Aprendizado a partir de Dados

1° Semestre de 2015

Cleber Zanchettin
UFPE - Universidade Federal de Pernambuco
CIn - Centro de Informática

O problema do aprendizado

Exemplo: Prever como um usuário avalia um filme

10% melhora = Prêmio de 1 milhão de dólares (Netflix)

- A essência do aprendizado de máquina:
  - **Existe um padrão**
  - **Não é possível construir uma equação matemática**
  - **Existem dados disponíveis**

Componentes do aprendizado

Metafora: Aprovação de crédito

Applicant information:

age	23 years
gender	male
annual salary	$30,000
years in residence	1 year
years in job	1 year
current debt	$15,000
...	...

Aprovar crédito?

Componentes do aprendizado

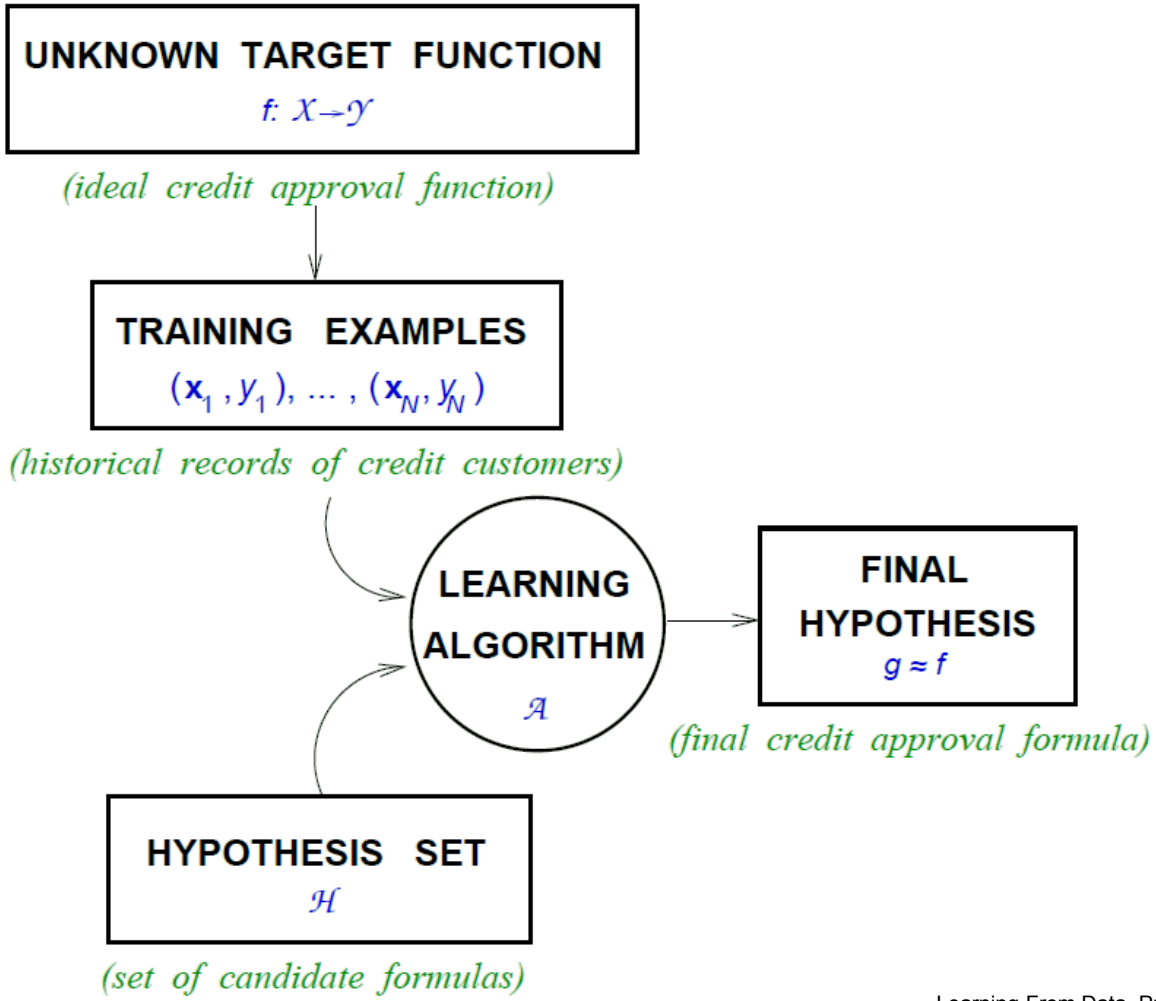
Formalization:

- Input: $\mathbf{x}$ (*customer application*)
- Output: y (*good/bad customer?*)
- Target function: $f : \mathcal{X} \rightarrow \mathcal{Y}$ (*ideal credit approval formula*)
- Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$ (*historical records*)

$\downarrow \quad \downarrow \quad \downarrow$

- Hypothesis: $g : \mathcal{X} \rightarrow \mathcal{Y}$ (*formula to be used*)

Componentes do aprendizado



Learning From Data, Prof. Yaser Abu-Mostafa

Componentes da solução

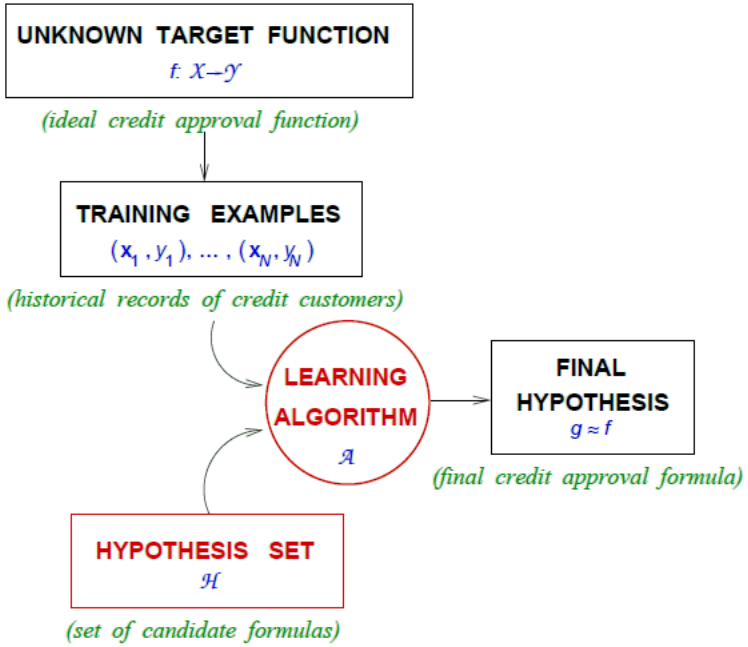
The 2 solution components of the learning problem:

- The Hypothesis Set

  $\mathcal{H} = \{h\}$ $\quad$ $g \in \mathcal{H}$

- The Learning Algorithm

Together, they are referred to as the *learning model*.



A simple hypothesis set - the 'perceptron'

For input $\mathbf{x} = (x_1, \cdots, x_d)$ 'attributes of a customer'

$$\text{Approve credit if} \quad \sum_{i=1}^{d} w_i x_i > \text{threshold},$$

$$\text{Deny credit if} \quad \sum_{i=1}^{d} w_i x_i < \text{threshold}.$$

This linear formula $h \in \mathcal{H}$ can be written as

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right)$$

Learning From Data, Prof. Yaser Abu-Mostafa

Separação Linear

- Sabe-se que se formarmos uma **combinação linear de duas variáveis**, e igualá-la a um número, então os pontos no espaço bidimensional podem ser divididos em três categorias:

