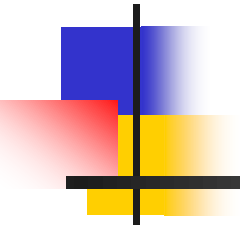
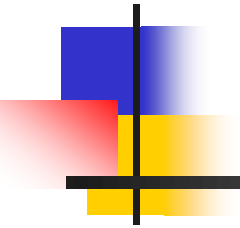
Gradiente Descendente





Redes Neurais Artificiais

Definições

- Motivação para pesquisas em Redes Neurais Artificiais (RNAs):
  - O cérebro humano processa informações de uma forma inteiramente diferente de um computador digital convencional.
- O cérebro é um sistema de processamento de informação altamente **Complexo**, **Não-Linear** e **Paralelo**.
- Seja 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.
  - Como o computador digital convencional faz isso?

Definições

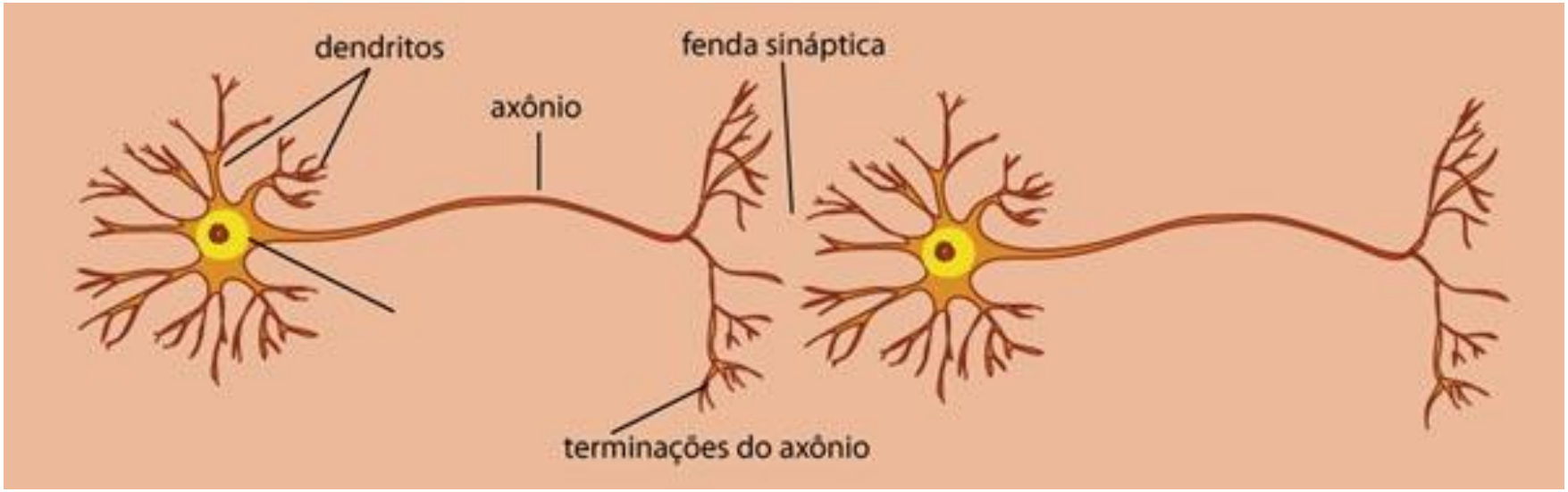
- Como...
  - O **cérebro** é capaz de realizar o reconhecimento perceptivo, e outras tantas tarefas complexas, em um intervalo tão curto de tempo?
  - Ao passo que tarefas de complexidade muito menor podem levar dias para serem executadas em um **computador convencional**?

- 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



Histórico

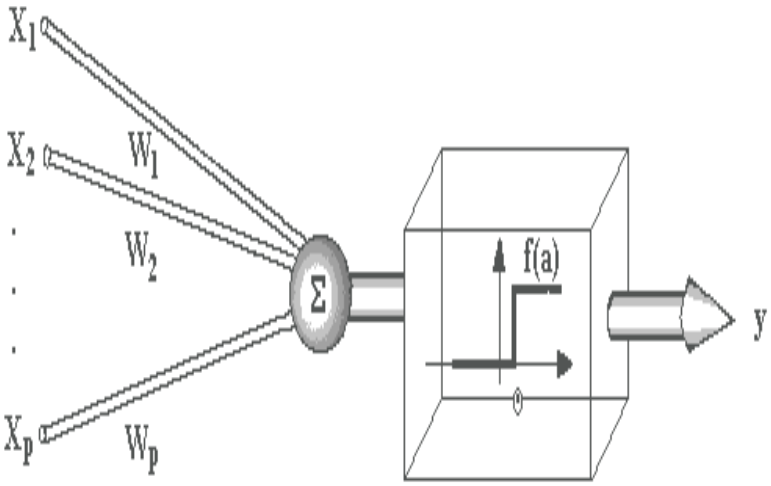
- McCulloch e Pitts (1943), Hebb (1949), e Rosemblatt (1958). Estas publicações introduziram o primeiro modelo de redes neurais simulando "máquinas", o modelo básico de rede de auto-organização, e o modelo Perceptron de aprendizado supervisionado, respectivamente.

- Nos anos 60 e 70, importantes trabalhos sobre modelos de redes neurais em visão, memória, controle e auto-organização como: Amari, Anderson, Cooper, Cowan, Fukushima, Grossberg, Kohonen, von der Malsburg, Werbos e Widrow.

- Alguns históricos sobre a área costumam "pular" os anos 60 e 70 e apontar um reínicio da área com a publicação dos trabalhos de Hopfield (1982) relatando a utilização de redes simétricas para otimização e de Rumelhart, Hinton e Williams que introduziram o método Backpropagation.

O Neurônio Artificial

- McCullock e Pitts 1943,
- sinais são apresentados à entrada;
- cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade;
- é feita a soma ponderada dos sinais que produz um nível de atividade;
- se este nível de atividade exceder um certo limite (threshold) a unidade produz uma determinada resposta de saída.

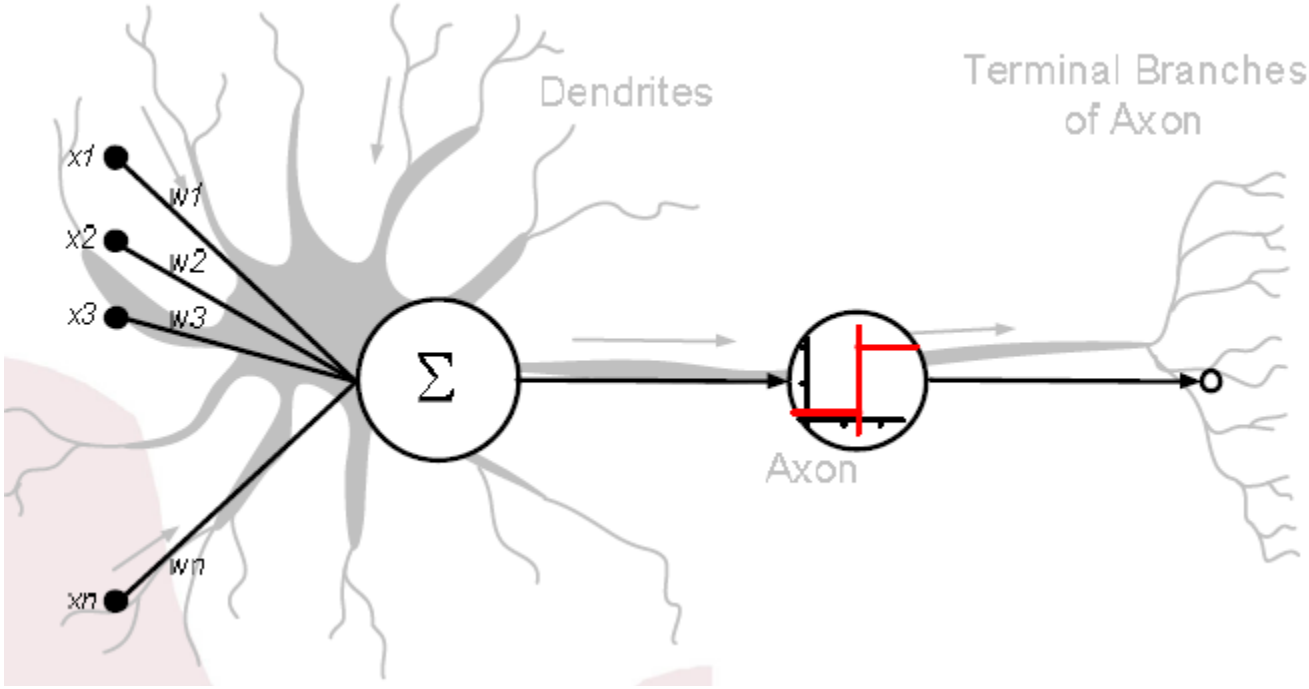


Exemplo

- sinais de entrada X1, X2, ..., Xp (0 ou 1)
- pesos w1, w2, ..., wp, valores reais.
- limitador t;
- Neste modelo, o nível de atividade a é dado por:
  - a = w1X1 + w2X2 + ... + wpXp
- A saída y é dada por:
  - y = 1, se a >= t ou
  - y = 0, se a < t.

Percepton

- Funções de classificação binária
- Função de ativação



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.

