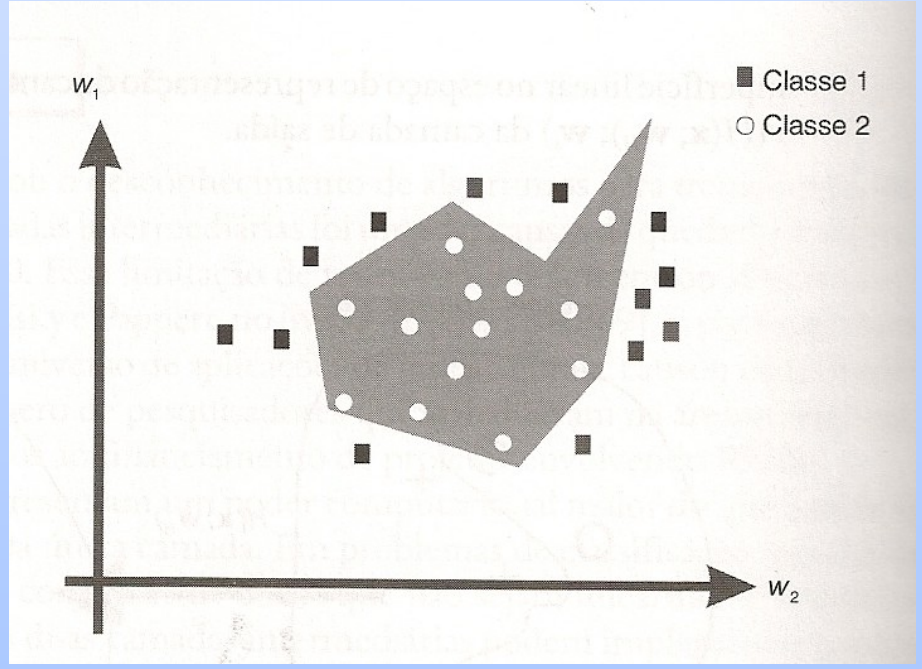


Arquitetura

- Número de camadas
  - Maioria dos problemas práticos raramente precisam mais que duas camadas.
  - Primeira camada: cada neurônio contribui com retas para formação da superfície no espaço de entrada.
  - Segunda camada: cada neurônio combina as retas formando regiões convexas

Arquitetura

- Número de camadas



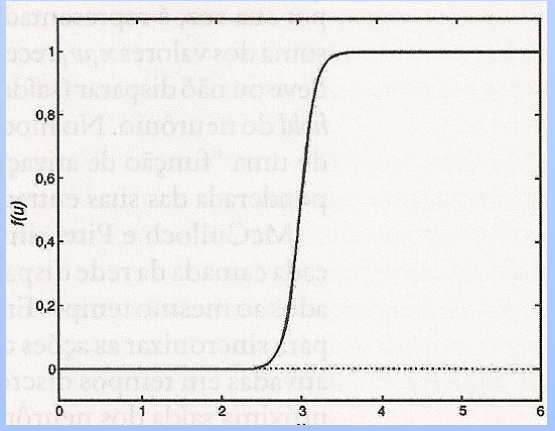
Arquitetura

- Número de neurônios
  - Refere-se a capacidade de generalização da rede.
  - Quanto maior o número de neurônios, maior a capacidade de resolver problemas.
  - Não há na literatura definição formal acerca da quantidade de neurônios.
  - Empirismo: adiciona-se ou reduz-se de acordo com a medida de tolerância da rede.

Arquitetura

- Funções de ativação
  - Sigmoidais nas camadas intermediárias e Lineares na camada de saída.
  - Semelhante a degrau, contudo possui região semilinear, que pode ser importante na aproximação de funções contínuas.

$$f(u) = \frac{1}{1 + e^{-\beta u}}$$

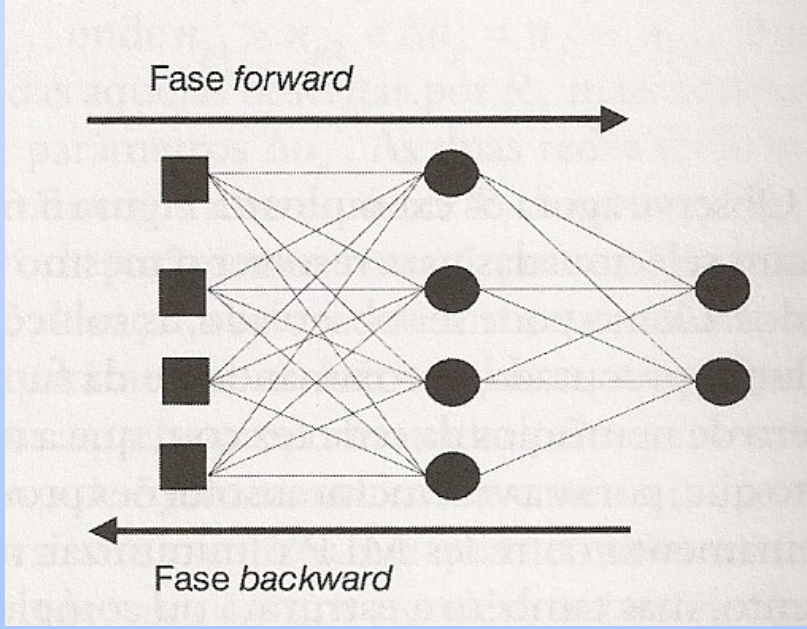


Aprendizado

- Uma RNA é composta por:
  - Um conjunto de neurônios com capacidade de processamento.
  - Uma topologia de conexão que defina como estes neurônios estão conectados.
  - Uma regra de aprendizado.
- Redes MLP:
  - Diversos neurônios
  - Topologia de duas ou mais camada
  - Regra Delta Generalizada