  – a) pontos **pertencentes à linha** com coordenadas tais que

  $$w_1 . x_1 + w_2 . x_2 = \theta$$

  – b) pontos **em um lado da linha** tem coordenadas tais que

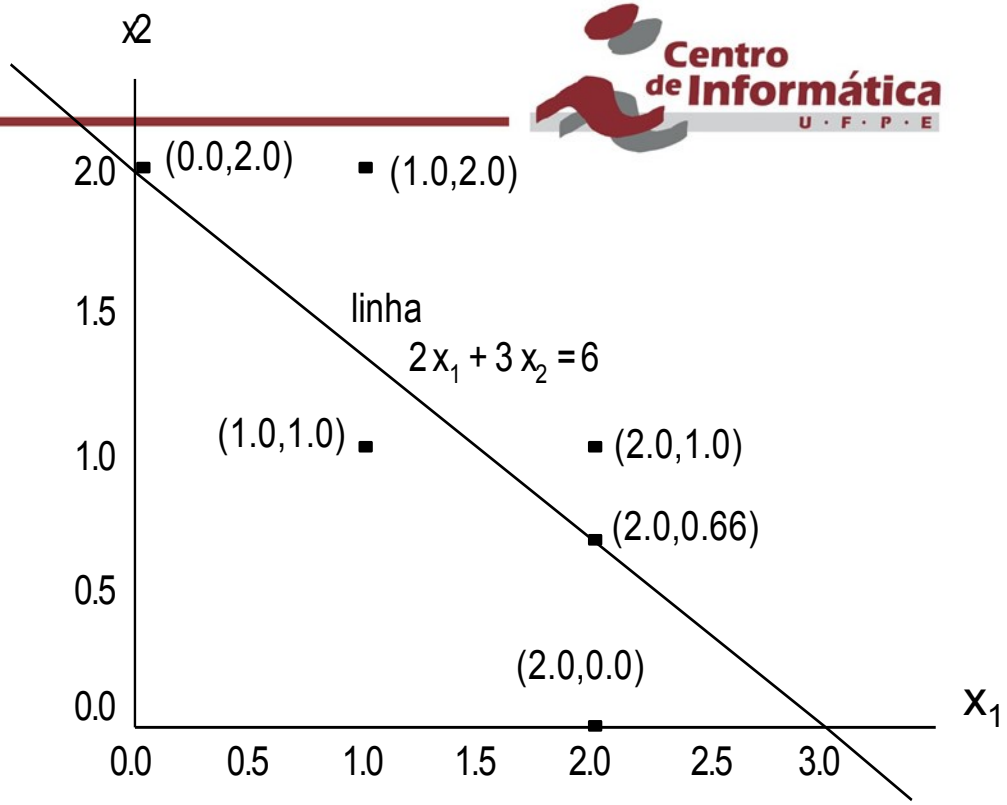
  $$w_1 . x_1 + w_2 . x_2 < \theta$$

  – c) pontos **no outro lado da linha** tem coordenadas tais que

  $$w_1 . x_1 + w_2 . x_2 > \theta .$$

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: $2\ x_1 + 3\ x_2 > 6$

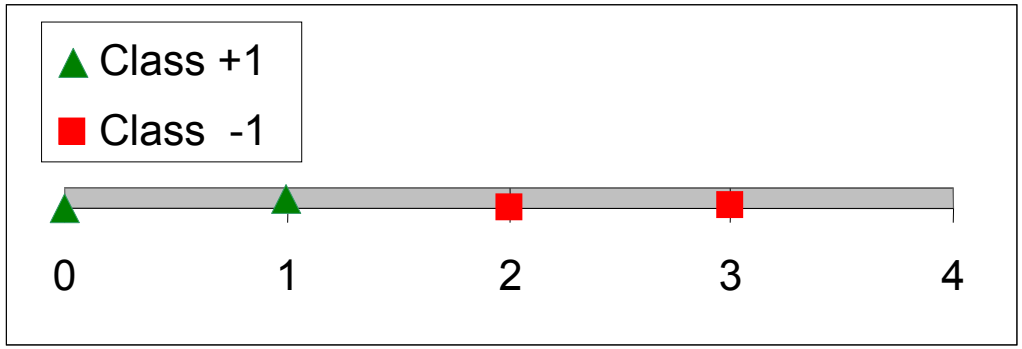
Abaixo: $2\ x_1 + 3\ x_2 < 6$

$\theta = 6$

Exemplo

Vantagens da separabilidade linear

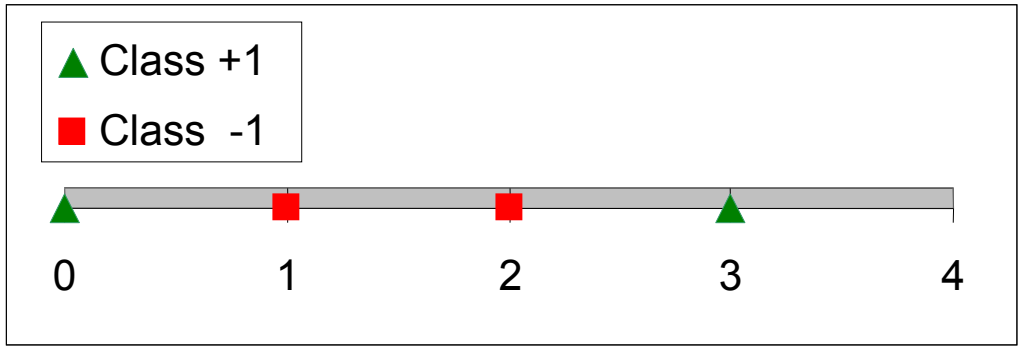
X_1	Class
0	+1
1	+1
2	-1
3	-1



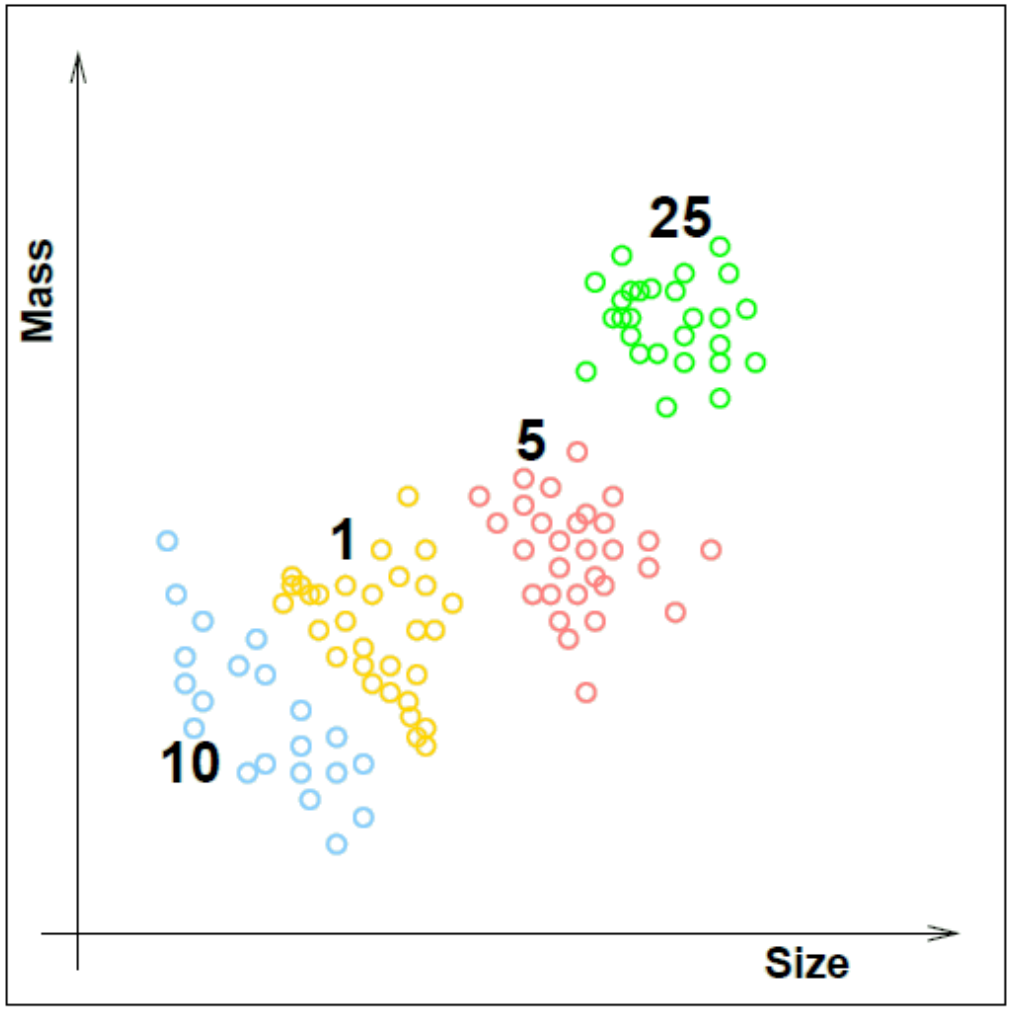
Exemplo

Como separar as duas classes com apenas um ponto?

X_1	Class
0	+1
1	-1
2	-1
3	+1



Exemplo: Reconhecimento de moedas



A simple hypothesis set - the 'perceptron'

For input $\mathbf{x} = (x_1, \cdots, x_d)$ 'attributes of a customer'

$$\text{Approve credit if} \quad \sum_{i=1}^{d} w_i x_i > \text{threshold},$$

$$\text{Deny credit if} \quad \sum_{i=1}^{d} w_i x_i < \text{threshold}.$$

This linear formula $h \in \mathcal{H}$ can be written as

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right)$$

Learning From Data, Prof. Yaser Abu-Mostafa

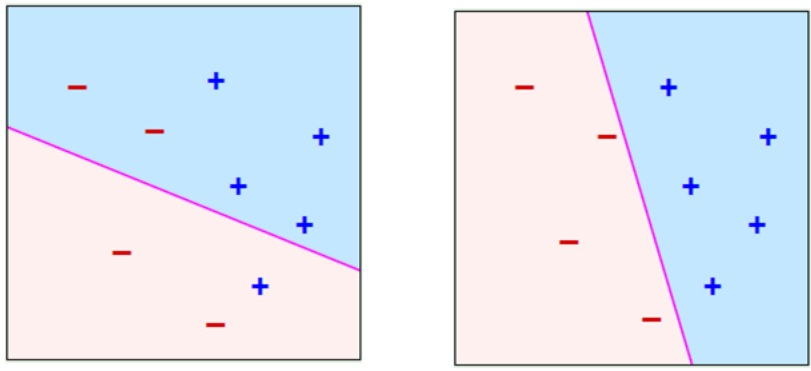
$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i\, x_i\right) + \quad w_0 \quad\right)$$

Introduce an artificial coordinate $x_0 = 1$:

$$h(\mathbf{x}) = \text{sign}\left(\sum_{i=0}^{d} w_i\, x_i\right)$$

In vector form, the perceptron implements

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathsf{T}}\mathbf{x})$$



'linearly separable' data

A simple learning algorithm - PLA

The perceptron implements

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\mathsf{T}}\mathbf{x})$$

Given the training set:

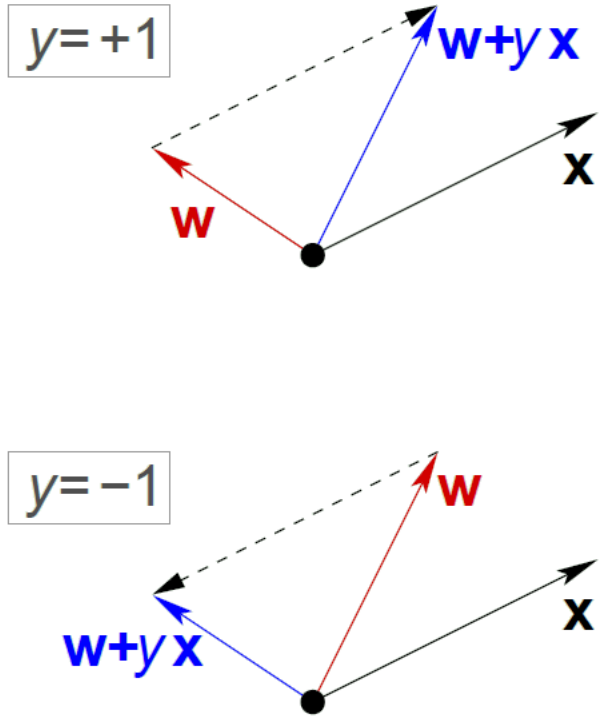
$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

pick a misclassified point:

$$\text{sign}(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n) \neq y_n$$

and update the weight vector:

$$\mathbf{w} \leftarrow \mathbf{w} + y_n\mathbf{x}_n$$



Iterations of PLA

- One iteration of the PLA:

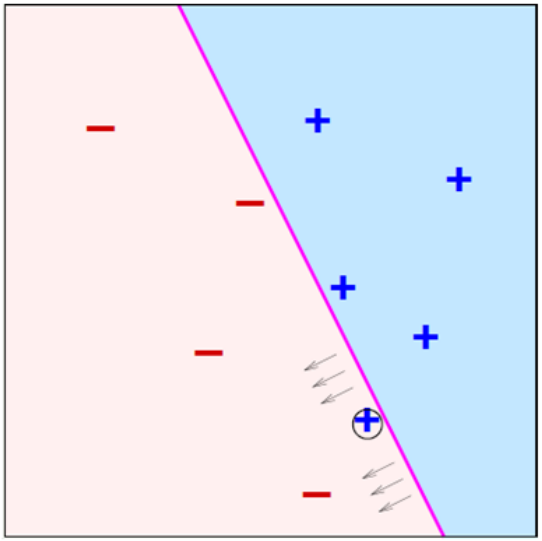
$$\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$$

where $(\mathbf{x}, y)$ is a misclassified training point.

- At iteration $t = 1, 2, 3, \cdots$, pick a misclassified point from

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

and run a PLA iteration on it.



- That's it!

* Bias data?

Introdução às Redes Neurais Artificiais

1° Semestre de 2015

Cleber Zanchettin
UFPE - Universidade Federal de Pernambuco
CIn - Centro de Informática

O que é uma Rede Neural ?

- O trabalho em Redes Neurais Artificiais (RNA), tem sido motivado desde o começo pelo reconhecimento 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.