Áreas de aplicações

- Classificação de padrões
- *Clustering* /categorização
- Aproximação de funções
- Previsão
- Otimização
- Memória endereçável pelo conteúdo
- Controle
- entre outras

Processo de Aprendizagem

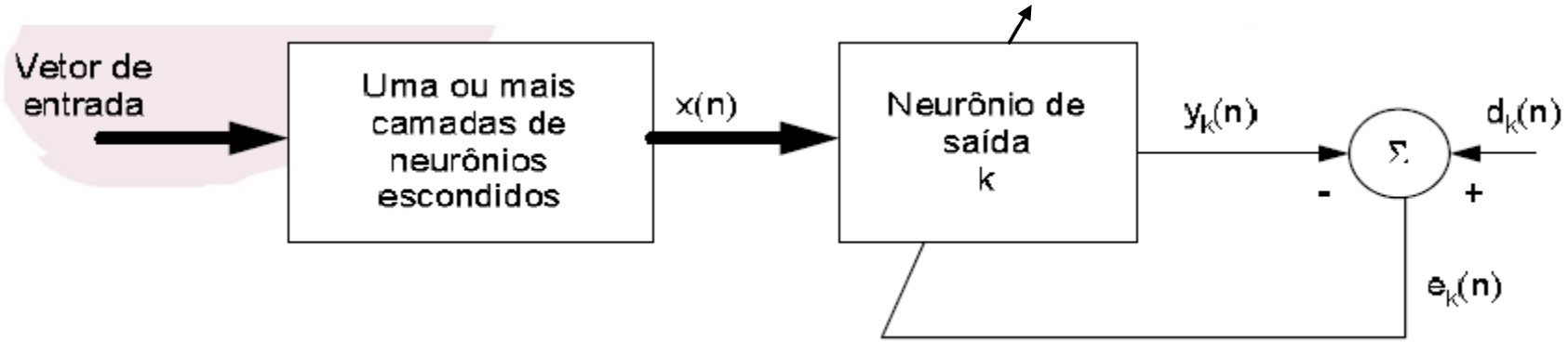
- Uma rede neural artificial pode se encontrar em duas fases:
  - A primeira fase é a de **aprendizagem**, ou **treinamento**, em que a rede se encontra no processo de aprendizado, **ajustando os parâmetros livres**, para poder posteriormente desempenhar a função destinada; e
  - A segunda fase é a de **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**

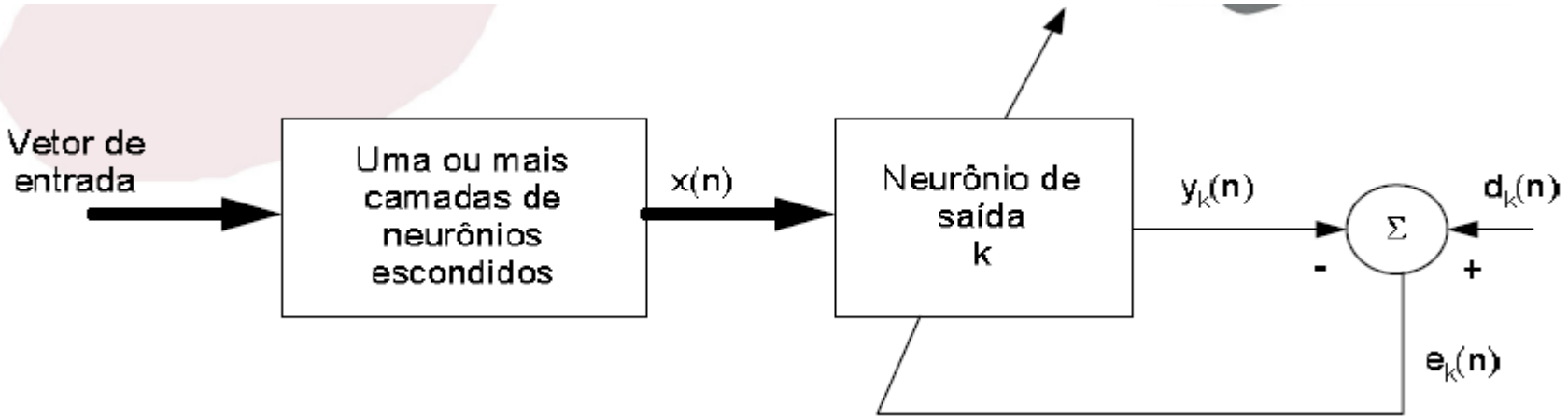
Aprendizagem por Correção de Erro



- O **sinal de saída** do neurônio **k** é representado por $\mathbf{y_k(n)}$, e a **resposta desejada** por $\mathbf{d_k(n)}$, produzido um **sinal de erro**:
  - $\mathbf{e_k\,(n) = d_k\,(n) - y_k\,(n)}$
- O sinal de erro $\mathbf{e_k(n)}$ 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 $\mathbf{y_k\,(n)}$ da resposta desejada $\mathbf{d_k(n)}$.
- Este objetivo é alcançado **minimizando-se** uma **função de custo** ou índice de desempenho, **E(n)**, definido em termos do sinal de erro como:

  $E\,(n) = ½\ ek^2(n)$

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:

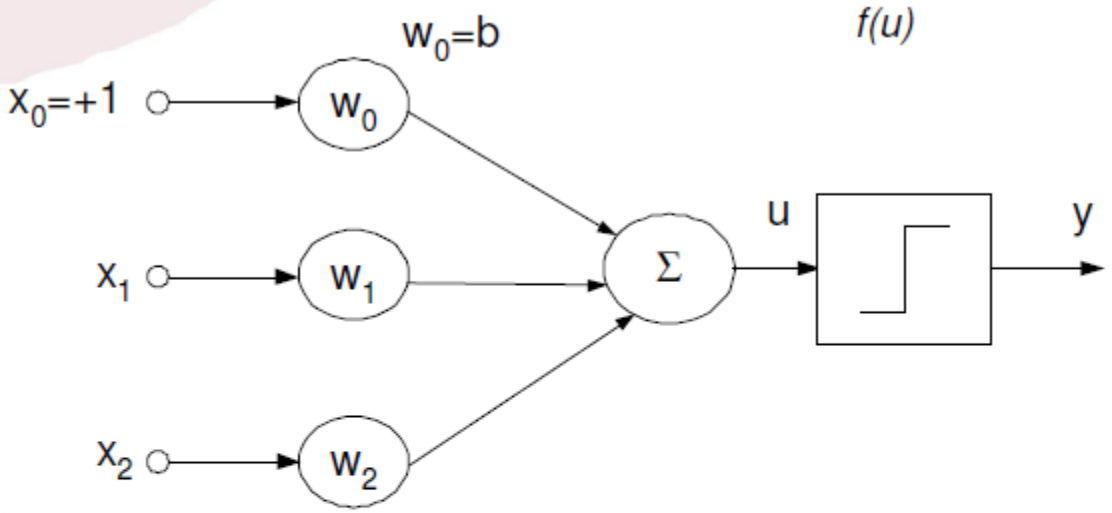
$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n).$$

O Perceptron

- O tipo mais simples de rede neural artificial foi proposto em 1958 por Frank Rosenblatt, conhecido como perceptron. A palavra em latim para o verbo compreender é “percipio“, e sua forma supina é “perceptum”, ou seja, a rede deve ser capaz de compreender o mundo exterior.
- Esse algoritmo de aprendizagem supervisionada considera um período de treinamento (com valores de entrada e saída) para definir se uma nova entrada pertence a alguma classe específica ou não.

O Perceptron

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



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

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:

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

Papel do Bias

- O uso do bias permite que fixemos o valor de ***threshold*** adotado em nossa **função de ativação**, sendo necessário então atualizar somente os **pesos** e o **bias** na rede.

- Como o bias pode ser encarado como sendo o peso para um neurônio cuja entrada é sempre 1, percebe-se que a mesma **regra para atualização dos pesos** é válida também para a atualização do bias.

Conceitos

- 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.

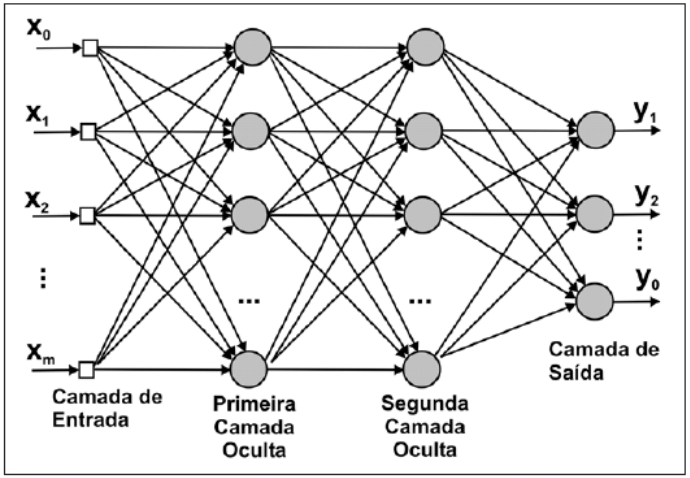


Figura 2 - Rede multilayer feedforward.

Conceitos

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

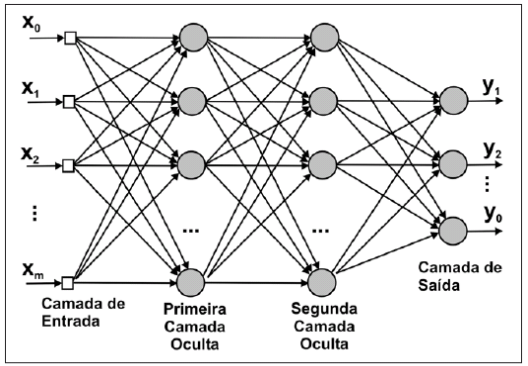


Figura 2 - Rede multilayer feedforward.

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

Conceitos

- Algoritmo Back-propagation
  - Década de 80. Novo "gás" para área de redes neurais.
- Gradiente descendente
  - Estimação 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.

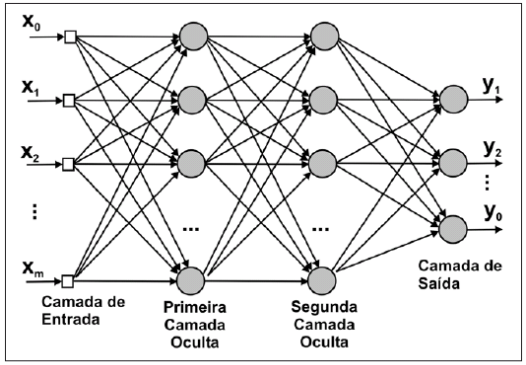


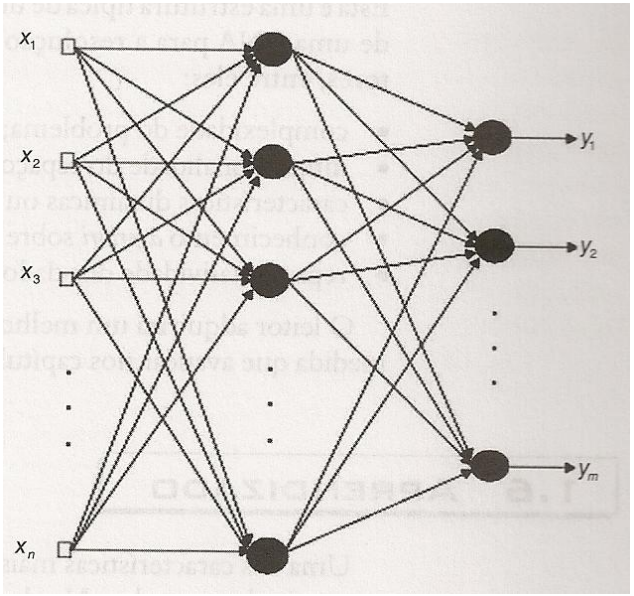
Figura 2 - Rede multilayer feedforward.