Aprendizado

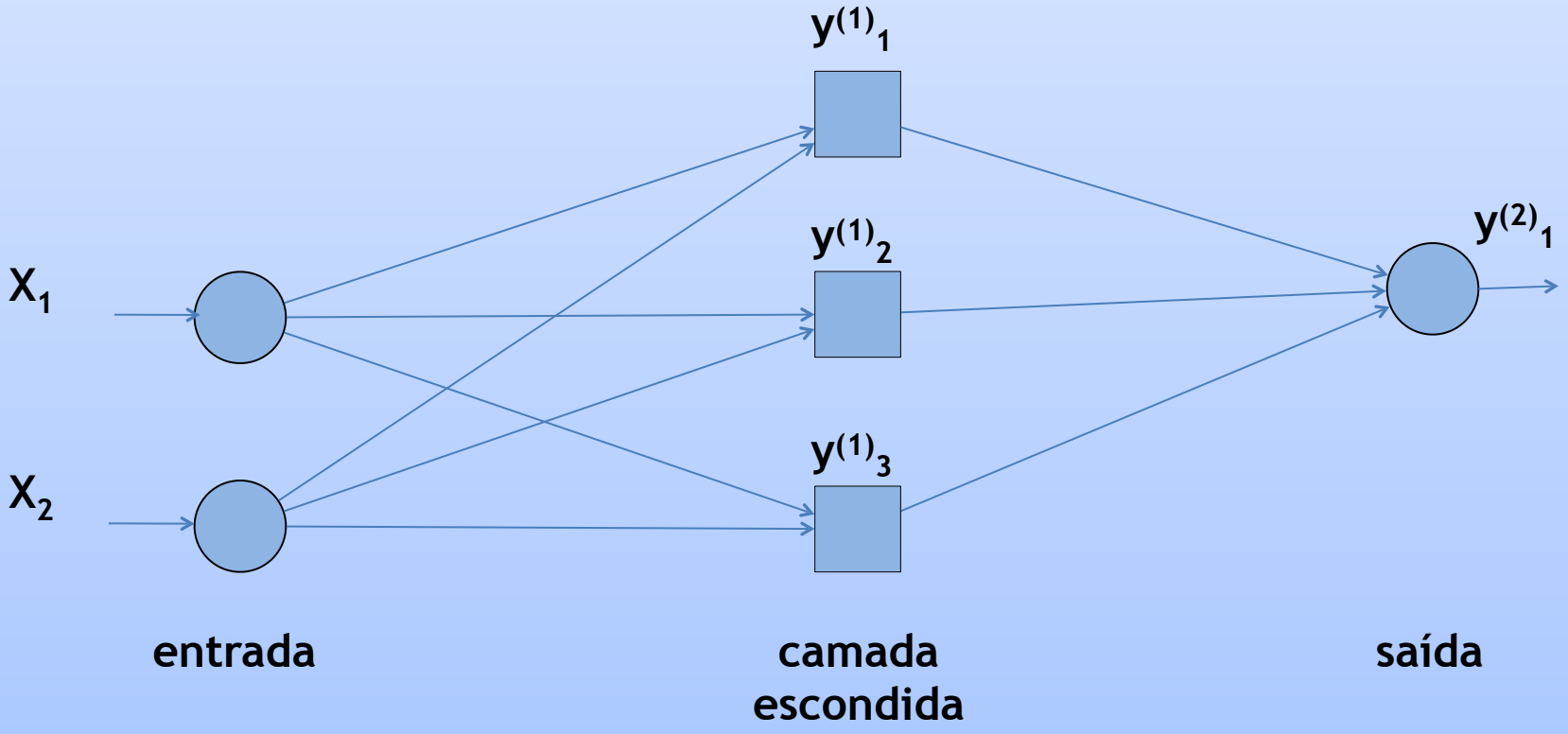
- Treinamento em duas fases:



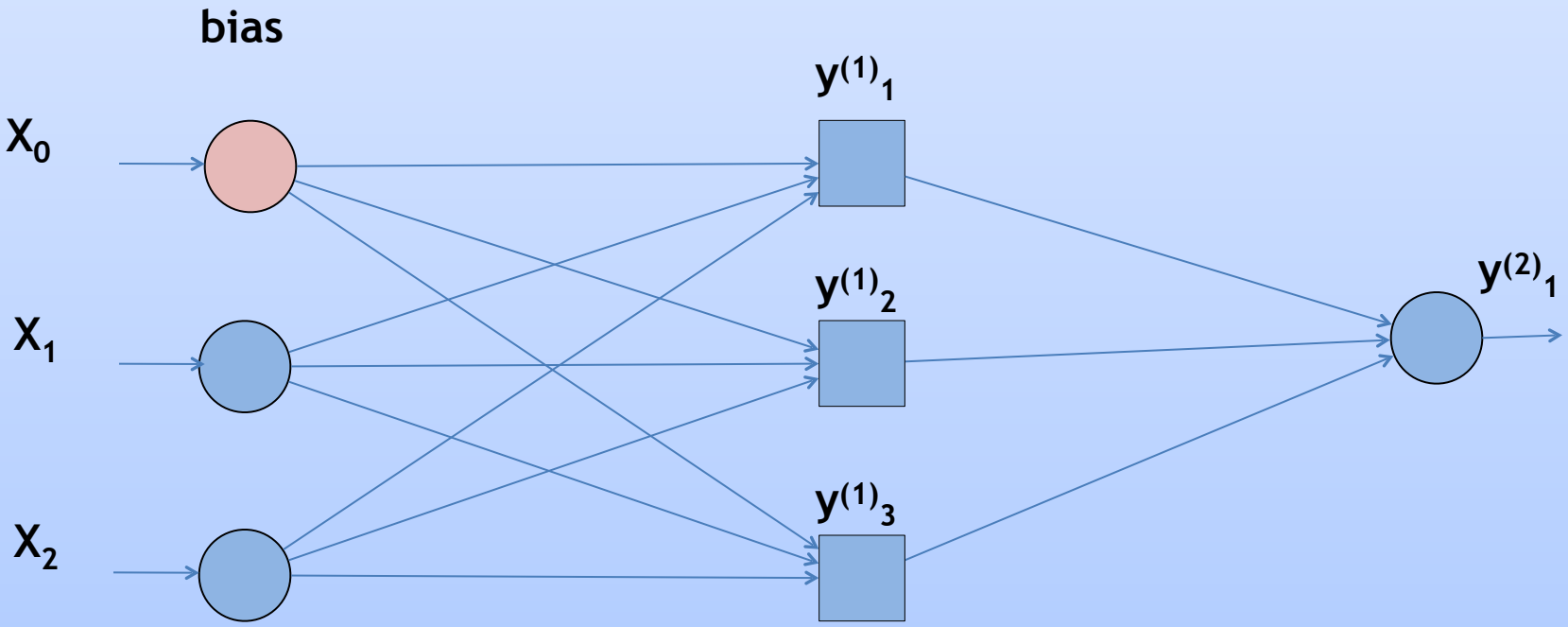
Implementação do Algoritmo

- ***Fase Forward***
  - Inicializar η;
  - Inicializar a matriz de pesos w com valores aleatórios;
  - Apresentar entrada 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