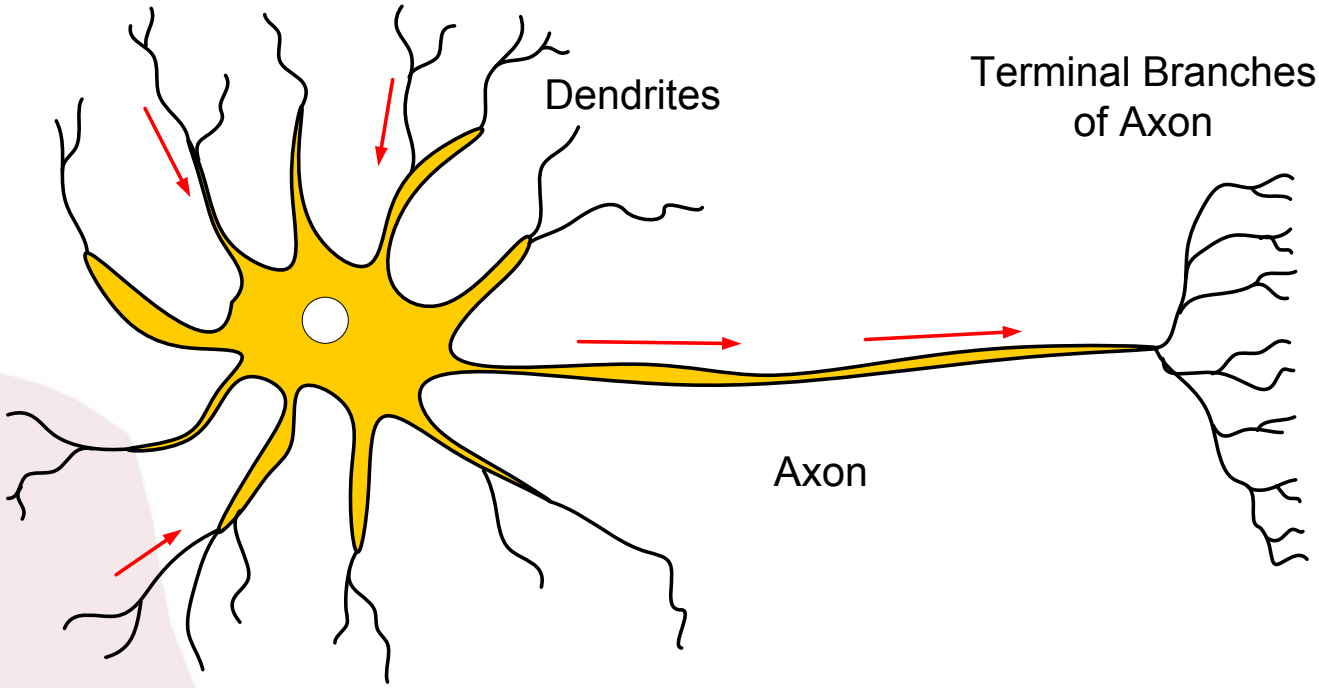
- O reconhecimento perceptivo (exemplo, reconhecer um rosto familiar em uma cena não-familiar) pode ser realizado pelo cérebro em poucos milésimos de segundo.

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 rede neural 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**.
- Esse 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

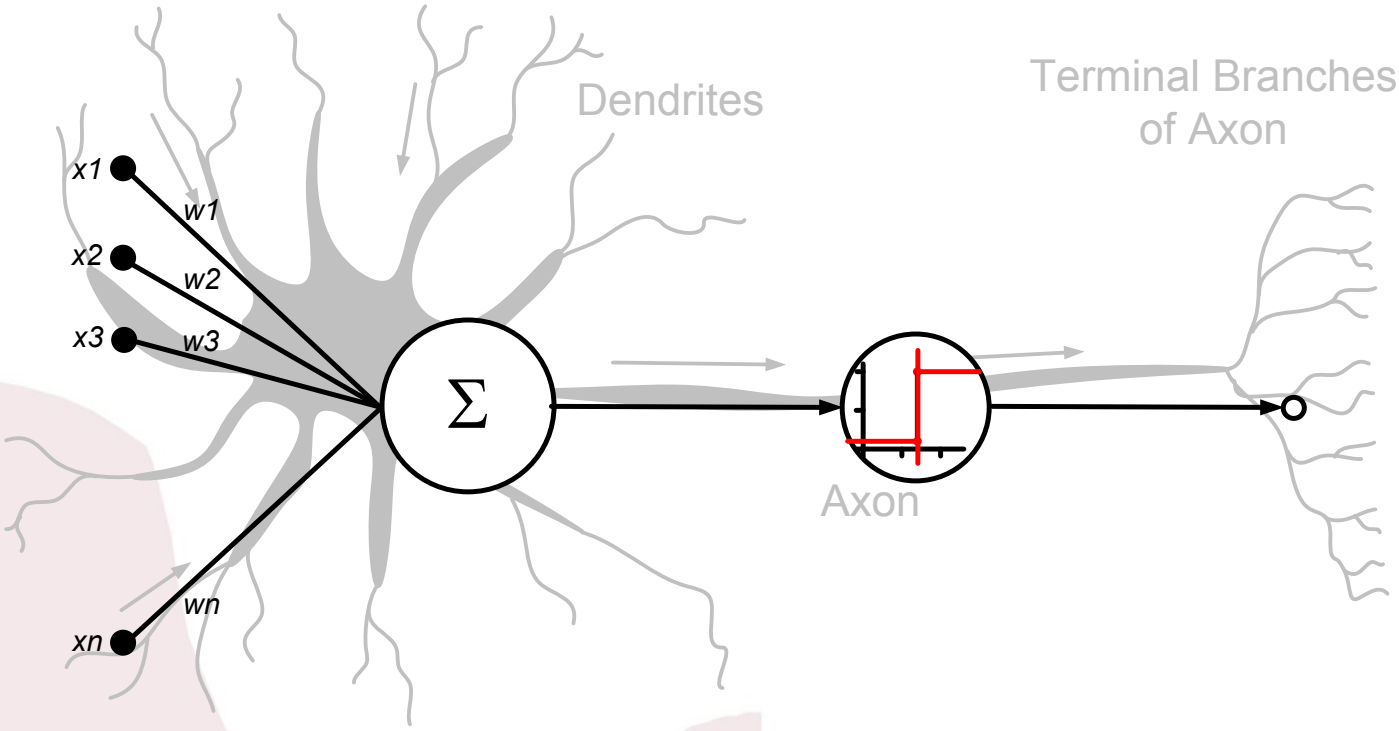
- A **memória** também é resultado de um processo adaptativo das sinapses.

- 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.

O 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.

Potenciais áreas de aplicação

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

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.

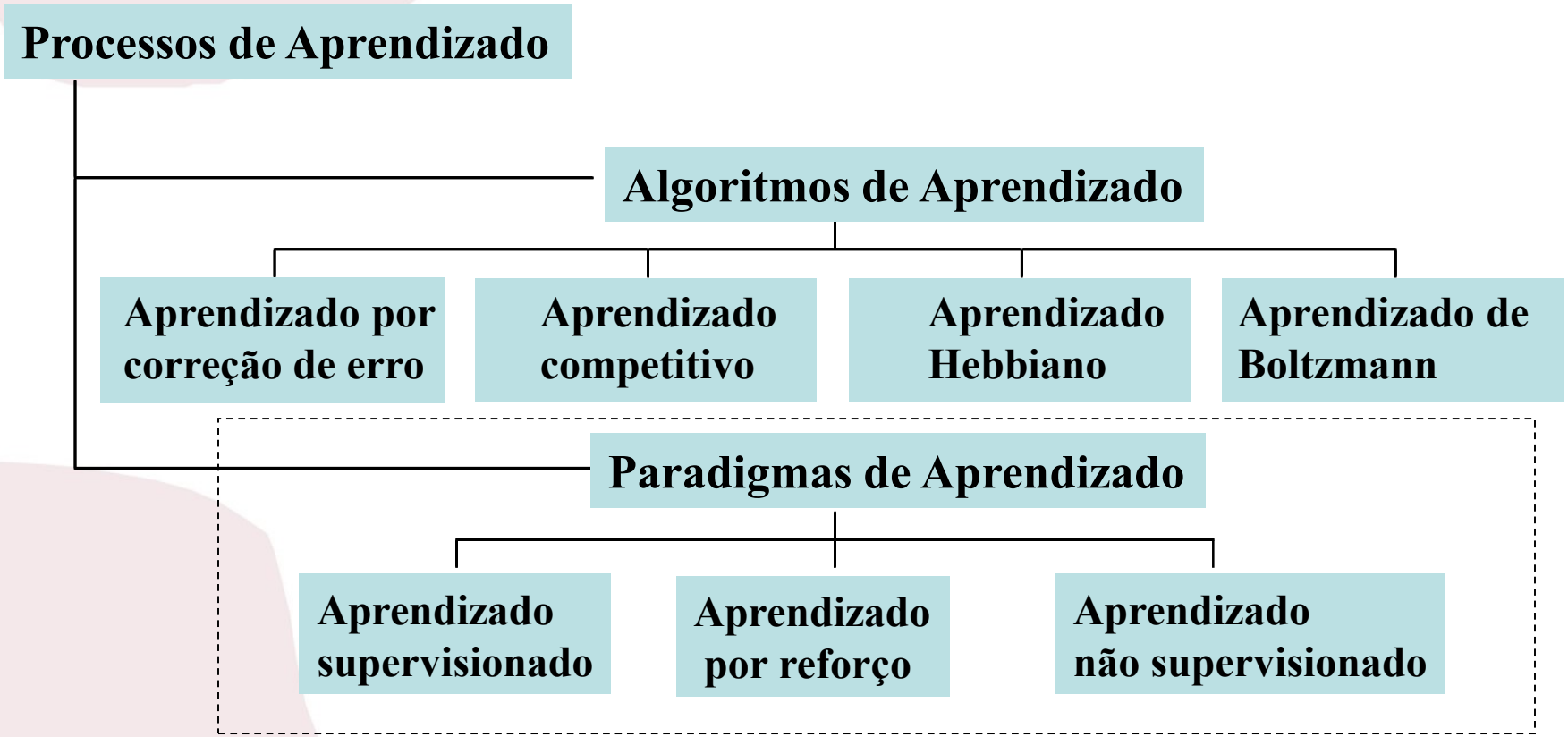
Processo de aprendizagem

O processo de aprendizagem implica na seguinte sequencia 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**

Aprendizado de Máquina



Paradigmas de Aprendizagem

Paradigma é a representação do padrão de modelos a serem seguidos. É um pressuposto filosófico matriz, ou seja, uma teoria, um conhecimento que origina o estudo de um campo científico; uma realização científica com métodos e ***valores*** que são concebidos como modelo; uma referência inicial como base de modelo para estudos e pesquisas.

Os principais paradigmas (modelos) de aprendizagem são:

1) **Supervisionado**

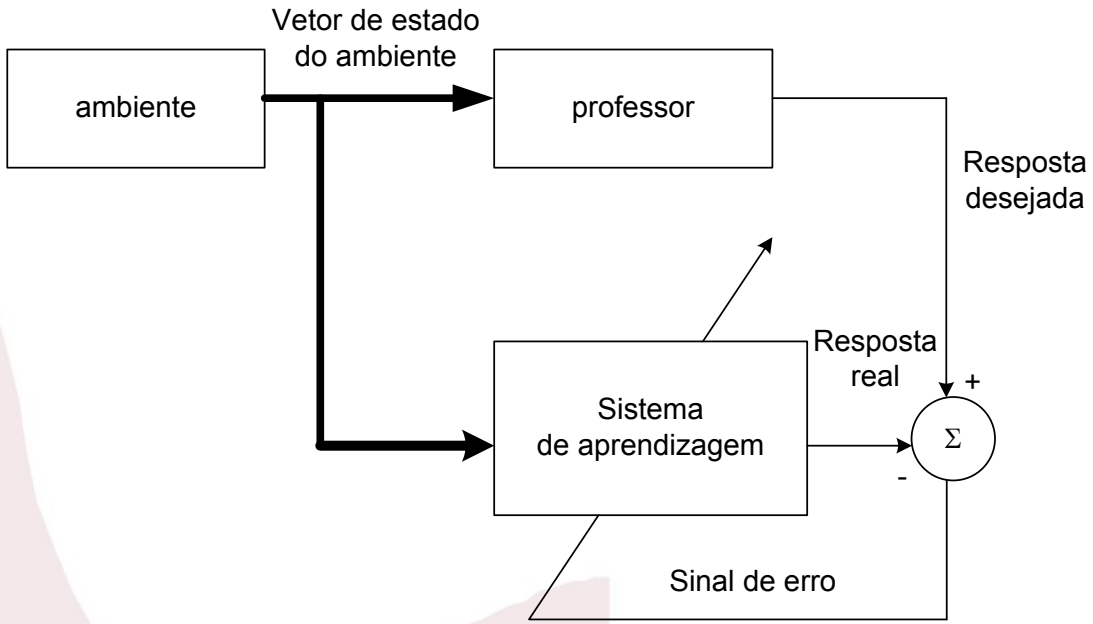
2) **Não-Supervisionado**

3) **Aprendizagem por Reforço**

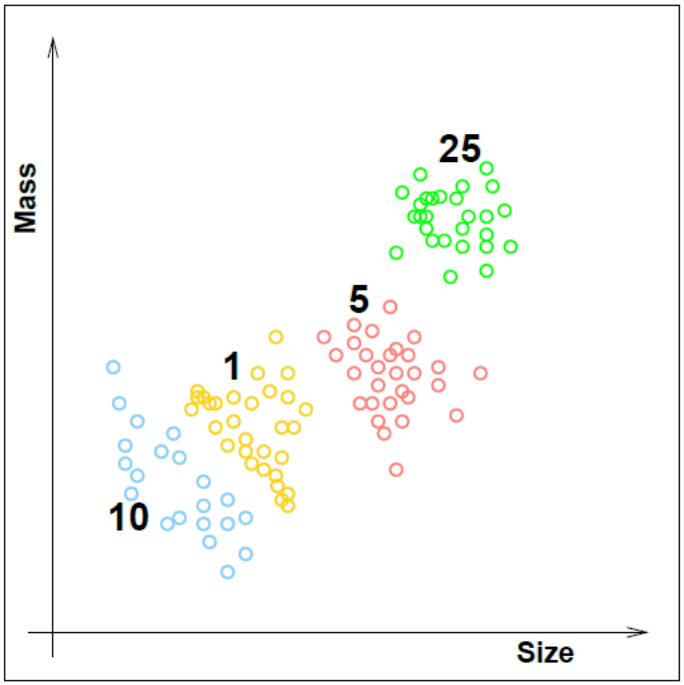
http://www.dee.ufc.br/~arthurp

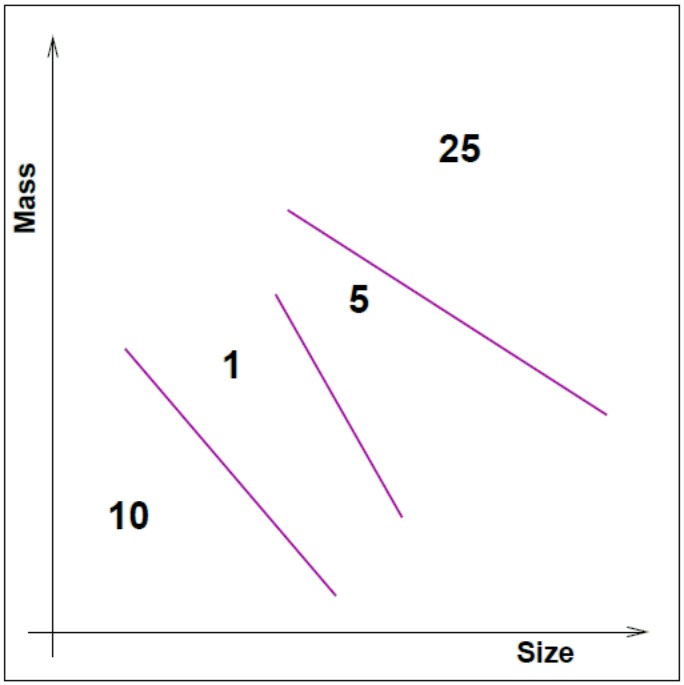
Supervisionado

- Também conhecida com aprendizagem com professor, consiste em que o professor tenha o conhecimento do ambiente, e fornece o conjunto de exemplos de entrada-resposta desejada.
- Com esse conjunto, o treinamento é feito usando a regra de aprendizagem por correção de erro.



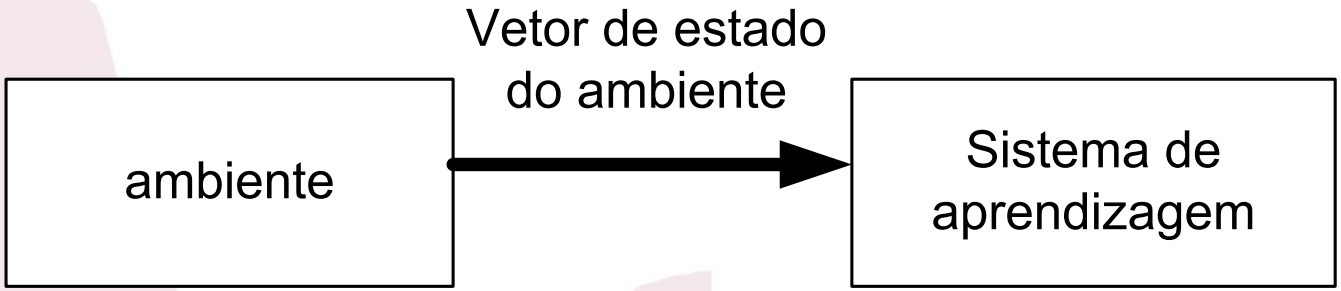
Exemplo: Reconhecimento de moedas



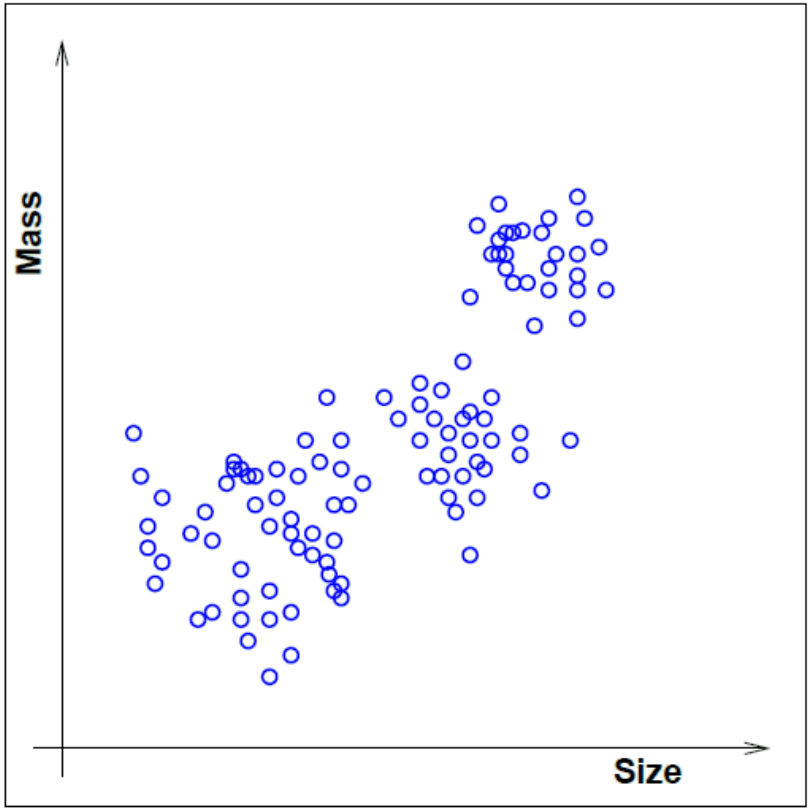


Não-supervisionado

- Neste caso nao há um professor para supervisionar o processo de aprendizagem. Isso significa que não há exemplos rotulados da função a ser aprendida pela rede.
- Nesse modelo, também conhecido como auto-organizado, são dadas condições para realizar uma medida da representação que a rede deve aprender, e os parâmetros livres da rede são otimizados em relação a essa medida.
- Para a realização da aprendizagem não-supervisionada pode-se utilizar a regra de aprendizagem competitiva.

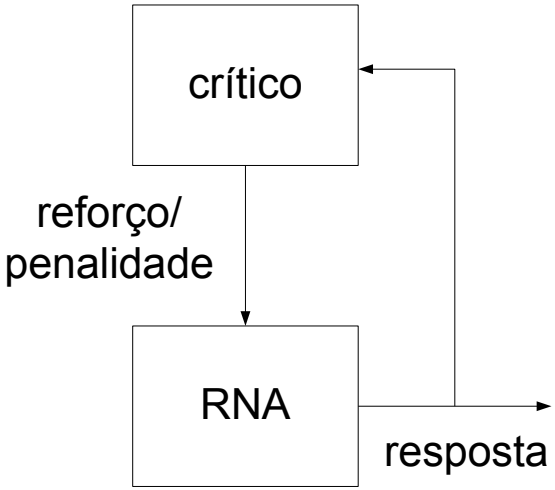


Instead of (input,correct output), we get (input, ?)



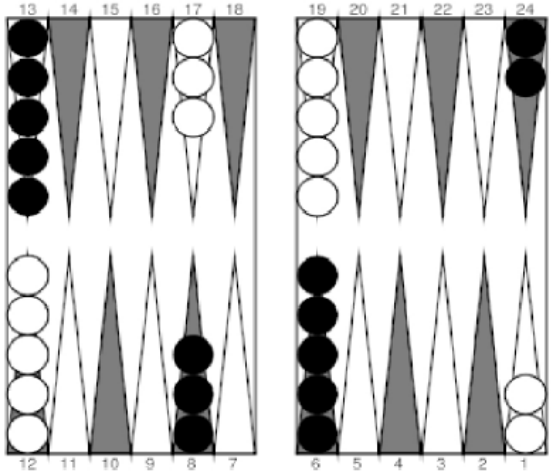
Aprendizagem por reforço

- Pode ser visto como caso particular de aprendizagem supervisionada.
- A principal diferença entre o aprendizado supervisionado e o aprendizado por reforço é a medida de desempenho usada em cada um deles.
- No aprendizado supervisionado, a medida de desempenho é baseada no conjunto de respostas desejadas usando um critério de erro conhecido, enquanto que no aprendizado por reforço a única informação fornecida à rede é se uma determinada saída está correta ou não.
- A idéia básica tem origem em estudos experimentais sobre aprendizado dos animais. Quanto maior a satisfação obtida com uma certa experiência em um animal, maiores as chances dele aprender.



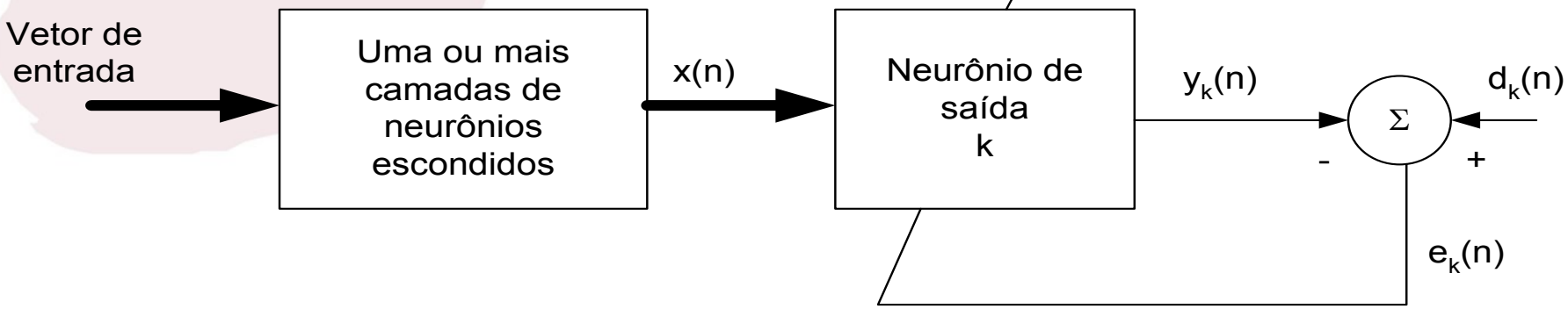
Instead of (input,correct output),

we get (input,*some* output,grade for this output)



The world champion was a neural network!