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

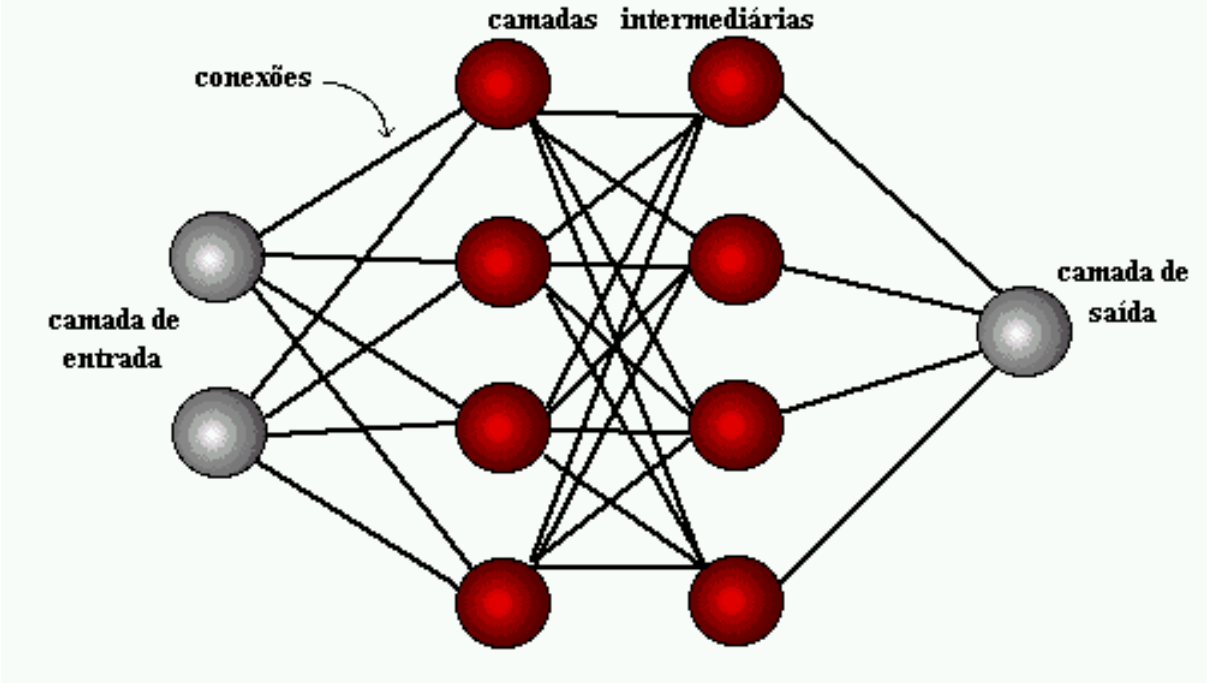


Arquitetura

- Usualmente as camadas são classificadas em três grupos:
  - **Camada de Entrada**: onde os padrões são apresentados à rede;
  - **Camadas Intermediárias ou Escondidas**: onde é feita a maior parte do processamento, através das conexões ponderadas; podem ser consideradas como extratoras de características;
  - **Camada de Saída**: onde o resultado final é concluído e apresentado.

Arquitetura

- Camadas



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

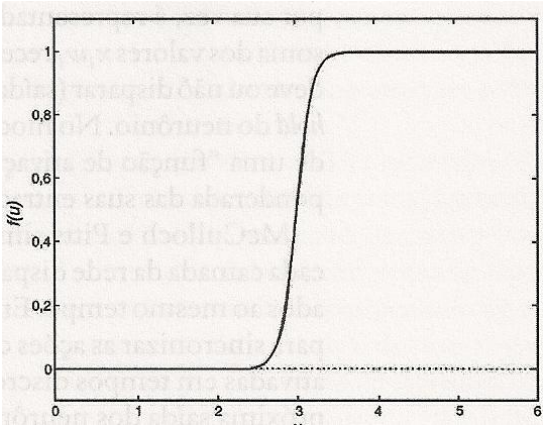
Arquitetura

- Número de 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.

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

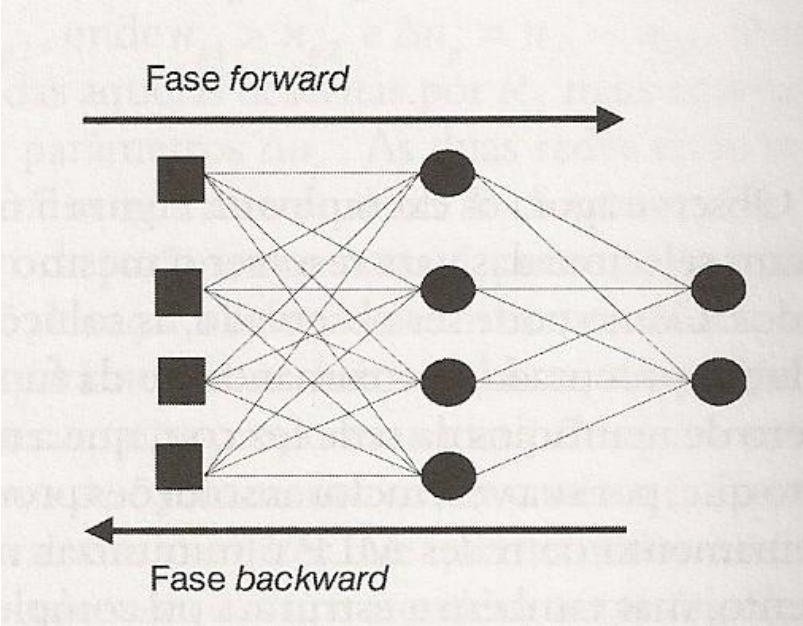


Aprendizado

- 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

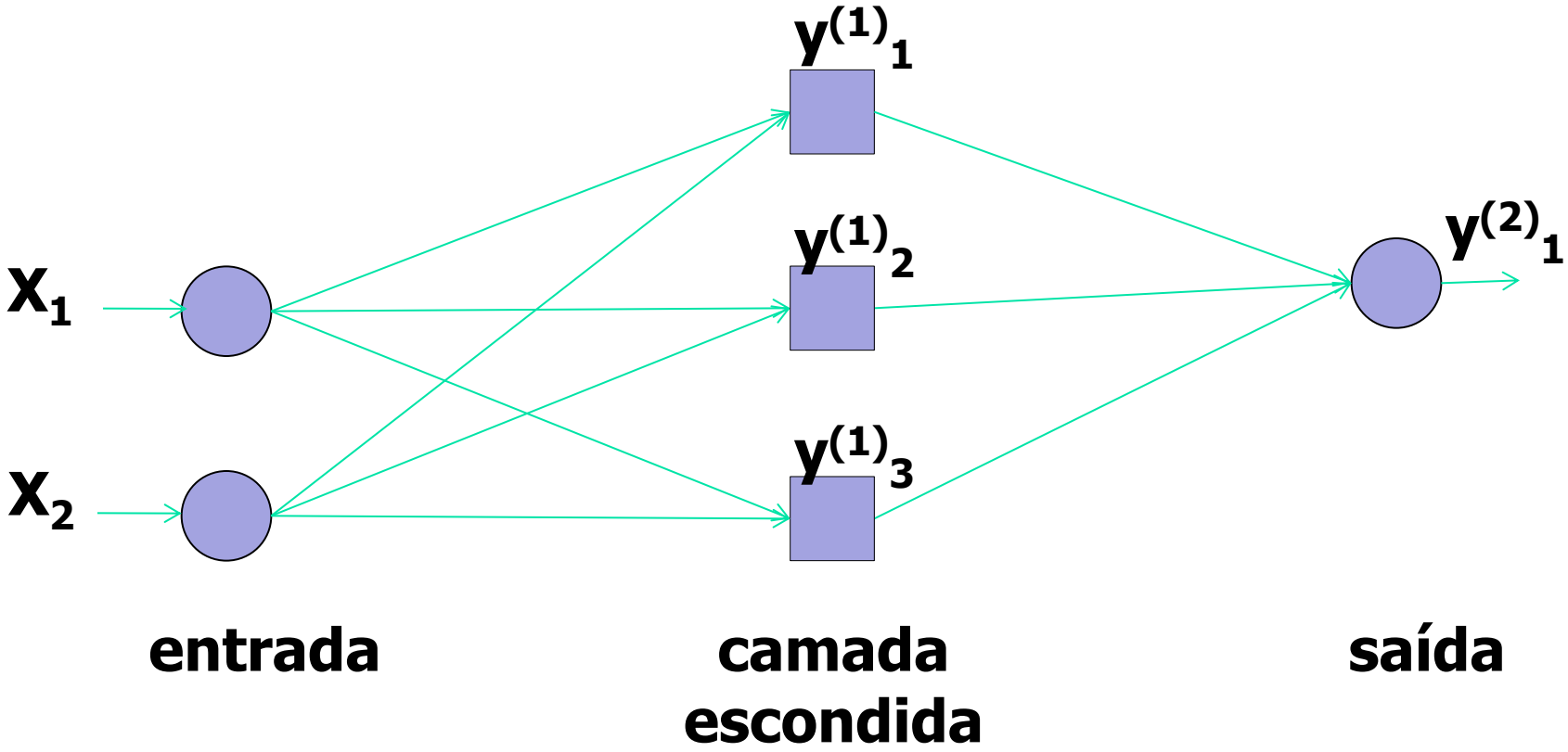
- Treinamento em duas fases:



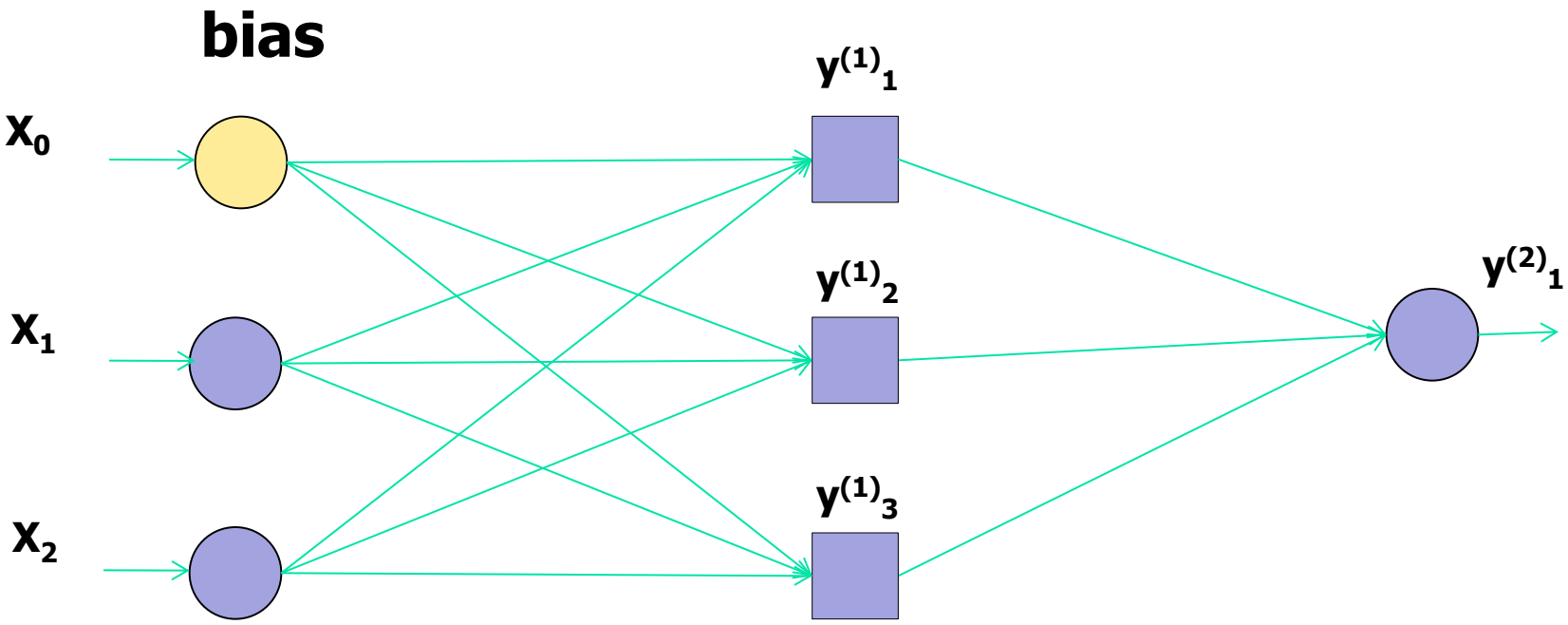
Implementação do Algoritmo

- ***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

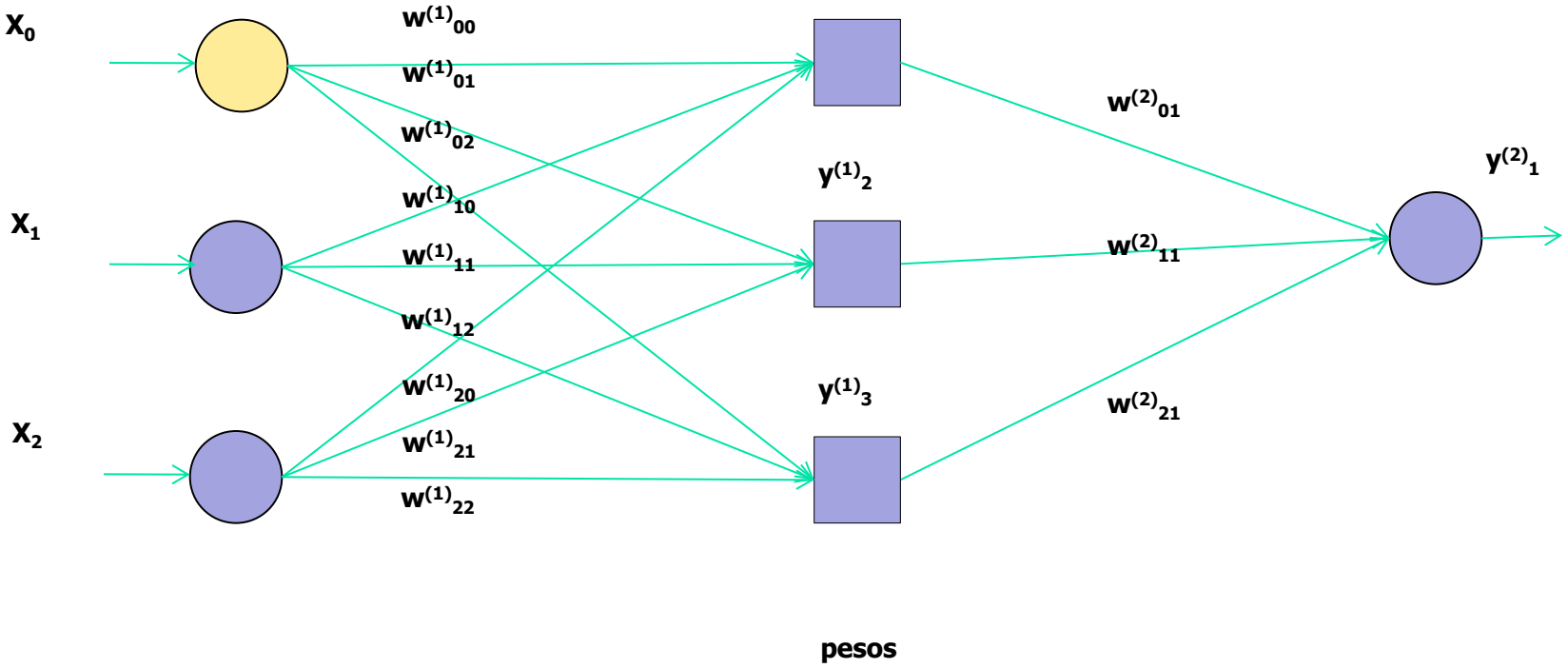
Treinamento



Treinamento



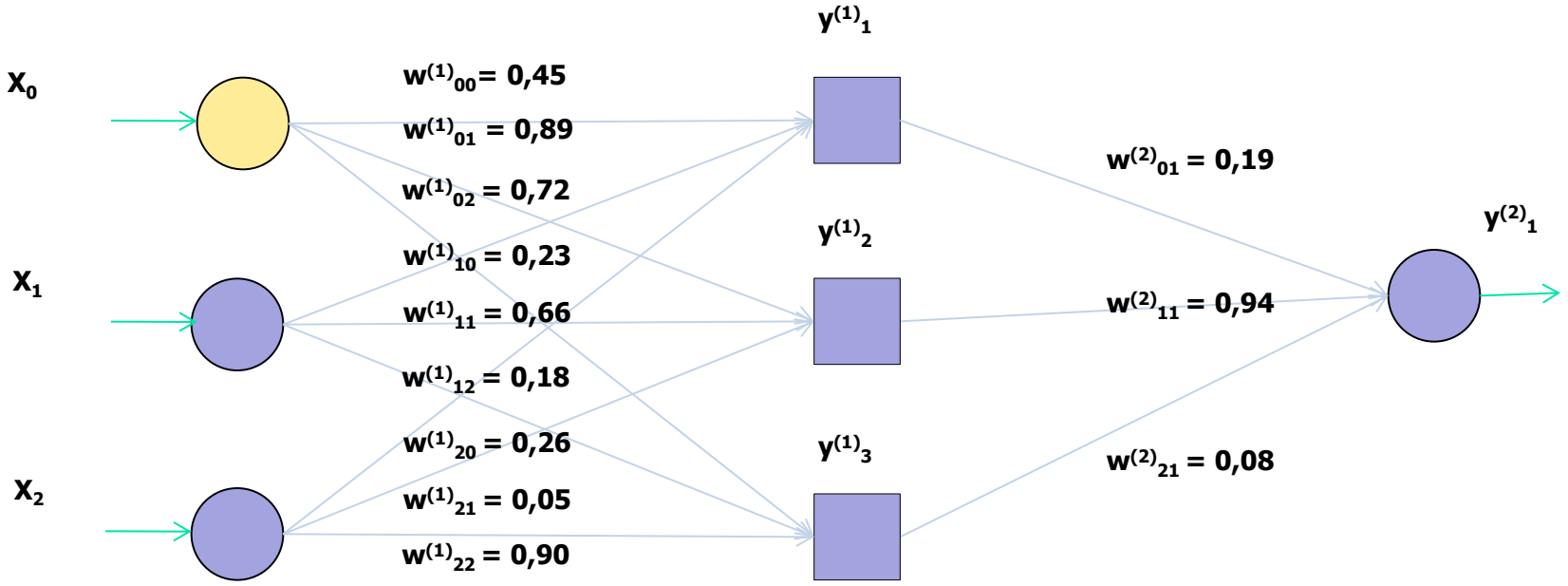
Treinamento



Treinamento

$\eta = 0,1$

1. Inicia todas as conexões com pesos aleatórios

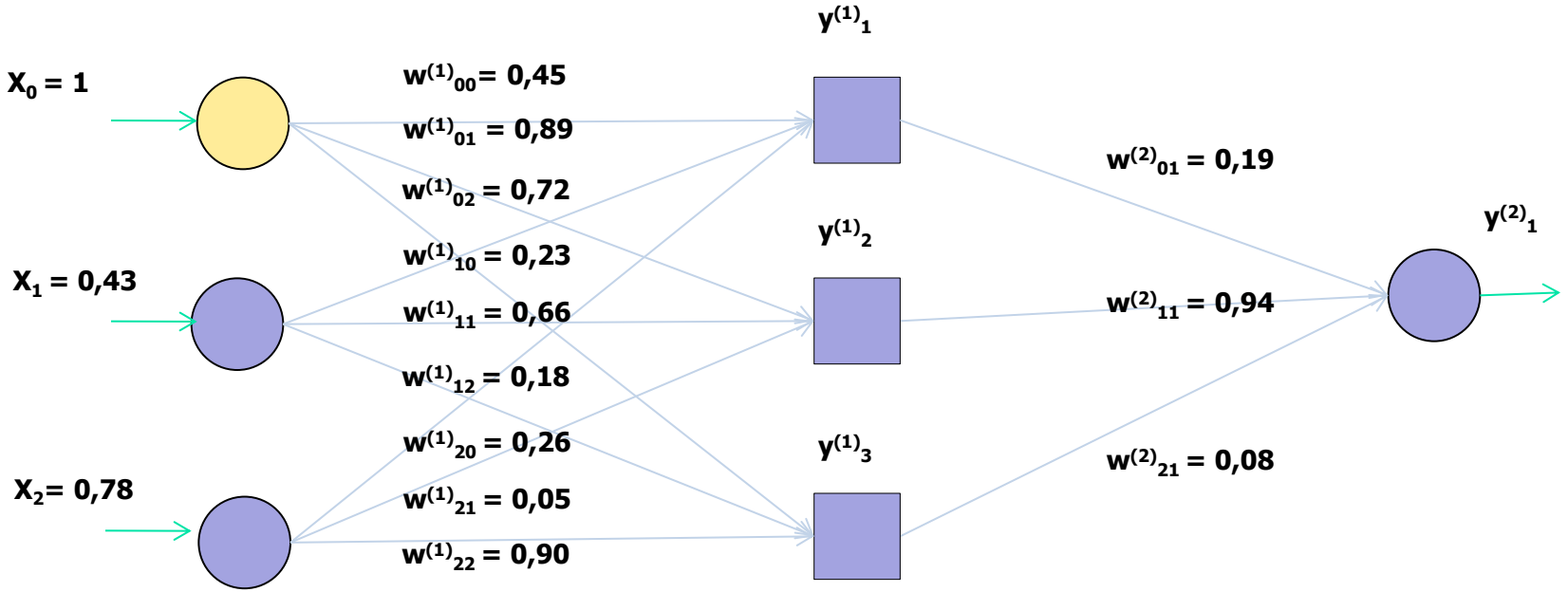


Treinamento

$\eta = 0,1$

d = 1