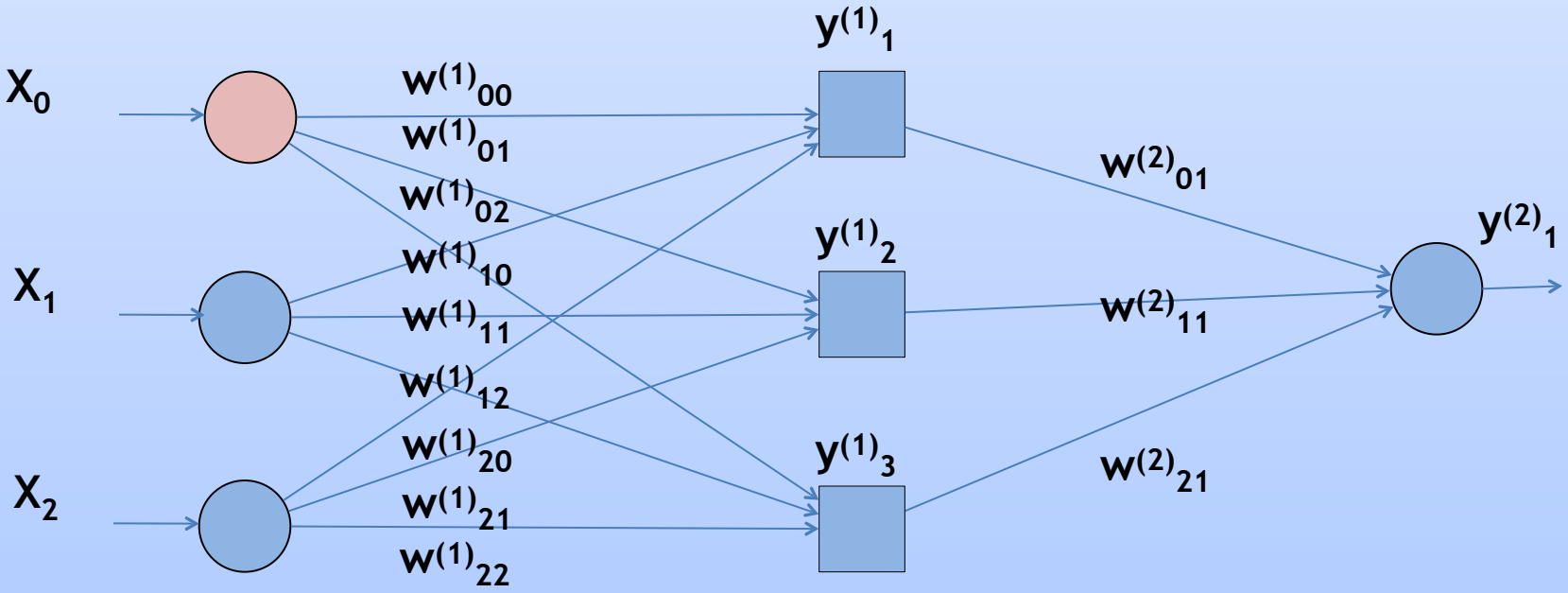
Treinamento



Treinamento



Treinamento

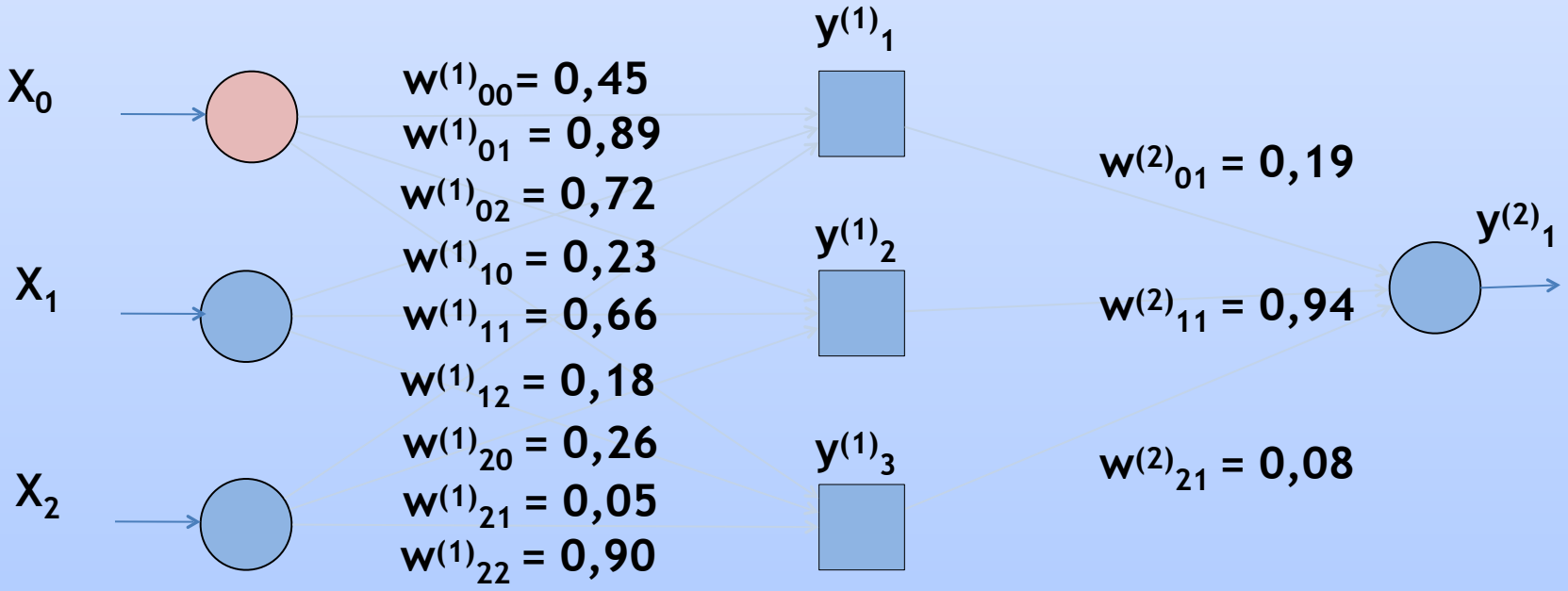


pesos

Treinamento

$\eta = 0,1$

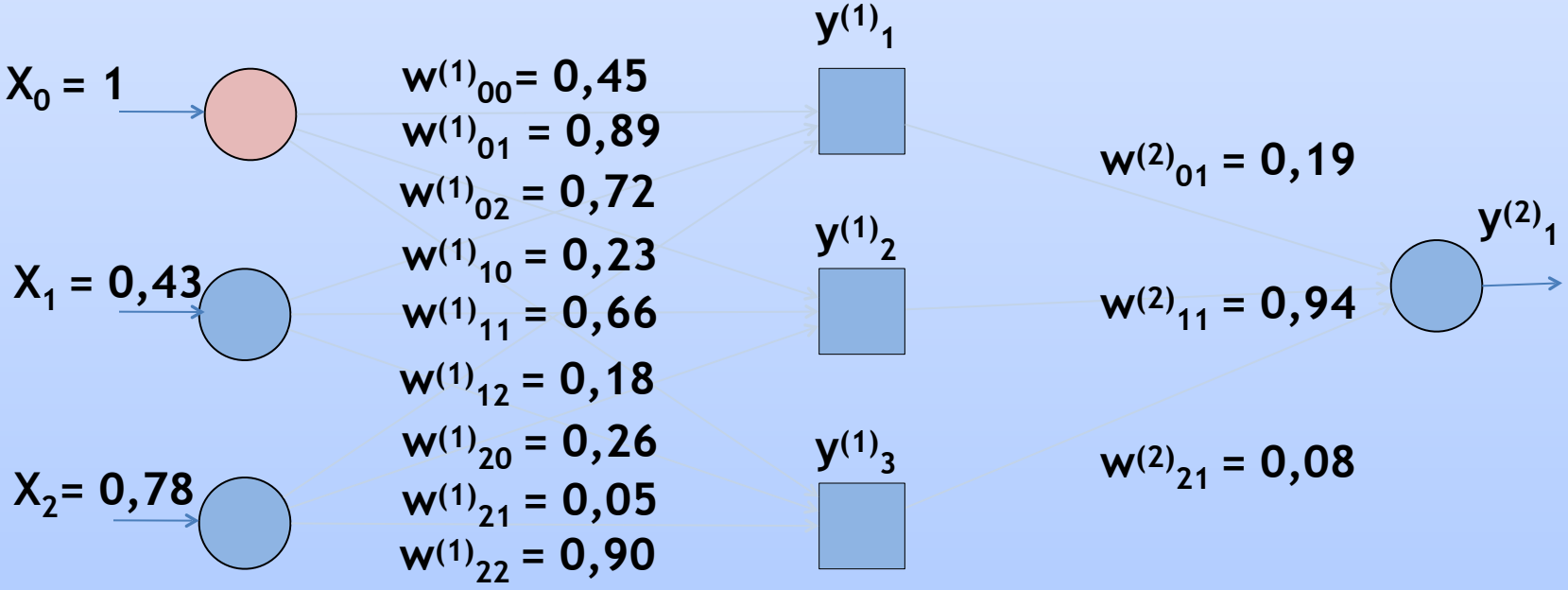
1. Inicia todas as conexões com pesos aleatórios