Aprendizagem por correção de erro



O sinal de saida do neurônio k é representado por $y_k(n)$, e a resposta desejada por $d_k(n)$, produzido um sinal de erro:

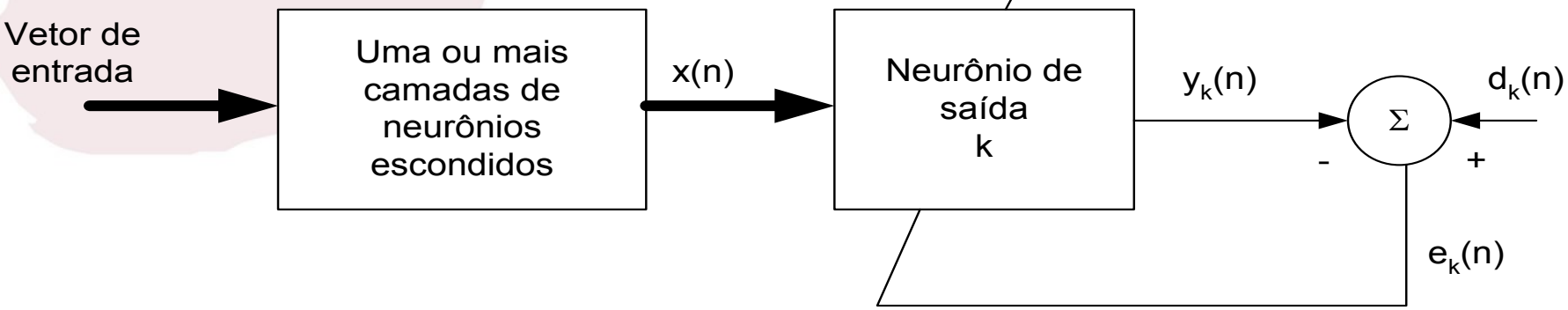
$$e_k(n) = d_k(n) - y_k(n)$$

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:

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

Aprendizagem baseada em memória

- As experiências são armazenadas em uma grande memória de exemplos de entrada-saída classificados corretamente:

$$\{(\mathbf{x}_i, d_i)\}\ i = 1...n,$$

onde $\mathbf{x}_i$ representa um vetor de entrada e d_i representa a resposta desejada correspondente.

- Num problema de classificação de padrões binários, por exemplo, há duas classes a serem consideradas, C1 e C2, e a resposta desejada d_i assume valor 0 para a classe C1 e valor 1 para a classe C2.

- Na classificação de um vetor de teste, o algoritmo busca os dados de treinamento em uma vizinhança local deste vetor.

- Todos os algoritmos de aprendizagem baseada em memória envolvem dois aspectos essenciais:
  - o critério utilizado para definir a vizinhança local do vetor de teste, e
  - a regra de aprendizagem aplicada aos exemplos de treinamento na vizinhança local do vetor de teste.

Aprendizagem Hebiana

- O postulado de aprendizado de Hebb é a mais antiga e mais famosa de todas as regras de aprendizagem, e é baseado no texto a seguir, contido no livro de Hebb (1949), *The Organization of Behavior*:
  - O peso de uma conexão sináptica deve ser ajustado se houver sincronismo entre as atividades: do neurônio pré-sináptico e do neurônio pós-sináptico.
  - Em termos matemáticos

$$\Delta w_{ij}(t) = \eta y_i(t) x_j(t)$$

que significa que a mudança do peso sináptico $\Delta w_{ij}(t)$ é proporcional ao valor do neurônio pós-sináptico $y_i(t)$ e ao valor do neurônio pré-sinático $x_j(t)$ multiplicado pelo fator de aprendizado η .

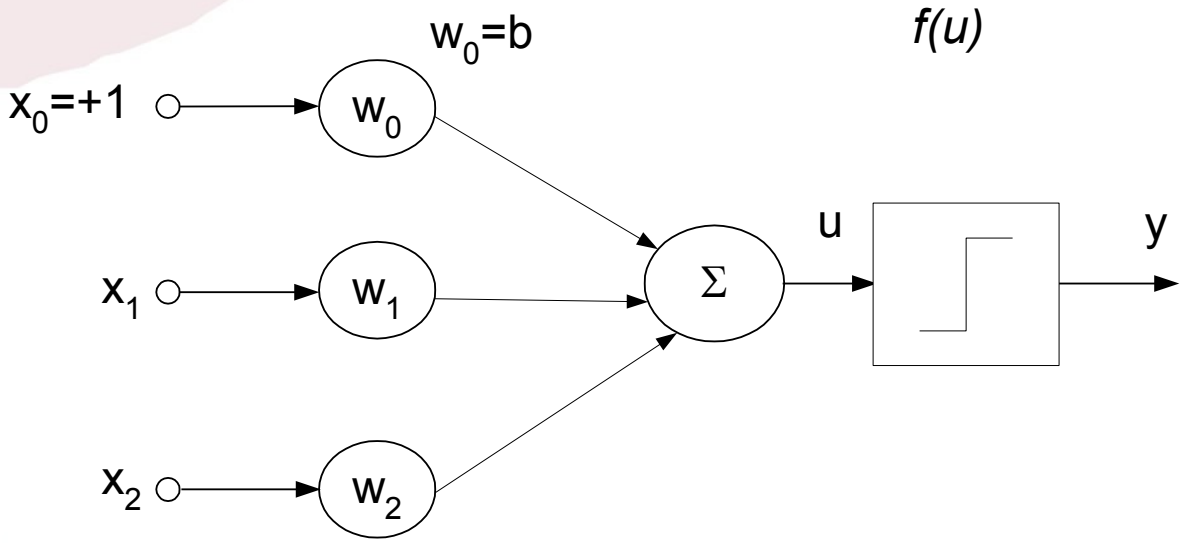
Redes Multi-Layer Perceptron - BP

1° Semestre de 2015

Cleber Zanchettin
UFPE - Universidade Federal de Pernambuco
CIn - Centro de Informática

Motivação

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:

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

O 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.

O papel do bias

- Exemplo
  - Suponha uma rede neural cuja função de ativação é a **step bipolar**, isto é, f(u) = 1, se u ≥ θ e f(u) = -1, caso contrário.
  - Sabemos que $y = x_1*w_1 + x_2*w_2 + \ldots + x_n*w_n$, então podemos resumir nosso problema à inequação:

$$x_1*w_1 + x_2*w_2 + \ldots + x_n*w_n \geq \theta$$

  - Que, sendo verdadeira, fará **f(u) ter valor 1**.

O papel do bias

- Então, para que nossa rede neural seja corretamente ajustada, precisaríamos ajustar não somente os pesos, mas também o ***threshold θ***, tarefa não muito simples, já que precisaríamos ter alguma fórmula de ajuste do *threshold*.

- Entretanto, podemos fazer algumas alterações:

$$x_1*w_1 + x_2*w_2 + \ldots + x_n*w_n \geq \theta$$

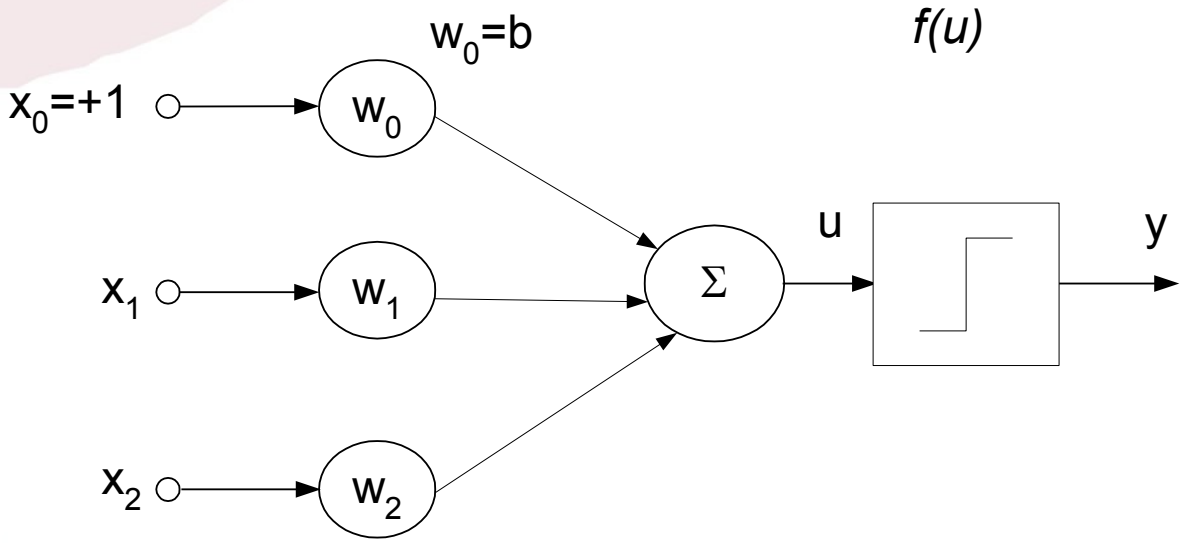
$$- \theta + x_1*w_1 + x_2*w_2 + \ldots + x_n*w_n \geq 0$$

$$1*(-\theta) + x_1*w_1 + x_2*w_2 + \ldots + x_n*w_n \geq 0$$

- Agora, podemos considerar o primeiro termo 1*(-θ) = x0*w0, isto é, o valor de entrada sempre 1, um peso cujo valor poderá ser ajustado da **mesma forma** que ajustamos os demais pesos, o que seria o mesmo que ajustar o *threshold*.

Voltando ao 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:

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

Separação Linear

- Sabe-se que se formarmos uma **combinação linear de duas variáveis**, e igualá-la a um número, então os pontos no espaço bidimensional podem ser divididos em três categorias:

  – a) pontos **pertencentes à linha** com coordenadas tais que

  $$w_1 . x_1 + w_2 . x_2 = \theta$$

  – b) pontos **em um lado da linha** tem coordenadas tais que

  $$w_1 . x_1 + w_2 . x_2 < \theta$$

  – c) pontos **no outro lado da linha** tem coordenadas tais que

  $$w_1 . x_1 + w_2 . x_2 > \theta .$$

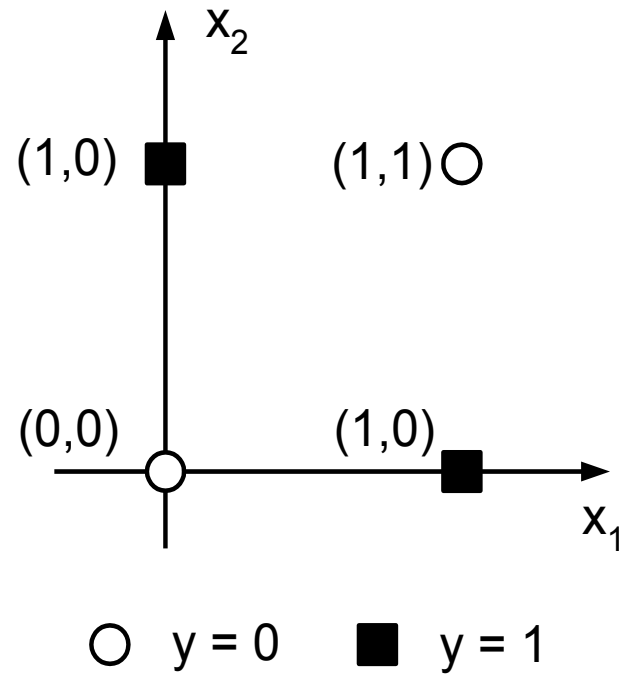
Nota: **bias** $b = -\theta$, pois,

$$y = f(w_1 x_1 + w_2 x_2 + w_0), \quad onde \quad w_0 = b$$

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).