2. Para de entrada X é apresentado

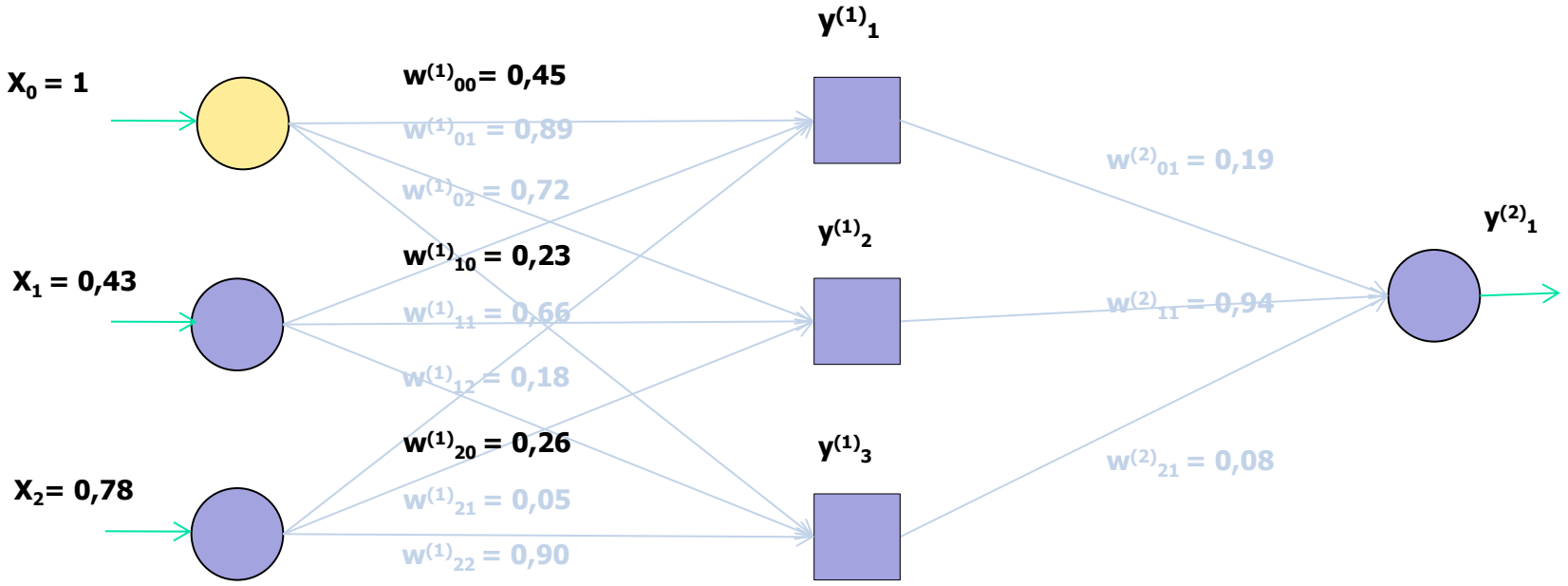


Treinamento

$\eta = 0,1$

d = 1

3. Calcula saída para 1ª camada

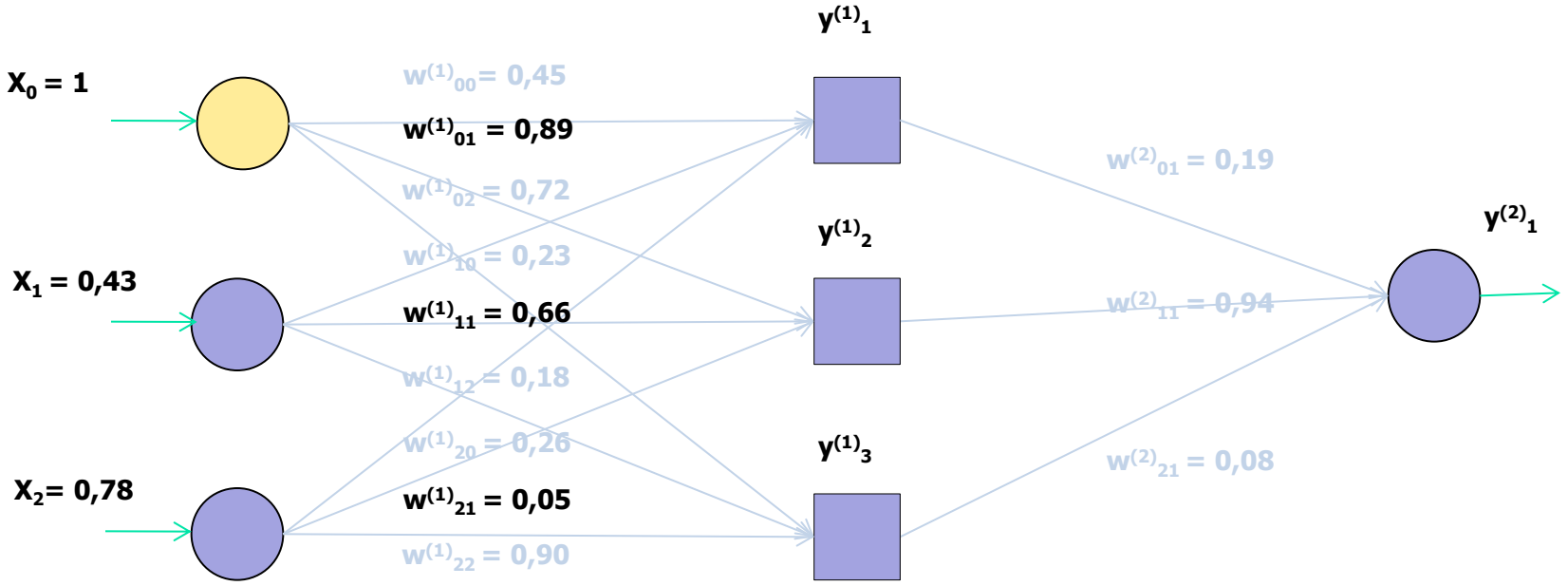


$u^{(1)}_1 = (1*0,45) + (0,43*0,23) + (0,78*0,26) = 0,7517$

Treinamento

$\eta = 0,1$
d = 1

3. Calcula saída para 1ª camada



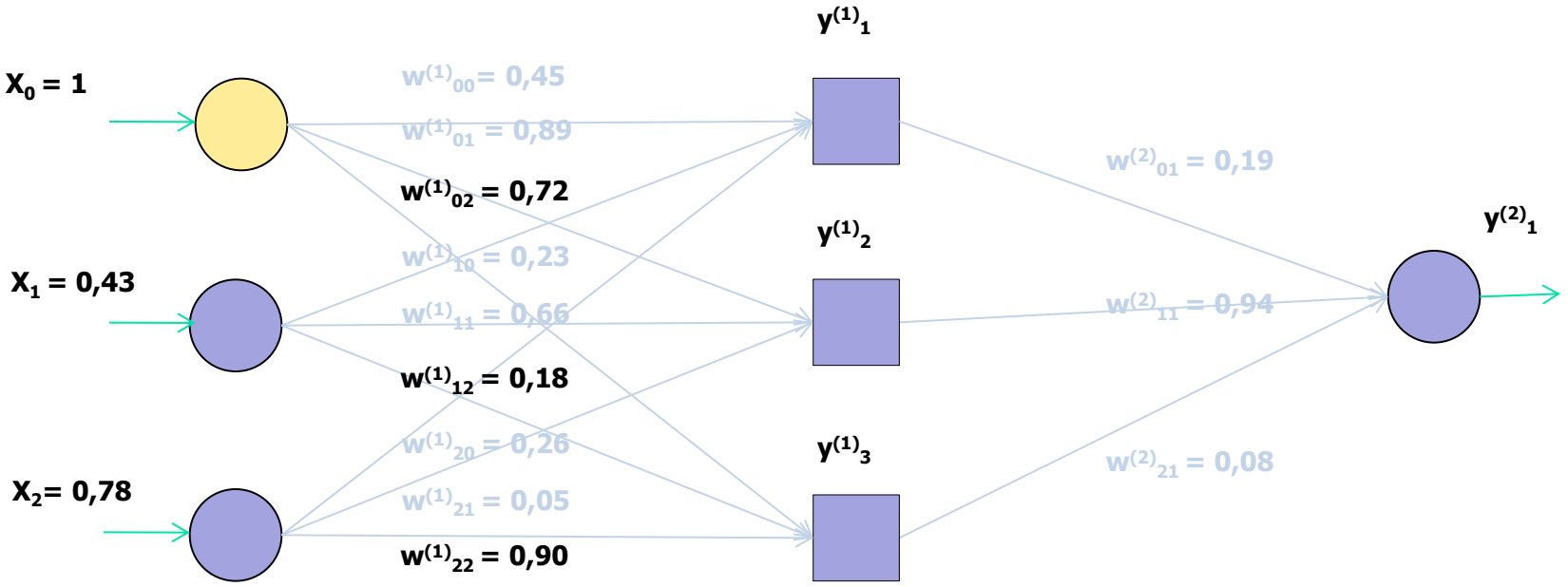
$u^{(1)}_1 = (1*0,45) + (0,43*0,23) + (0,78*0,26) = 0,7517$