Treinamento

$\eta = 0,1$
$d = 1$

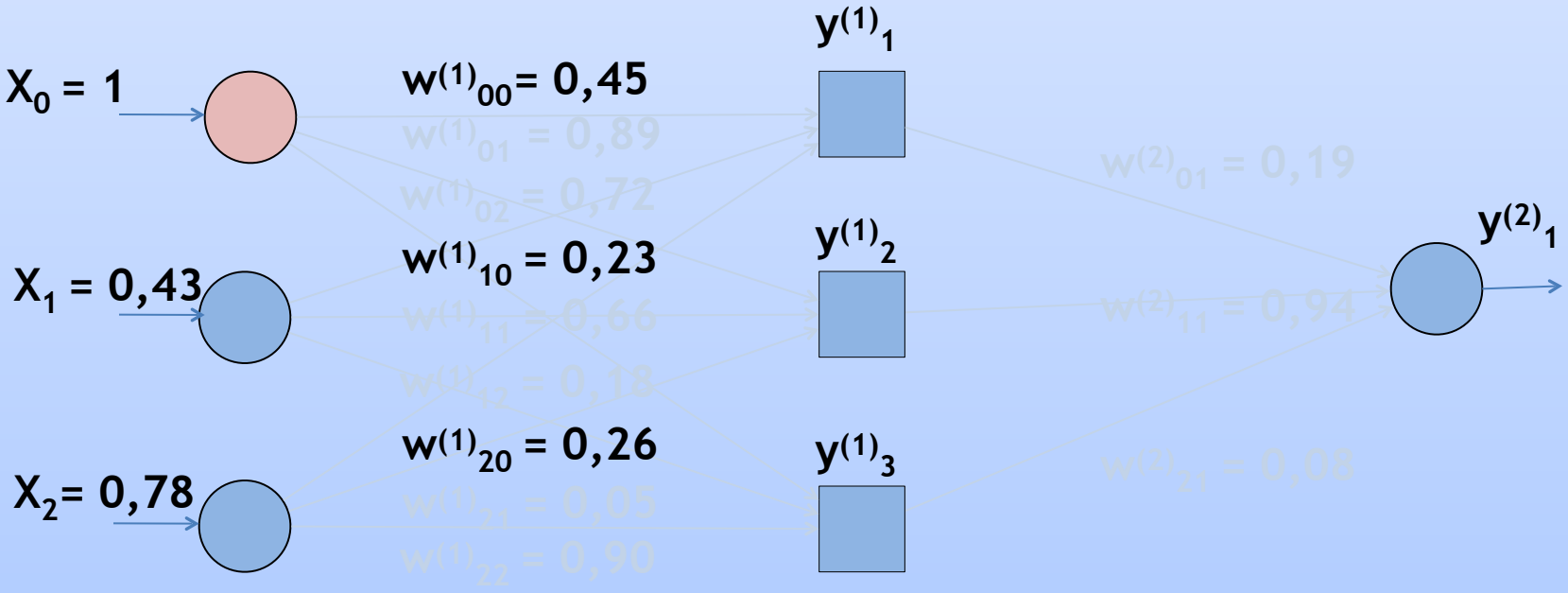
2. Para de entrada X é apresentado



Treinamento

$\eta = 0,1$
$d = 1$

3. Calcula saída para 1ª camada

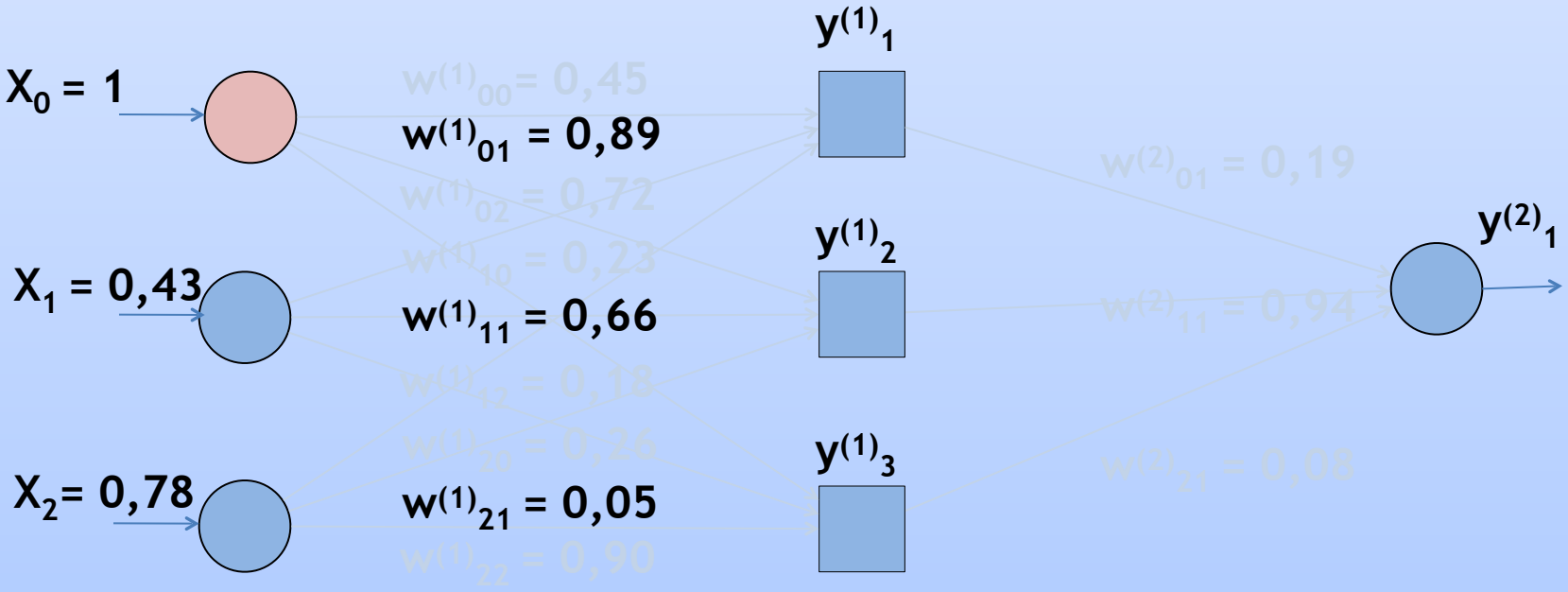


$u^{(1)}_1 = (1*0,45) + (0,43*0,23) + (0,78*0,26) = 0,7517$

Treinamento

$\eta = 0,1$
$d = 1$

3. Calcula saída para 1ª camada



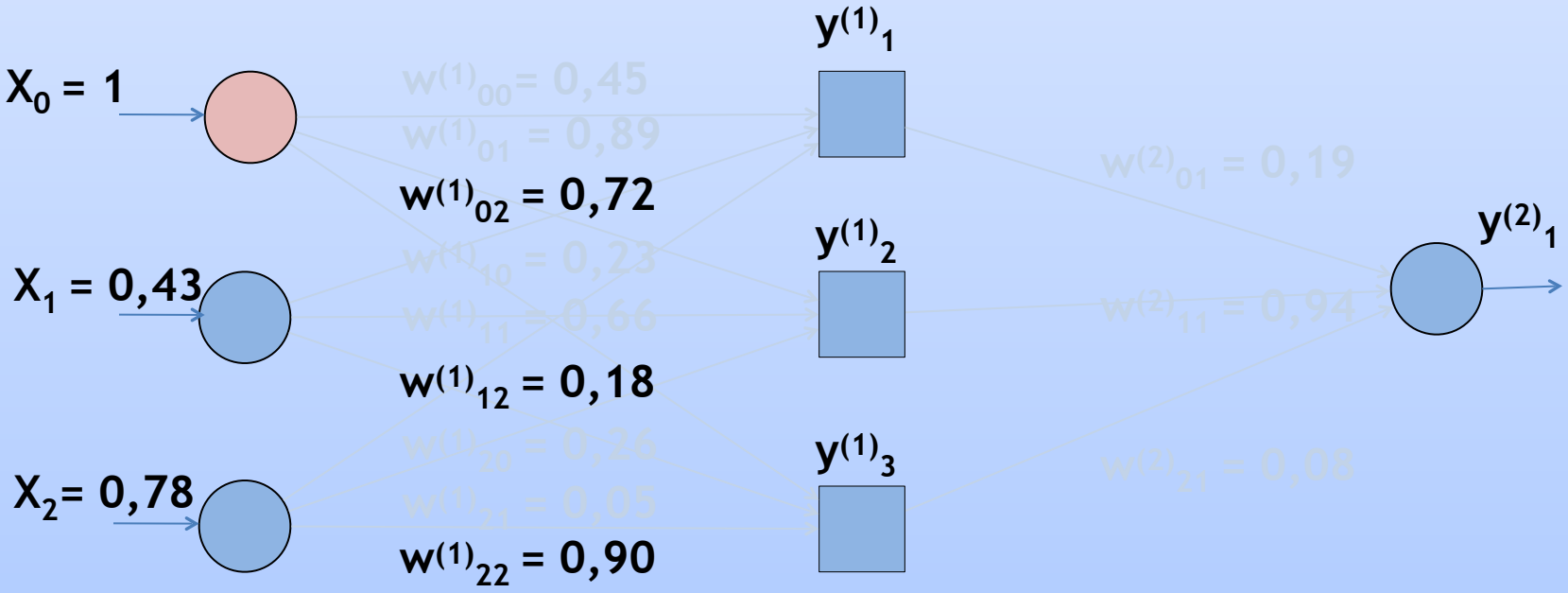
$u^{(1)}_1 = (1*0,45) + (0,43*0,23) + (0,78*0,26) = 0,7517$