Problema do XOR (ou-exclusivo)

Nenhuma combinação linear ou linha reta pode ser traçada tal que as entradas (0, 0) e (1, 1) fiquem numa categoria e (0, 1) e (1, 0) numa outra categoria, conforme **demonstração**:

Algebricamente tem-se:

$$w_1.0 + w_2.1 \geq \theta \quad \text{para o ponto} \quad (0,1), \quad \text{e}$$

$$w_1.1 + w_2.0 \geq \theta \quad \text{para o ponto} \quad (1,0).$$

significando que os **dois pontos ficam num lado da linha L1**.

Mas, somando as duas inequações tem-se:

$$w_1.1 + w_2.1 \geq \theta$$

o que significa que **(1,1) estaria no mesmo lado da linha L1**, o que é indesejável.

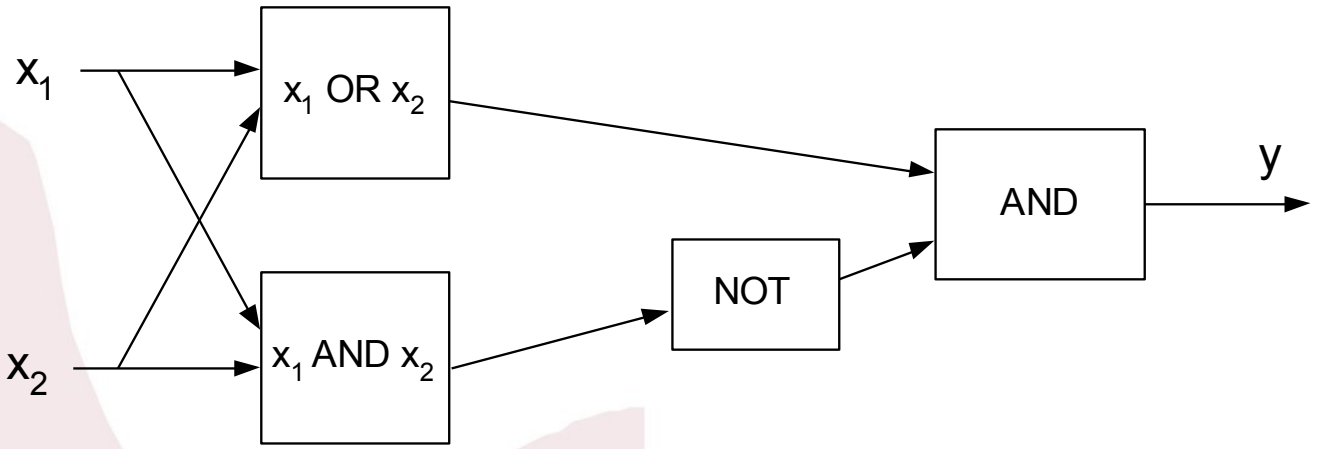
Perceptrons Multicamadas

A função XOR está além da capacidade de um Perceptron simples.

Contudo, um **Perceptron simples pode implementar funções lógicas elementares**: AND, OR e NOT.

Assim, se uma função pode ser expressa como uma **combinação dessas funções lógicas elementares**, então essa função pode ser implementada usando mais neurônios.

Por exemplo, XOR pode ser expressa por: $(x_1$ or $x_2)$ and (not $(x_1$ and $x_2))$.

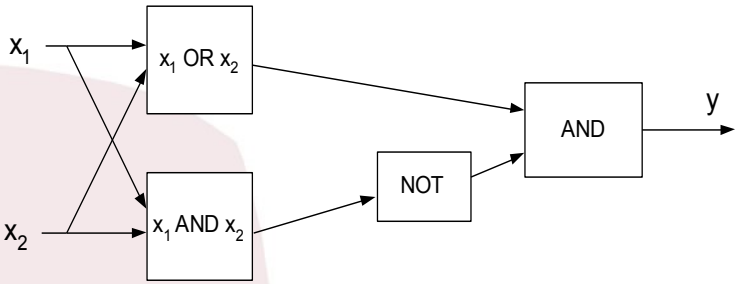


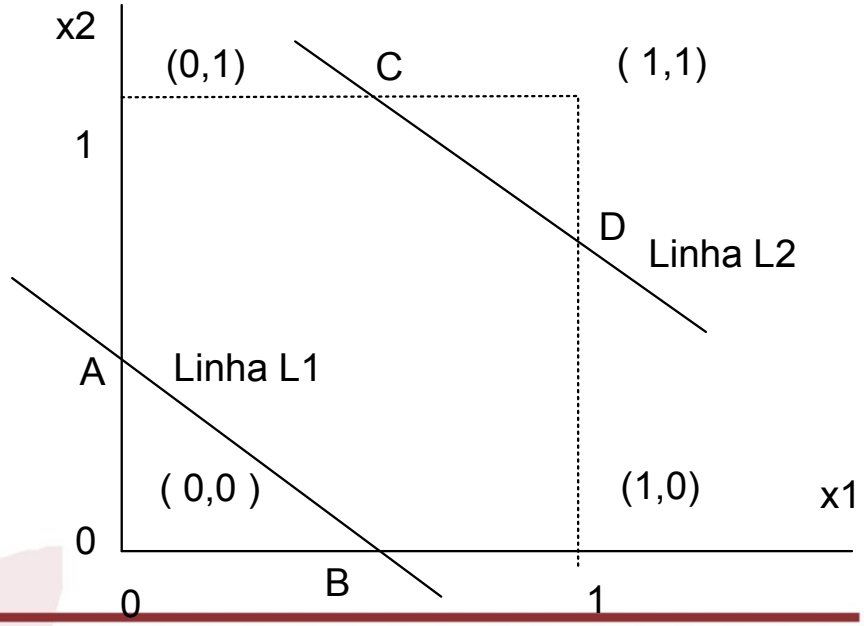
Separação linear para XOR de duas camadas

Em termos de separação linear (demarcação), isso equivale ao traçado de **duas linhas**.

A parte **acima** da linha **L1** corresponde à função **OR**, e a parte **abaixo** da linha **L2** corresponde à função **NOT AND**.

A área **entre as duas linhas** corresponde à função **XOR**, o que corresponde à área ABCD, que contem os pontos (0,1) e (1,0).





Redes Multi-Layer Perceptron

O perceptron de múltiplas camadas (Multi-Layer Perceptron – MLP) é uma rede do tipo **Perceptron** com **pelo menos uma** camada intermediária.

Apesar desse tipo de rede apresentar soluções para funções linearmente não-separáveis, como visto anteriormente, torna-se **necessário** um **algoritmo de treinamento** capaz de definir de forma **automática** os pesos.

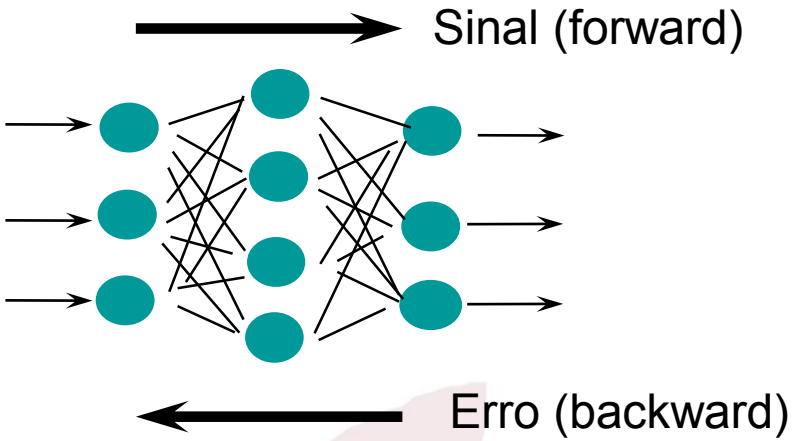
O algoritmo desta rede é uma **generalização** da regra delta de Widrow & Hoff para o **treinamento da Adaline**, e denomina-se ***Backpropagation***.

Redes Multi-Layer Perceptron

Este algoritmo consiste basicamente dos passos:

- **propagação positiva** do sinal: durante este processo todos os pesos são mantidos fixos; e
- **retropropagação** do erro: durante este processo os pesos da rede são ajustados baseados na **regra de correção de erro**.

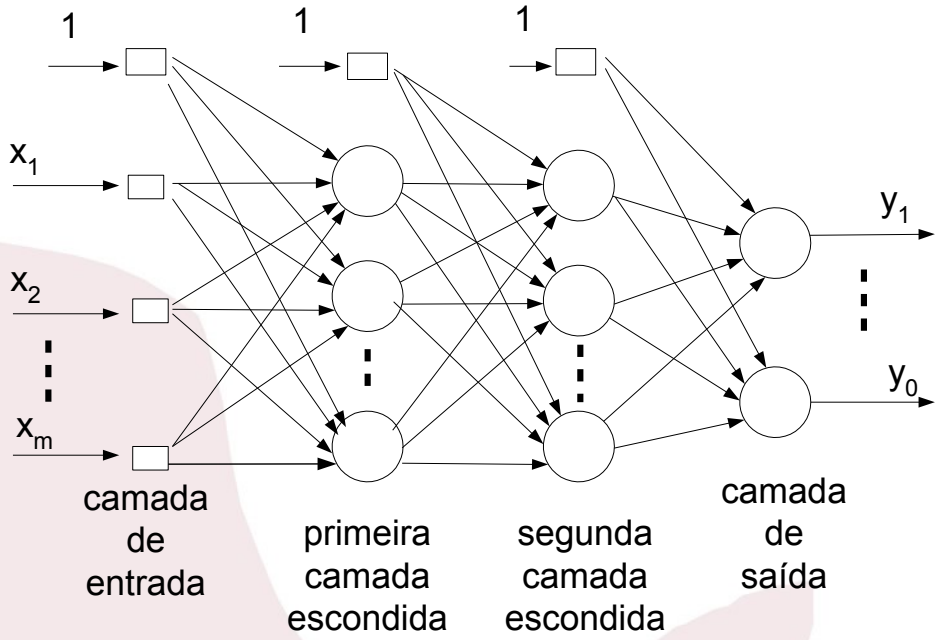
Como a propagação do erro é calculado no sentido inverso do sinal, o algoritmo é denominado de **retropropagação** do erro.

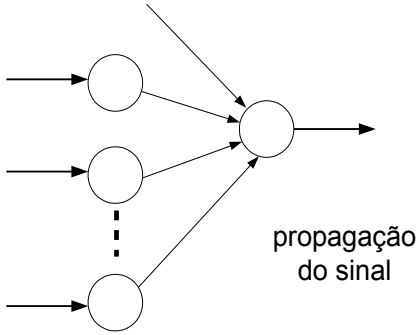


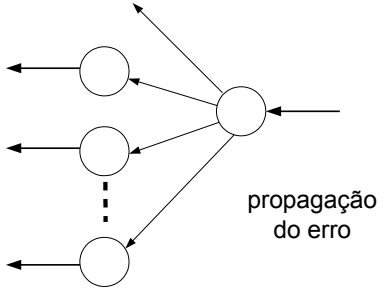
Características da rede

Uma **rede MLP típica** possui as seguintes características:

- os neurônios possuem uma **função de ativação não-linear**, **diferenciável**, do tipo **sigmoidal** (por exemplo, função logística e tangente-hiperbólica);
- a rede possui **uma ou mais camadas** intermediárias; e
- a rede possui uma **alta conectividade**.