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

Treinamento

$\eta = 0,1$
d = 1

3. Calcula saída para 1ª camada



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

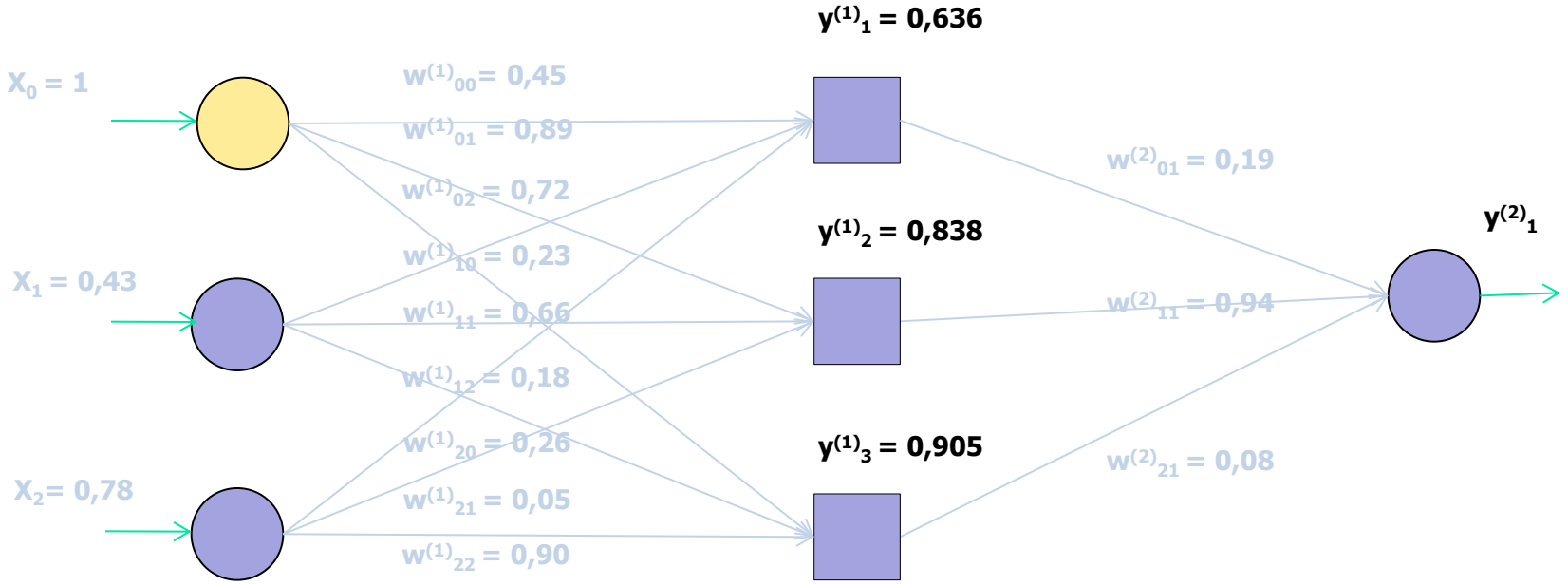
$u^{(1)}_2 = (1*0,89) + (0,43*0,66) + (0,78*0,05) = 1,2128$

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

Treinamento

$\eta = 0,1$
d = 1

4. Calcula saída da função de ativação



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

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

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

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

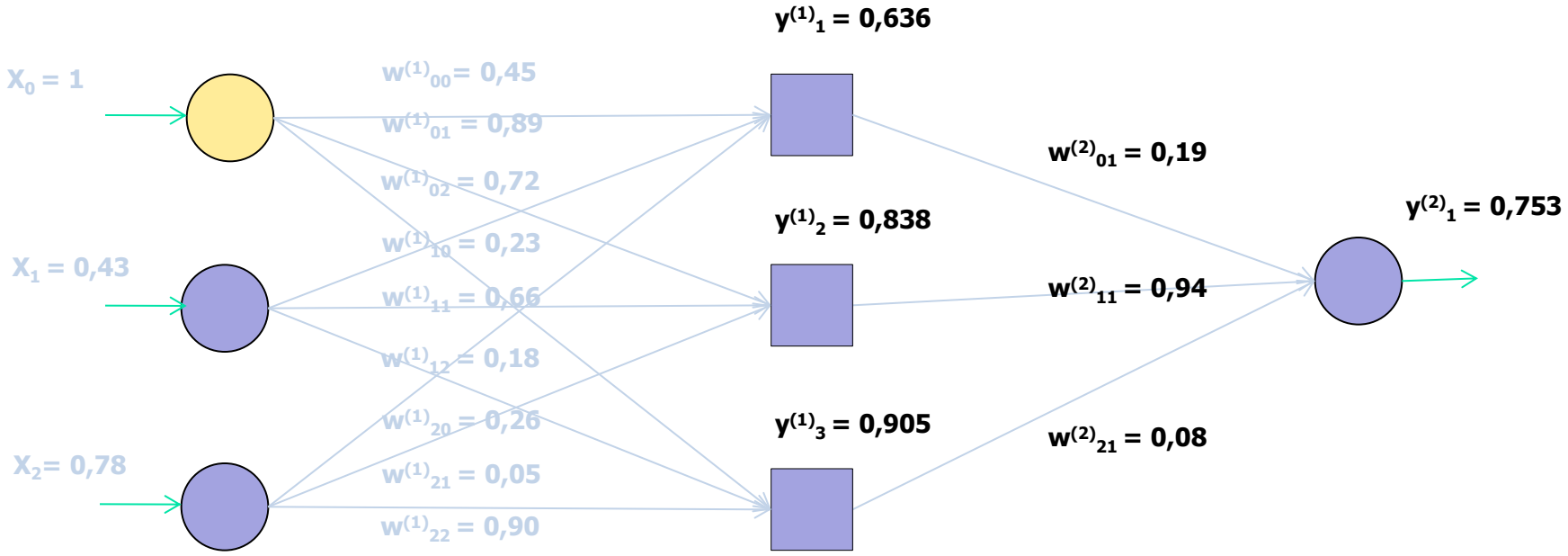
$y^{(1)}_2 = \text{TANH}(u^{(1)}_1) = 0,838$