$u^{(1)}_2 = (1*0,89) + (0,43*0,66) + (0,78*0,05) = 1,2128$

Treinamento

$\eta = 0,1$
$d = 1$

3. Calcula saída para 1ª camada



$u^{(1)}_1 = (1*0,45) + (0,43*0,23) + (0,78*0,26) = 0,7517$

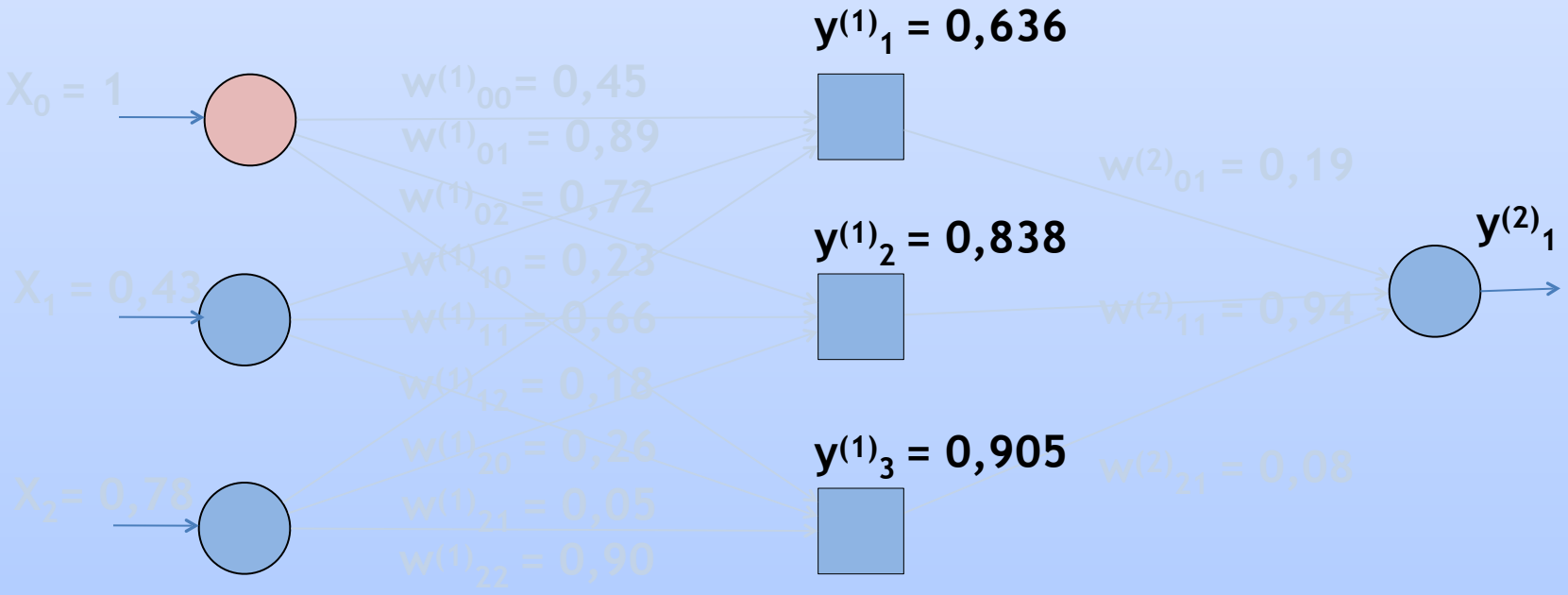
$u^{(1)}_2 = (1*0,89) + (0,43*0,66) + (0,78*0,05) = 1,2128$