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

Treinamento de redes MLP

- Treinamento estático
  - MLPs com formatos e tamanhos diferentes podem utilizar mesma regra de aprendizado
    - Topologias diferentes podem resolver o mesmo problema
  - Regra mais utilizada: Backpropagation

Backpropagation

- Com o desenvolvimento do algoritmo de treinamento Backpropagation, por Rumelhart, Hinton e Williams em 1986, precedido por **propostas semelhantes** ocorridas nos anos 70 e 80.
  - Em 1974, Werbos descobriu o algoritmo enquanto desenvolvia sua tese de doutorado em estatística e o chamou de **Algoritmo de Realimentação Dinâmica**.
  - Parker em 1982 redescobriu o algoritmo e chamou-o de **Algoritmo de Aprendizado Lógico.**
- Mostrou-se que é possível treinar **eficientemente** redes com camadas intermediárias.

Backpropagation

- Nessas redes, cada camada tem uma **função específica:**

  – A **camada de saída** recebe os estímulos da camada intermediária e **constrói o padrão** que será a resposta.

  – As **camadas intermediárias** funcionam como **extratoras de características**, seus pesos são uma codificação de características apresentadas nos padrões de entrada, e

  – Permitem que a rede crie sua **própria representação**, mais rica e complexa, do problema.

Treinamento

A **generalização da regra delta** de Widrow &Hoff é usada para ajustar os pesos e bias da rede de forma a **minimizar o erro** entre a saída da rede e a saída desejada.

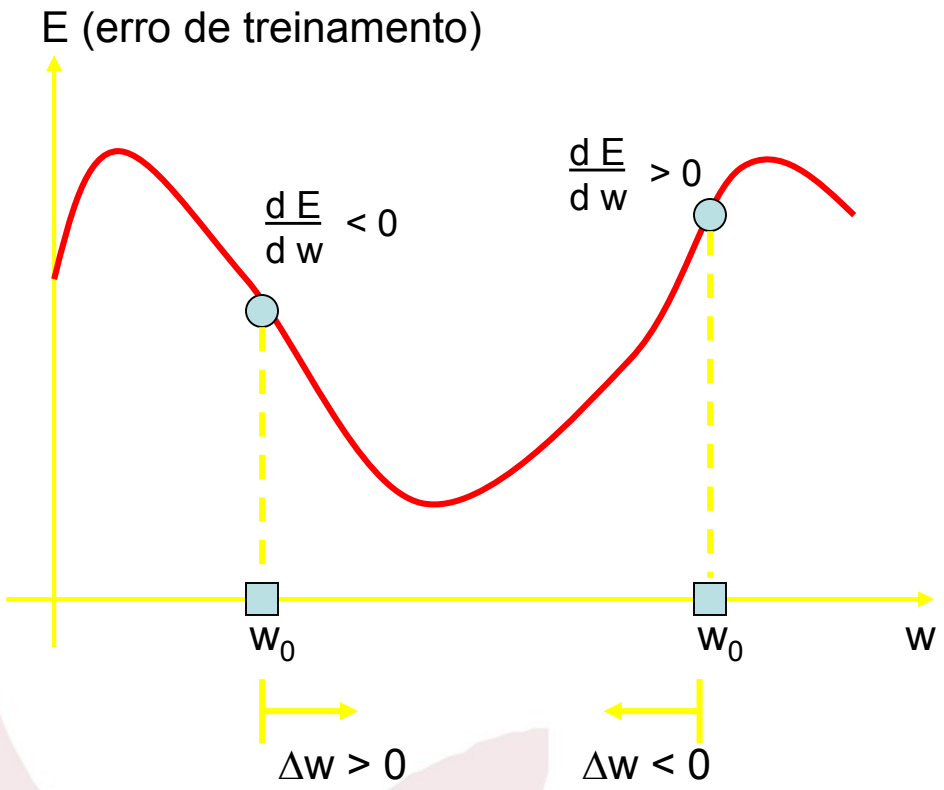
Para isso é calculado o **gradiente da função de erro** em relação aos pesos e bias, pois atualizando os pesos e bias **na direção oposta** a esse gradiente **minimiza-se o erro**.

Como o erro é calculado apenas para a camada de saída, **o algoritmo** de backpropagation responde **como determinar** a influência do erro nas **camadas intermediárias** da rede.

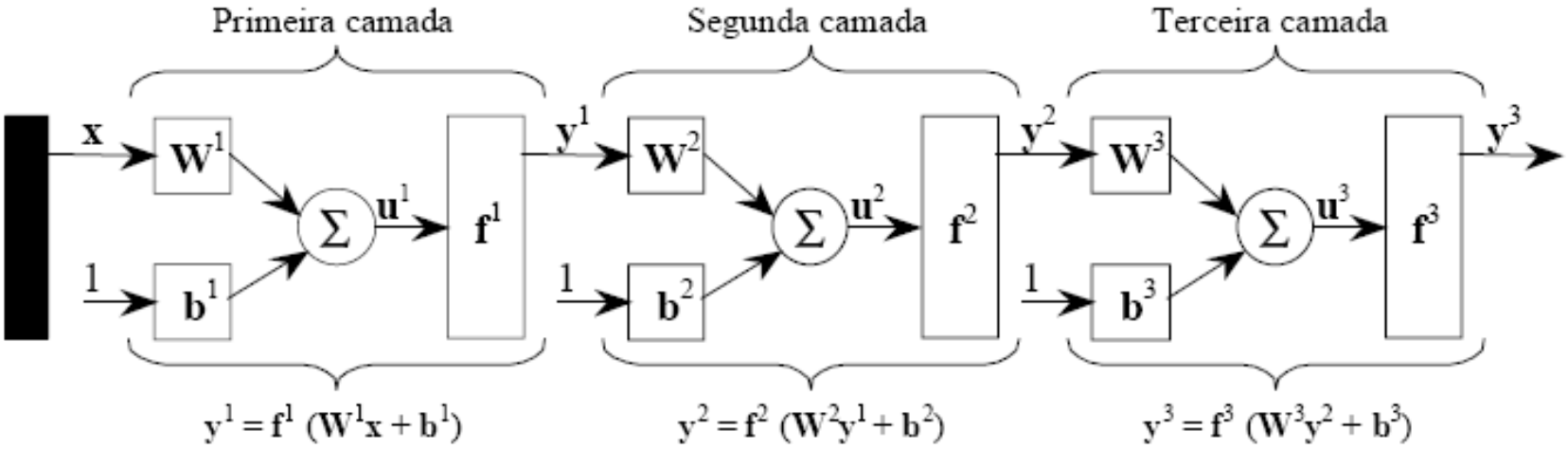
$$w_{ij}^{m}(t+1) = w_{ij}^{m}(t) - \alpha \frac{\partial \Im(t)}{\partial w_{ij}^{m}}, \quad b_{i}^{m}(t+1) = b_{i}^{m}(t) - \alpha \frac{\partial \Im(t)}{\partial b_{i}^{m}}$$

Treinamento

– Interpretação gráfica: Busca do mínimo global.



Cálculo da saída



No algoritmo de Backpropagation **a saída de cada camada** da rede é calculada por:

$$\mathbf{y}^{m+1} = f^{m+1}(\mathbf{W}^{m+1}\mathbf{y}^{m} + \mathbf{b}^{m+1}), \quad \text{para} \quad m = 0,1,\ldots,M-1$$

onde **M** é o **número de camadas** da rede,

$\mathbf{y}^0 = \mathbf{x}$, a entrada da primeira camada é o **vetor de entradas**, e

$\mathbf{y} = \mathbf{y}^M$, a saída da rede é a **saída da última** camada.

Algoritmo MLP-BP

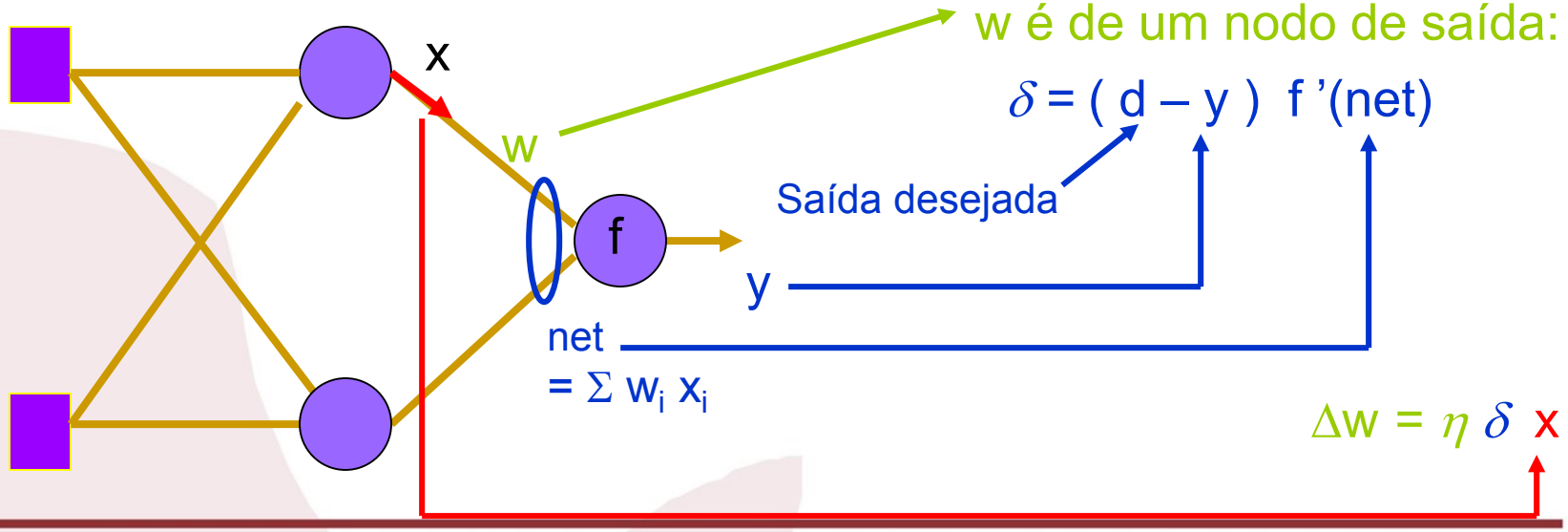
- Inicialize aleatoriamente pesos e bias;
- Enquanto critério de parada não satisfeito:
  - Para cada padrão de entrada:
    - Compute a soma ponderada dos nodos escondidos;
    - Compute a resposta dos nodos escondidos;
    - Modifique os pesos que chegam à camada de saída;
    - Modifique os pesos que chegam a cada uma das camadas escondidas;
  - Fim-Para
- Fim-Enquanto