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

Treinamento

$\eta = 0,1$
d = 1

5. Calcula saída para 2ª camada



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

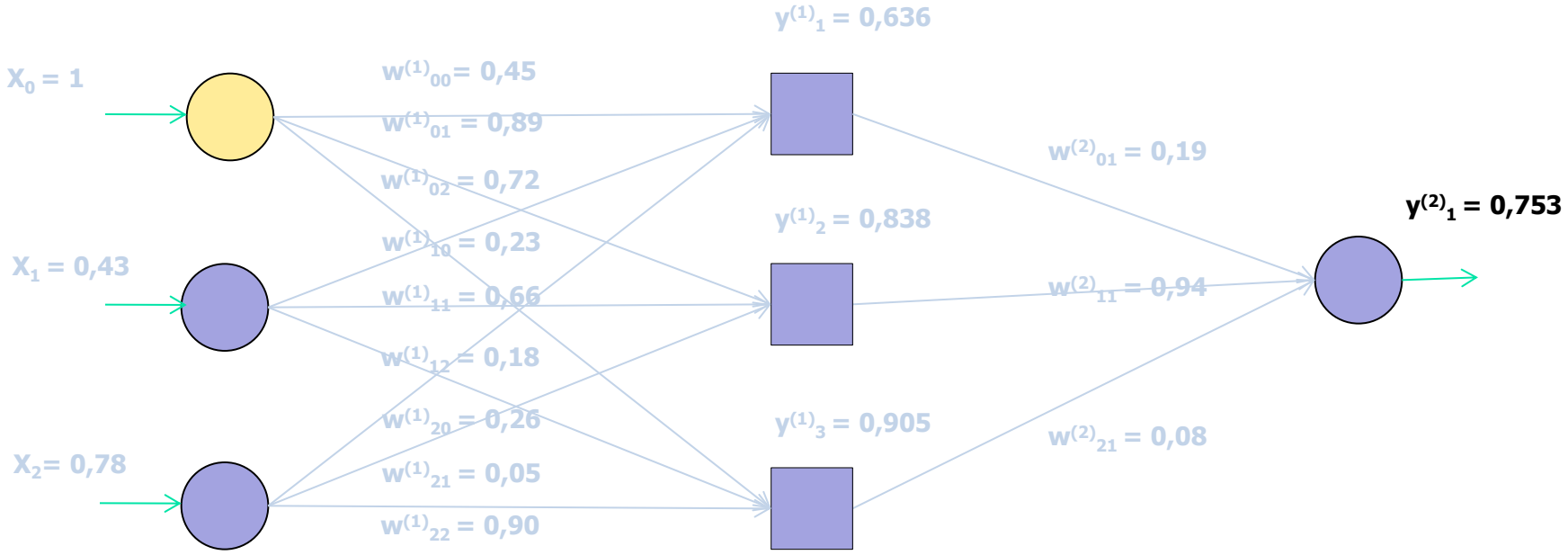
$y^{(2)}_1 = TANH(u^{(2)}_1) = 0,753$

Treinamento

$\eta = 0,1$

d = 1

6. Calcula variação do erro



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

$$E(k) = \tfrac{1}{2} (0,247)^2 = 0,03$$

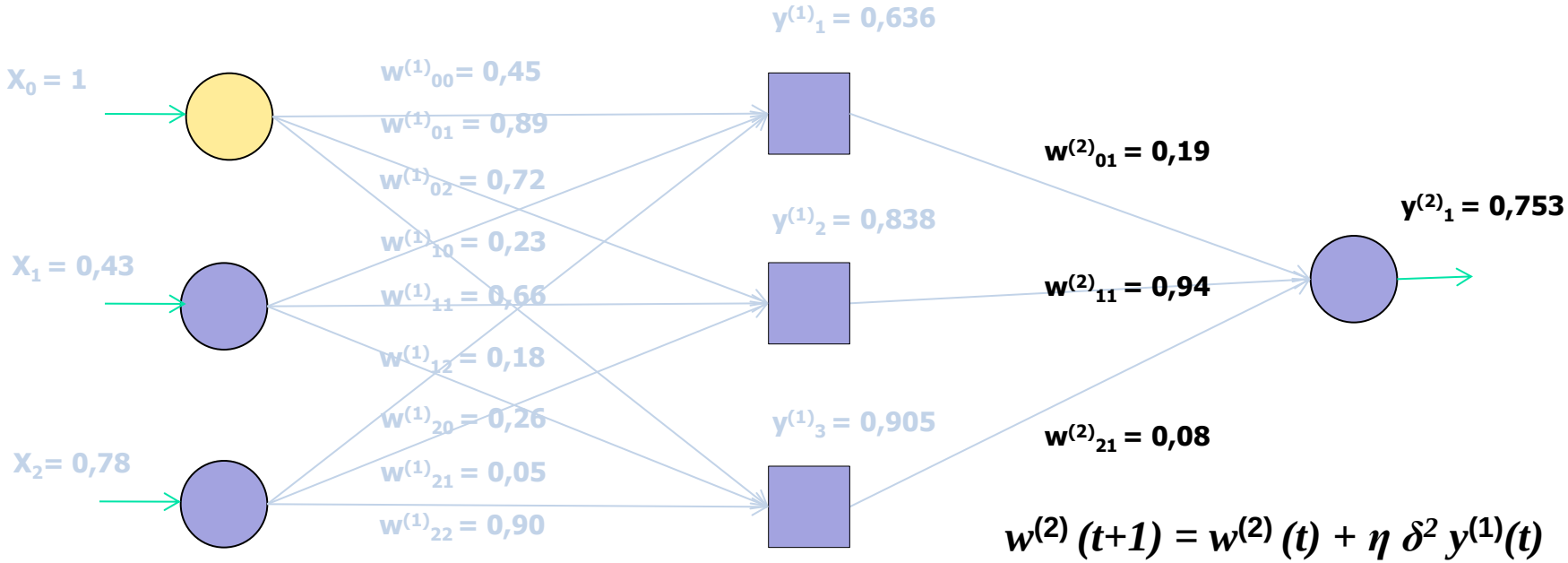
Implementação do Algoritmo

- ***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



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

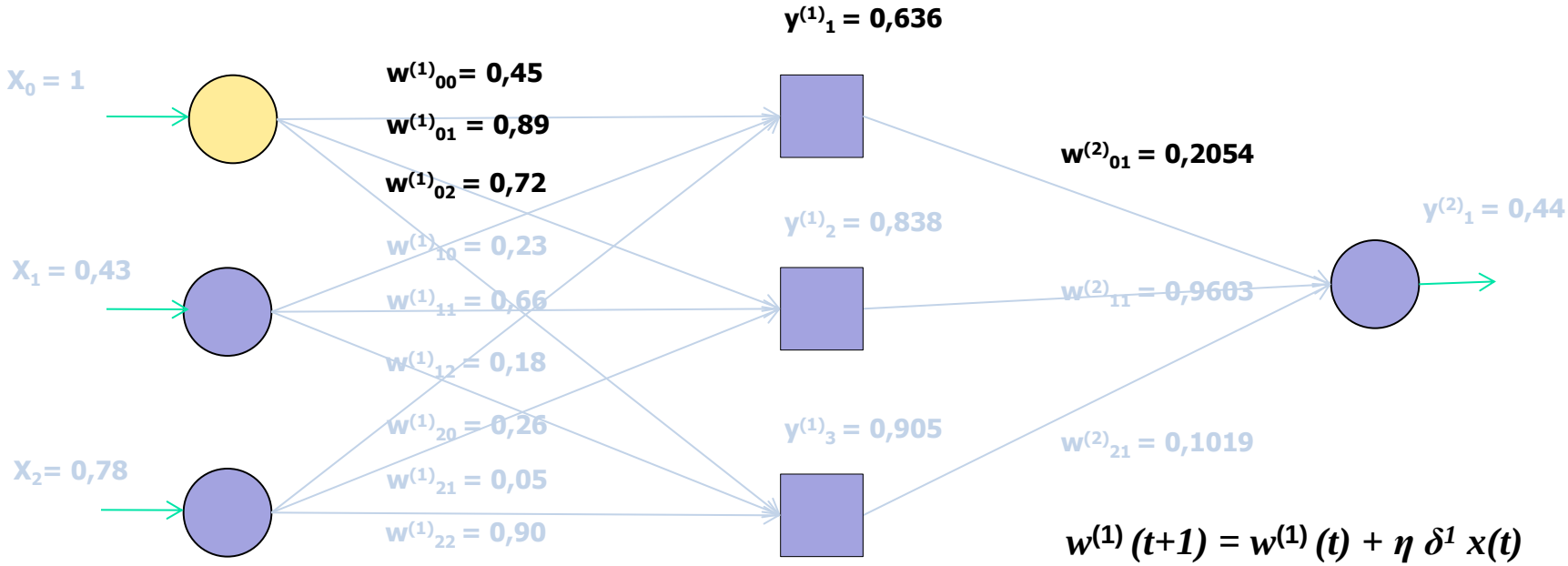
	$w^{(2)}(t)$	η	$\delta^{(2)}$	$x(t)$	$w^{(2)}(t+1)$
$w^{(2)}_{01}$	0.19	0.1	0.2423	0.636	0.2054
$w^{(2)}_{11}$	0.94	0.1	0.2423	0.838	0.9603
$w^{(2)}_{21}$	0.08	0.1	0.2423	0.905	0.1019

$$\delta^{(2)}(t) = (d(t) - y(t)) * y'^{(2)}_1$$
$$\delta^{(2)}(t) = 0,247 * \text{ATANH}(0,753)$$
$$\delta^{(2)}(t) = 0,247 * 0,981 = 0,2423$$

Treinamento

$\eta = 0,1$
d = 1

7. Calcula variação dos pesos da 1ª camada



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

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

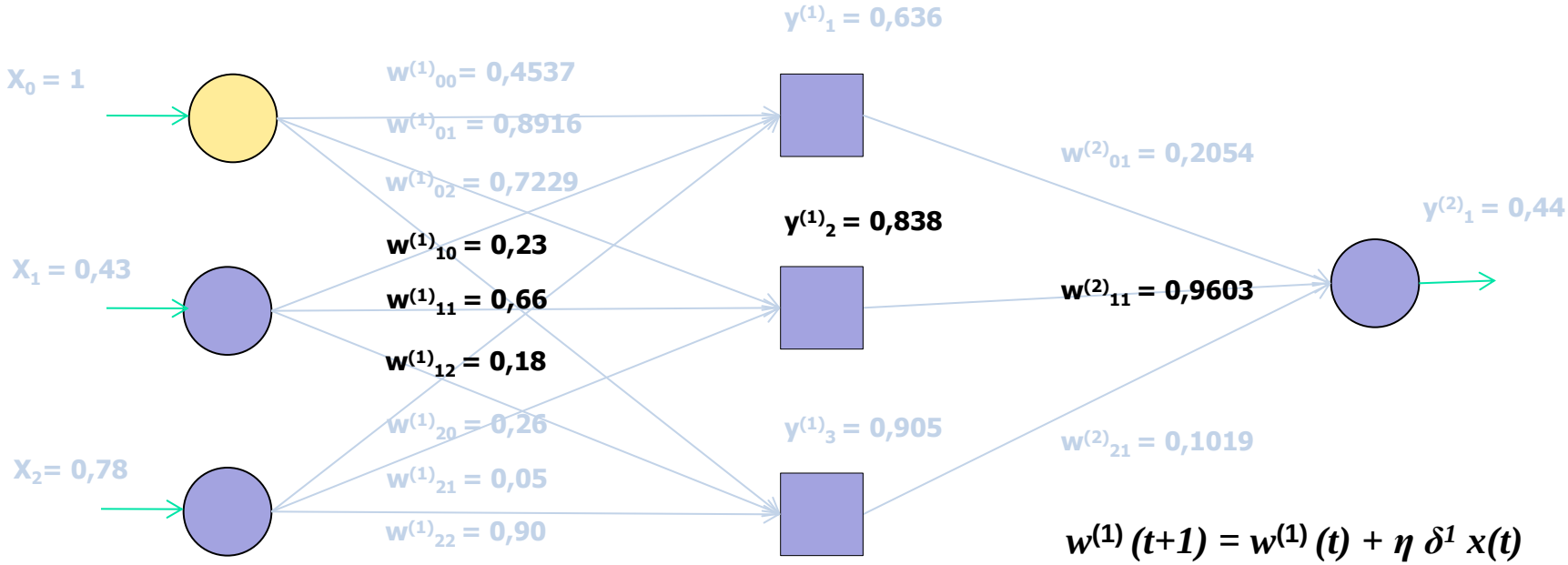
$$\delta^{(1)}_1(t) = (0,2423 * 0,2054) * \text{ATANH}(0,636) = 0,0374$$

	$w^{(1)}(t)$	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



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

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

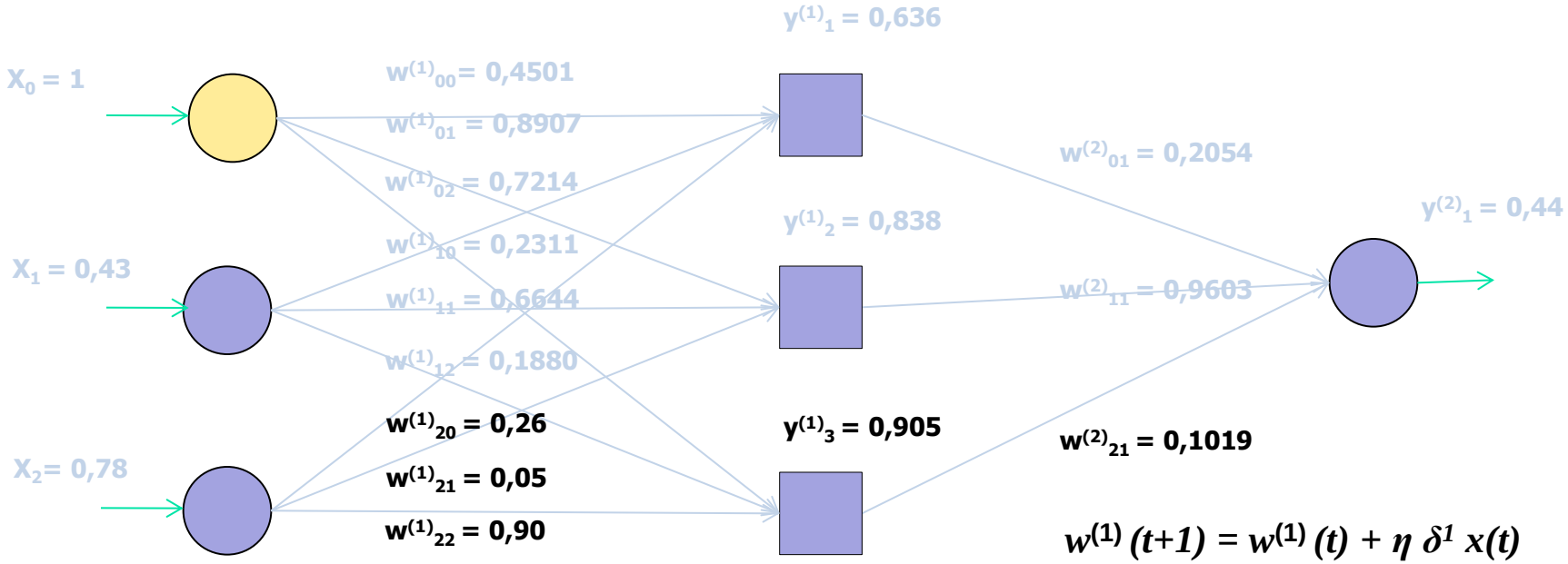
$$\delta^{(1)}_2(t) = (0,2423 * 0,9603) * \text{ATANH}(0,838) = 0,2821$$