$u^{(1)}_3 = (1*0,72) + (0,43*0,18) + (0,78*0,90) = 1,4994$

Treinamento

$\eta = 0,1$
$d = 1$

4. Calcula saída da função de ativação



$u^{(1)}_1 = (1*0,45) + (0,43*0,23) + (0,78*0,26) = 0,7517$

$u^{(1)}_2 = (1*0,89) + (0,43*0,66) + (0,78*0,05) = 1,2128$

$u^{(1)}_3 = (1*0,72) + (0,43*0,18) + (0,78*0,90) = 1,4994$

$y^{(1)}_1 = TANH(u^{(1)}_1) = 0,636$

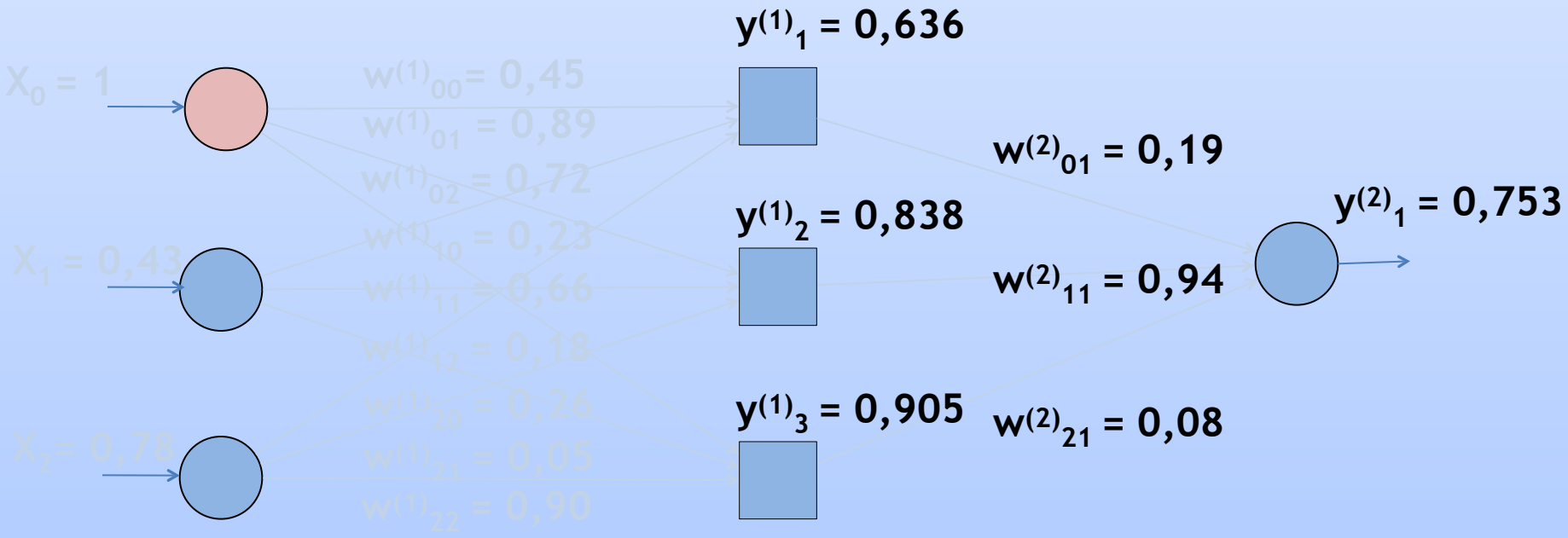
$y^{(1)}_2 = TANH(u^{(1)}_1) = 0,838$

$y^{(1)}_3 = TANH(u^{(1)}_1) = 0,905$

Treinamento

$\eta = 0,1$
$d = 1$

5. Calcula saída para 2ª camada



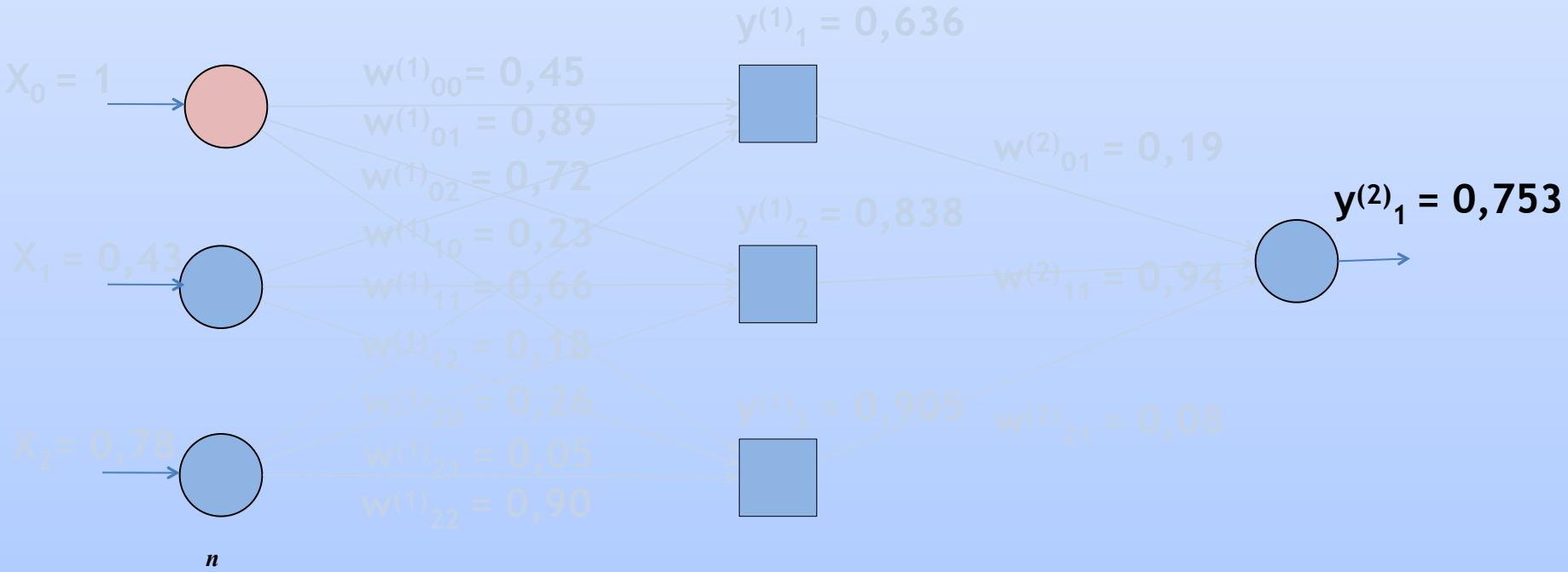
$u^{(2)}_1 = (0,636*0,19) + (0,838*0,94) + (0,905*0,08) = 0,981$

$y^{(2)}_1 = TANH(u^{(2)}_1) = 0,753$

Treinamento

$\eta = 0,1$
$d = 1$

6. Calcula variação do erro



$$E(k) = 1/2 \sum_{i=0}^{n} (d_i(t) - y_i(t))^2$$

$$E(k) = ½ (0,247)^2 = 0,03$$

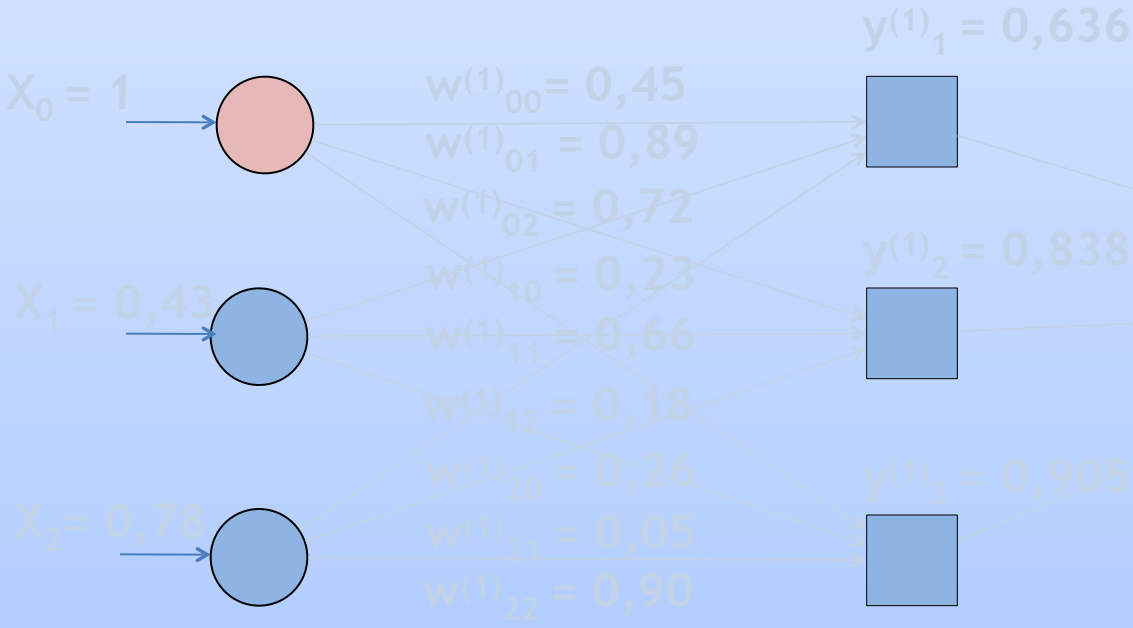
Implementação do Algoritmo

- ***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

Treinamento

$\eta = 0,1$
$d = 1$

7. Calcula variação dos pesos da 2ª camada



$w^{(2)}_{01} = 0,19$

$w^{(2)}_{11} = 0,94$

$w^{(2)}_{21} = 0,08$

$y^{(2)}_1 = 0,753$

$w^{(2)}(t+1) = w^{(2)}(t) + \eta\, \delta^2\, y^{(1)}(t)$

$\delta^{(2)}(t) = (d(t) - y(t)) * y'^{(2)}_1$

$\delta^{(2)}(t) = 0,247 * \text{ATANH}(0,753)$

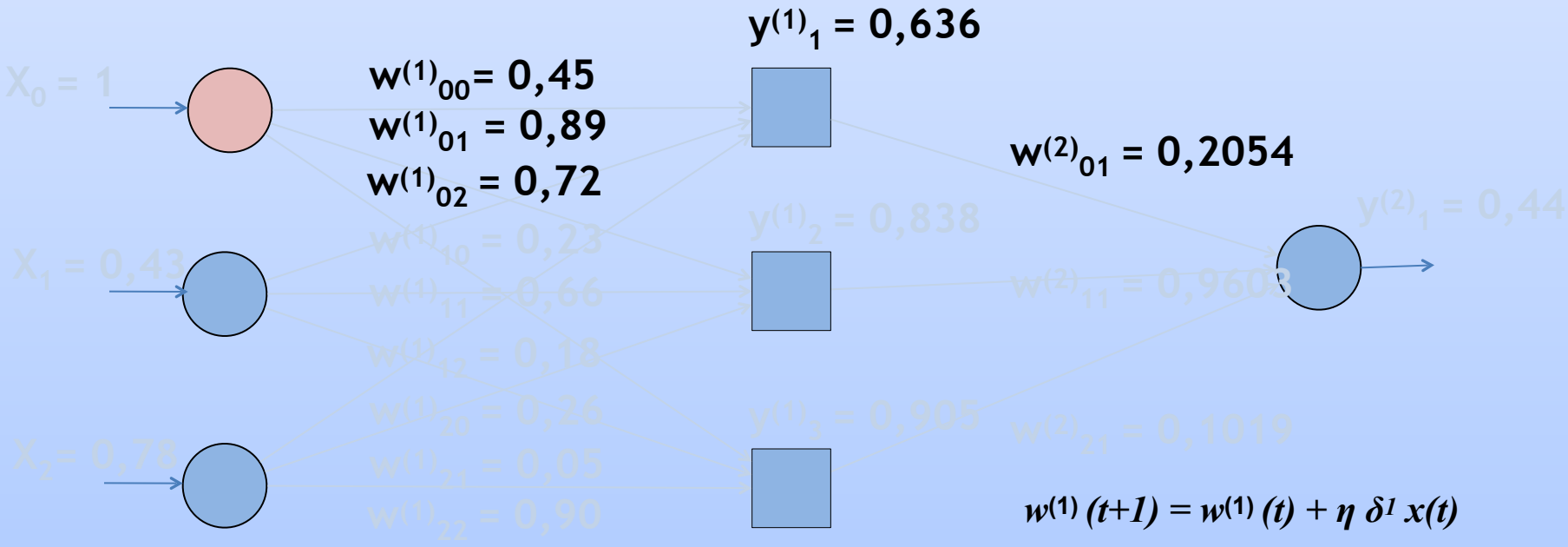
$\delta^{(2)}(t) = 0,247 * 0,981 = 0,2423$

	$w^{(2)}(t)$	η	$\delta^{(2)}$	$y(t)$	$w^{(2)}(t+1)$
$w^{(2)}_{01}$	0.19	0.1	0.2423	0.636	0.2054
$w^{(2)}_{11}$	0.94	0.1	0.2423	0.838	0.9603
$w^{(2)}_{21}$	0.08	0.1	0.2423	0.905	0.1019

Treinamento

$\eta = 0,1$
$d = 1$

7. Calcula variação dos pesos da 1ª camada



$$w^{(1)}(t+1) = w^{(1)}(t) + \eta\, \delta^{1}\, x(t)$$

$$\delta^{(1)}{}_1(t) = \left(\sum_{k=1}^{n} \delta^{(2)} * W_{kj}^{(2)}\right) * y'^{(1)}{}_1$$

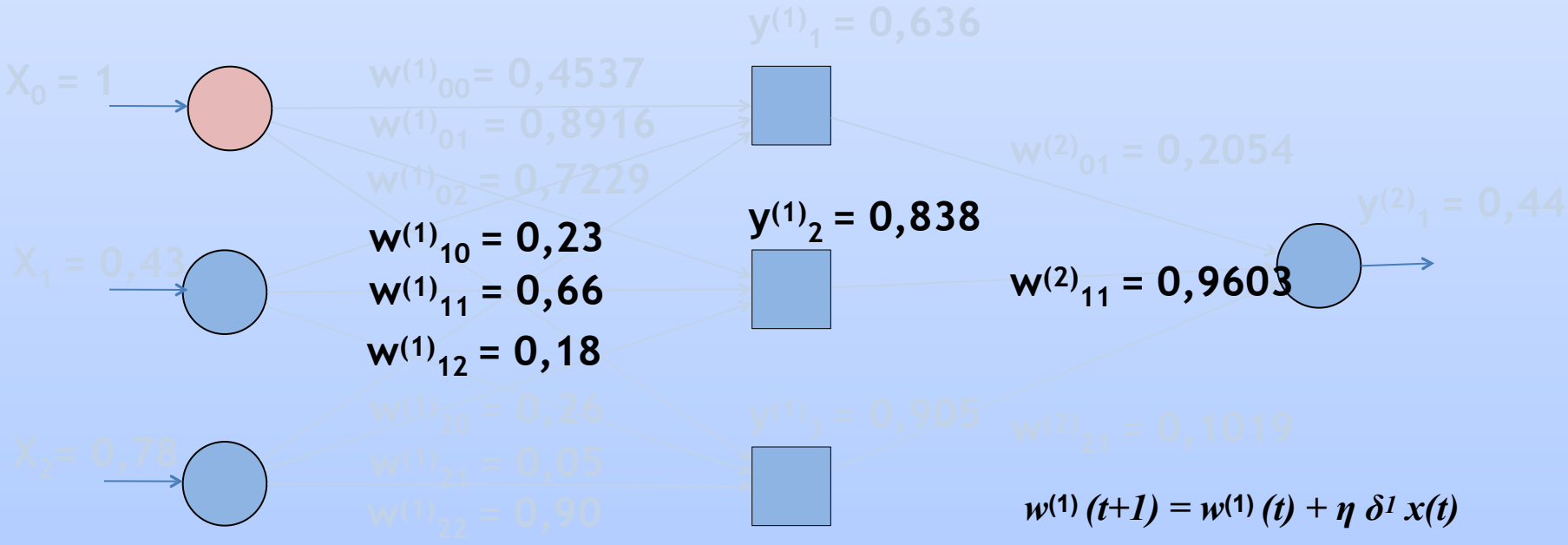
$$\delta^{(1)}{}_1(t) = (0,2423 * 0,2054) * \text{ATANH}(0,636) = 0,0374$$

	$w^{(1)}(t)$	η	$\delta^{(1)}$	$x(t)$	$w^{(1)}(t+1)$
$w^{(1)}_{01}$	0.45	0.1	0.0374	1	0.4537
$w^{(1)}_{11}$	0.89	0.1	0.0374	0.43	0.8916
$w^{(1)}_{21}$	0.72	0.1	0.0374	0.78	0.7229

Treinamento

$\eta = 0,1$
$d = 1$

7. Calcula variação dos pesos da 1ª camada



$$w^{(1)}(t+1) = w^{(1)}(t) + \eta\, \delta^{1}\, x(t)$$

$$\delta^{(1)}_2(t) = \left(\sum_{k=1}^{n} \delta^{(2)} * W^{(2)}_{kj}\right) * y'^{(1)}_1$$