	$w^{(1)}(t)$	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



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

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

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

	$w^{(1)}(t)$	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

Utilização

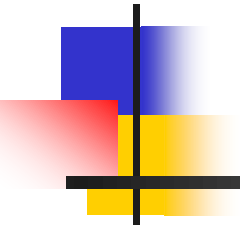
PONTOS FORTES

- São versáteis: podem ser usadas para a solução de diferentes tipos de problemas como previsão, agrupamento ou identificação de padrões;
- São capazes de identificar relações não-lineares entre as variáveis;
- São largamente utilizadas, estando disponíveis em vários *softwares.*

PONTOS FRACOS

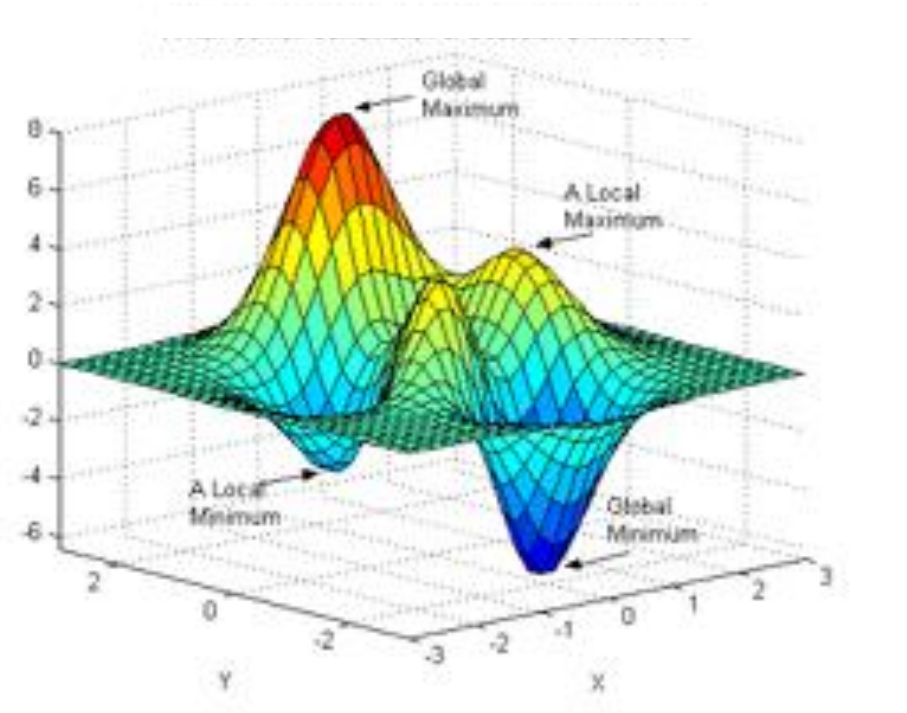
- Os resultados não são explicáveis: a análise é feita dentro da rede e só o resultado é fornecido pela "caixa-preta";
- A rede pode convergir para uma solução inferior: não há garantias de que a rede encontre a melhor solução possível; ela pode convergir para um máximo local.

Gradiente Descendente



Conceitos

- Gradiente Descendente é um algoritmo de otimização usado para minimizar algumas funções movendo-se iterativamente na direção da descida mais íngreme, conforme definido pelo negativo do gradiente. Nos modelos de machine learning, usamos gradiente descendente para atualizar os parâmetros do nosso modelo.



Conceitos

- O método do Gradiente Descendente se baseia na derivada parcial da função f(x) para cada valor X, ou seja:

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_d} \right]$$

- O Gradiente descendente (GD) e um método para encontrar o mínimo de uma função de forma iterativa:

Algoritmo: Escolha um chute inicial, $\beta^{(0)} \in \mathrm{R}^p$, repita:

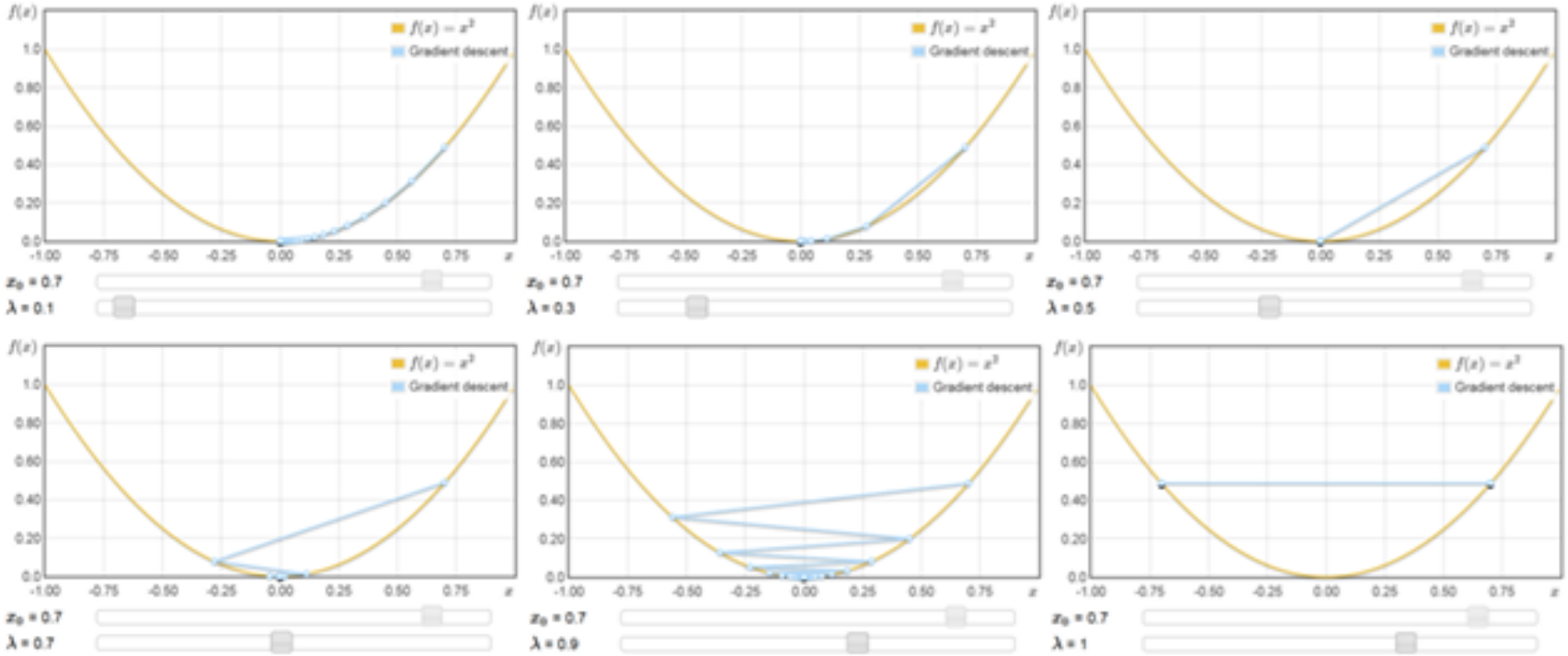
$$\beta^{(k+1)} = \beta^{(k)} - \alpha_k \nabla J(\beta^{(k)}), \ k = 0, 1, \ldots$$

pare quando atingir convergência.

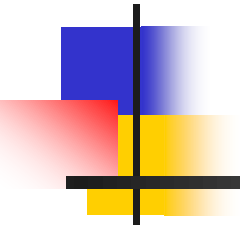
Onde J é a função custo e β são os pesos da minha função

Taxa de Aprendizagem α

- Taxa de aprendizagem controla o tamanho do passo em cada iteração;
- Selecionar o valor **α** correto é crítico
  - Se tomarmos pequeno, o método fica lento;
  - Se muito grande, o método diverge.



Gradiente Descendente Estocástico



Conceitos

- Gradiente Descendente utilizamos a amostra completa para atualizar os parâmetros (e um processo determinístico);
- Entretanto, se o tamanho da amostra de treino for muito grande o Gradiente Descendente levara muito tempo em cada passo;
- A diferença no Gradiente Descendente Estocástico (GDE) está na utilização de somente uma observação em cada iteração;
- Então, cada passo e realizado com uma v.a. de um processo estocástico. Tornando o método mais atrativo;

Conceitos

- Gradiente Descendente utilizamos a amostra completa para atualizar os parâmetros (e um processo determinístico);
- Entretanto, se o tamanho da amostra de treino for muito grande o Gradiente Descendente levara muito tempo em cada passo;
- A diferença no Gradiente Descendente Estocástico (GDE) está na utilização de somente uma observação em cada iteração;
- Então, cada passo e realizado com uma v.a. de um processo estocástico. Tornando o método mais atrativo;

Conceitos

- Considere o par (xi yi) amostrado do treinamento. A atualização dos parâmetros é dada por

Algoritmo: Escolha um chute inicial, $\beta^{(0)} \in \mathrm{R}^{p+1}$, repita:

$$\beta^{(k+1)} = \beta^{(k)} - \alpha_k \nabla J(\beta^{(k)}; \boldsymbol{x}_i, y_i), \ k = 0, 1, \ldots$$

pare quando atingir convergência.