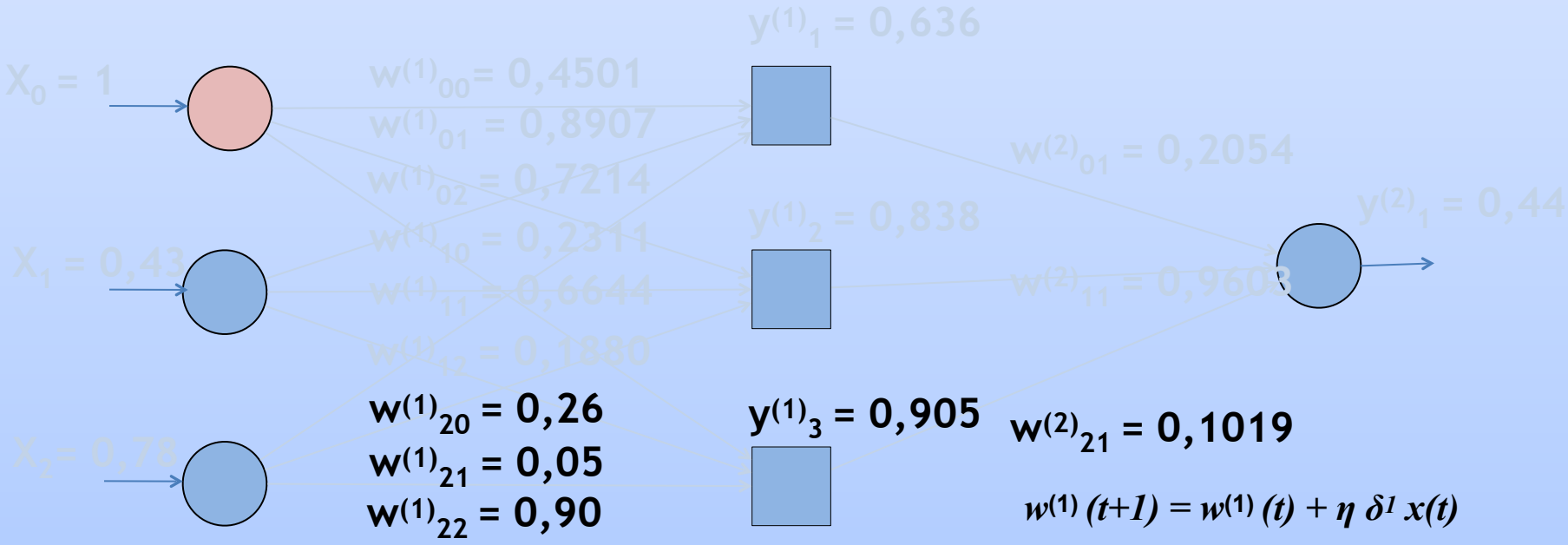
$$\delta^{(1)}_2(t) = (0,2423 * 0,9603) * \text{ATANH}(0,838) = 0,2821$$

	$w^{(1)}(t)$	η	$\delta^{(1)}$	$x(t)$	$w^{(1)}(t+1)$
$w^{(1)}_{01}$	0.23	0.1	0.2821	1	0.2582
$w^{(1)}_{11}$	0.66	0.1	0.2821	0.43	0.6721
$w^{(1)}_{21}$	0.18	0.1	0.2821	0.78	0.2020

Treinamento

$\eta = 0,1$
$d = 1$

7. Calcula variação dos pesos da 1ª camada



$w^{(1)}_{20} = 0,26$
$w^{(1)}_{21} = 0,05$
$w^{(1)}_{22} = 0,90$

$y^{(1)}_3 = 0,905$

$w^{(2)}_{21} = 0,1019$

$$w^{(1)}(t+1) = w^{(1)}(t) + \eta\, \delta^1\, x(t)$$

$$\delta^{(1)}_3(t) = \left(\sum_{k=1}^{n} \delta^{(2)} * W^{(2)}_{kj}\right) * y'^{(1)}_1$$

$$\delta^{(1)}_3(t) = (0,2423 * 0,1019) * \text{ATANH}(0,905) = 0,0370$$

	$w^{(1)}(t)$	η	$\delta^{(1)}$	$x(t)$	$w^{(1)}(t+1)$
$w^{(1)}_{01}$	0.26	0.1	0.0370	1	0.2637
$w^{(1)}_{11}$	0.05	0.1	0.0370	0.43	0.0515
$w^{(1)}_{21}$	0.90	0.1	0.0370	0.78	0.9028

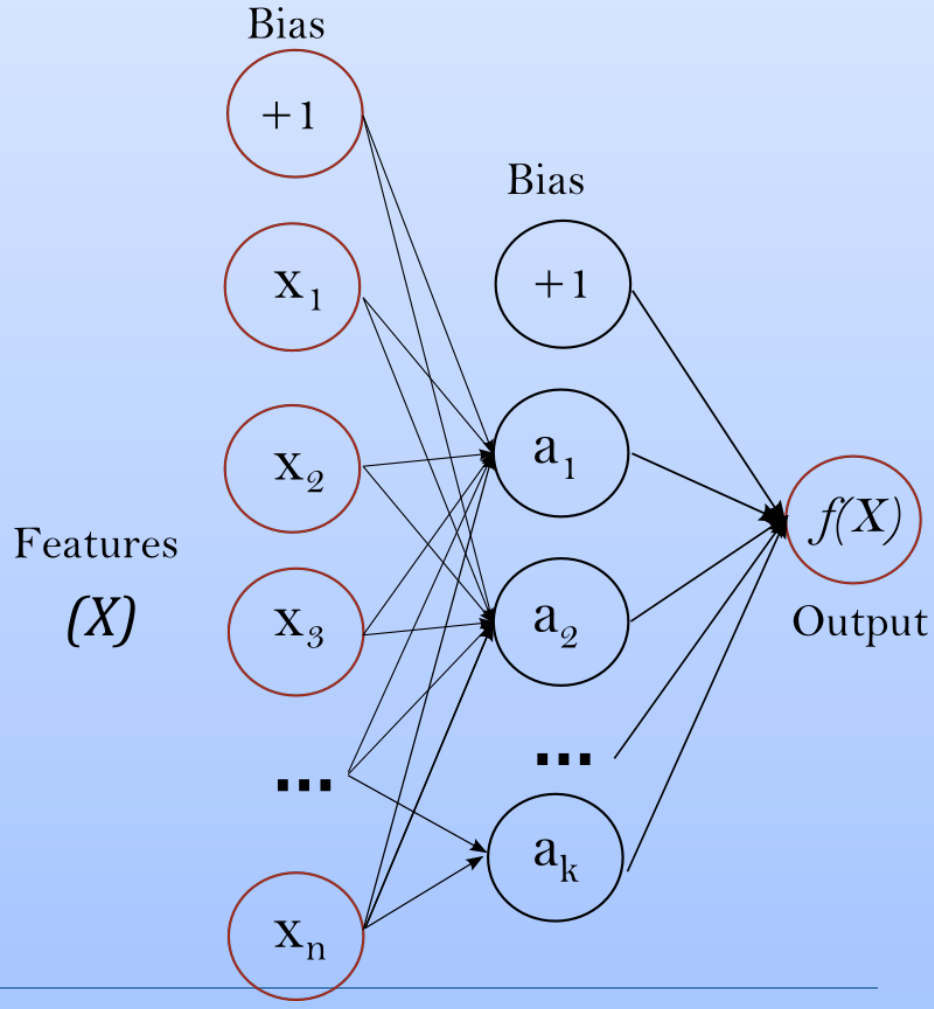
Treinamento

8. Repetir até k = número de interações desejada ou Erro = erro aceitável

...

Implementação em python via MLPClassifier

- Script disponível
  - Rede Neural para problemas supervisionados
    - NeuralNetwork.py



Principais parâmetros

- Número de camadas
  - Pode variar de 1 até 3 camadas escondidas
- Funções de ativação
  - Lineares e não-lineares
- Algoritmos de treinamento
  - Quasi-Newtom, Gradiente-descendente, Levenberg-Marquardt etc
- Épocas de Treinamento
  - Sensível a contexto