Backpropagation - Resumindo

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

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

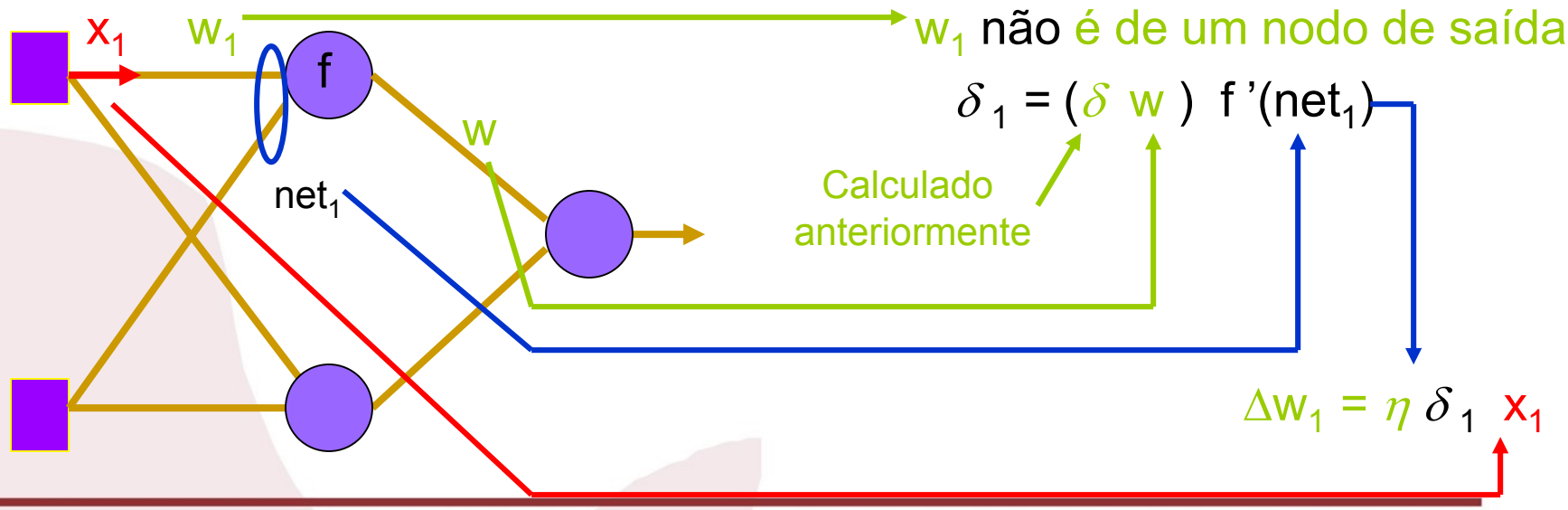
$$onde \quad \delta_j(t) = \begin{cases} (d_j - y_j)\, f'(net_j), & se\ for\ nodo\ de\ saída \\ (\sum_l \delta_l\, w_{lj})\, f'(net_j), & caso\ contrário \end{cases}$$



Backpropagation - Resumindo

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

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

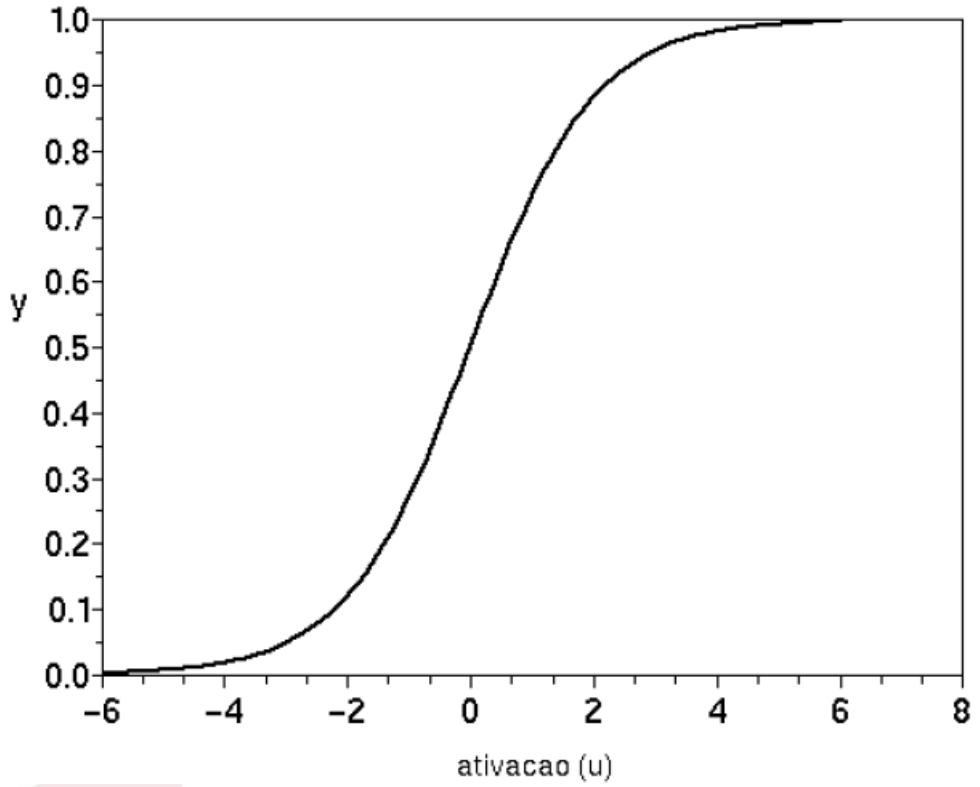


Exemplos de função de ativação

Função de ativação ***Sigmóide Logística***

$$y_i(t) = \frac{1}{1+\exp(-u_i(t))}$$

$$y_i(t) \in (0,1)$$

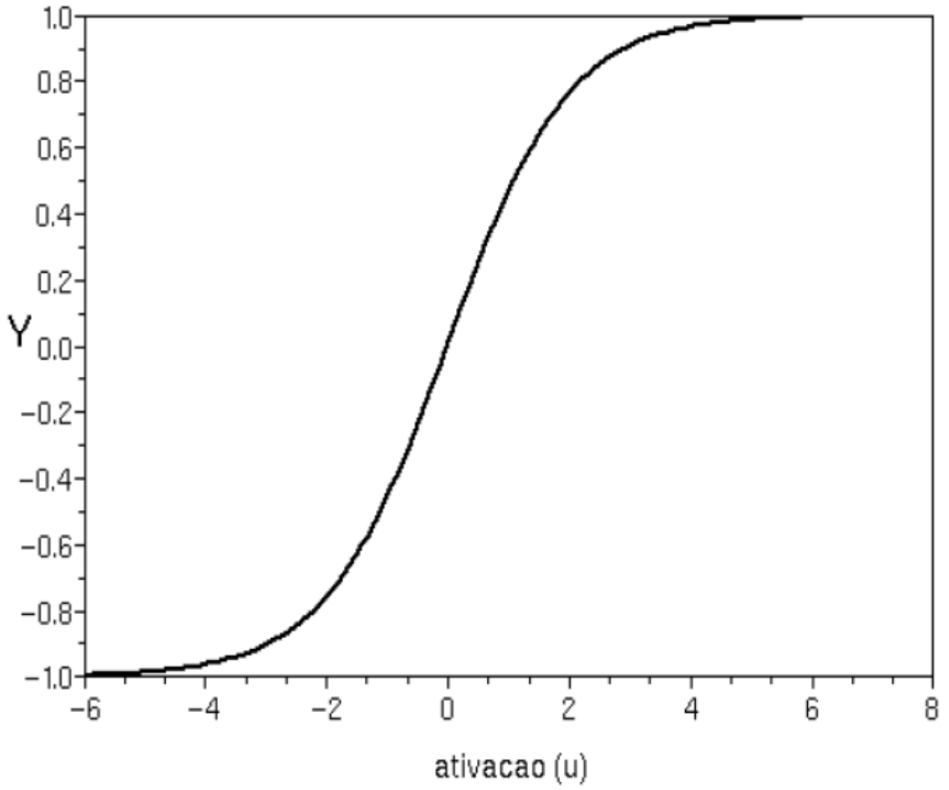


Exemplos de função de ativação

Função de ativação ***Tangente Hiperbólica***

$$y_i(t) = \frac{1 - \exp(-u_i(t))}{1 + \exp(-u_i(t))}$$

$$y_i(t) \in (-1,1)$$



Exemplos de função de ativação

Vantagens:

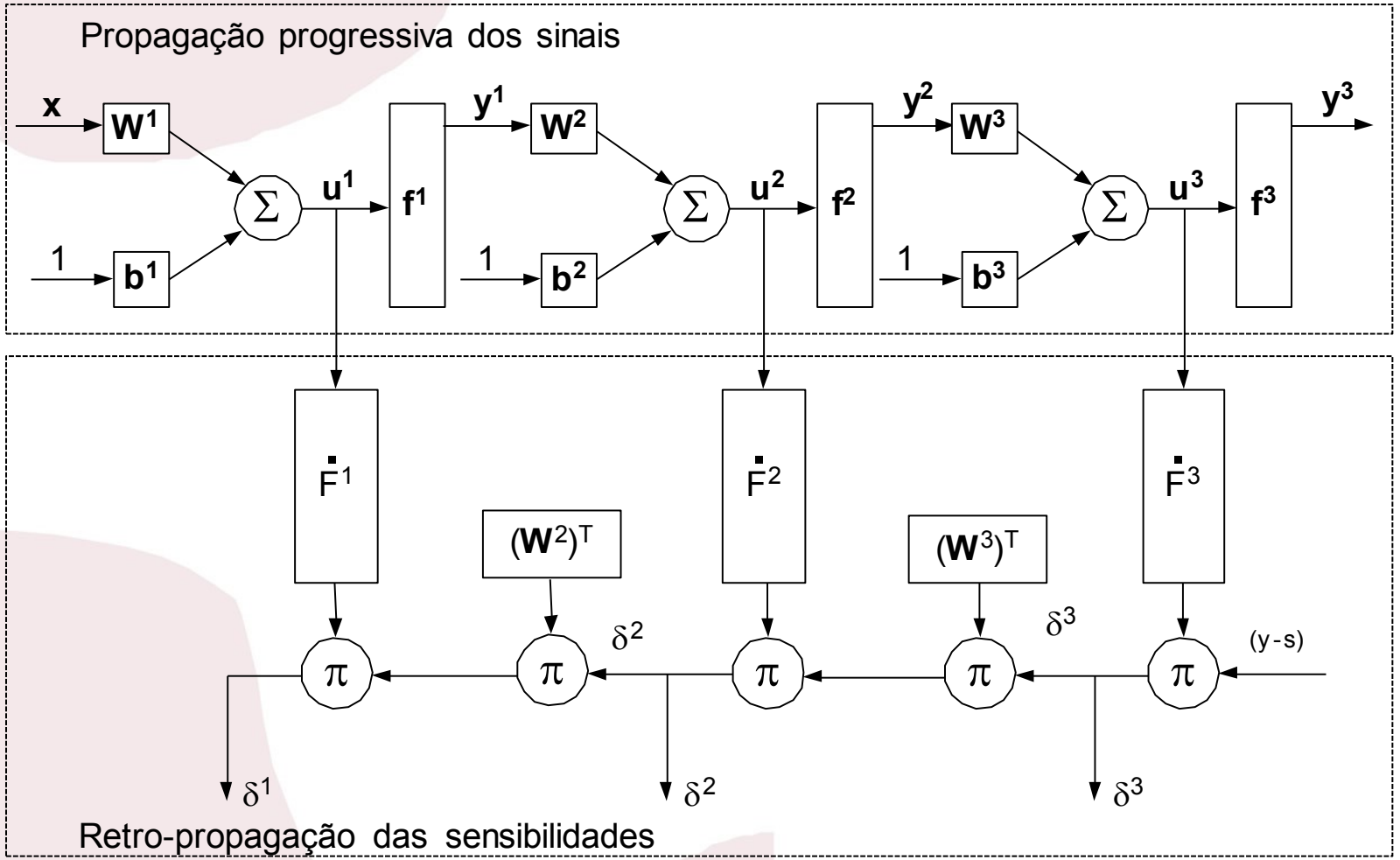
(1) Derivadas fáceis de calcular.
(2) Não-linearidade fraca (trecho central é quase linear)
(3) Interpretação da saída como taxa média de disparo (*mean firing rate*), em vez de simplesmente indicar se o neurônio está ou não ativado (ON-OFF).

Desvantagens:

(1) Elevado custo computacional para implementação em sistemas embarcados devido à presença da função EXP.

$$\exp(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Ilustração gráfica



Classes dos algoritmos de treinamento

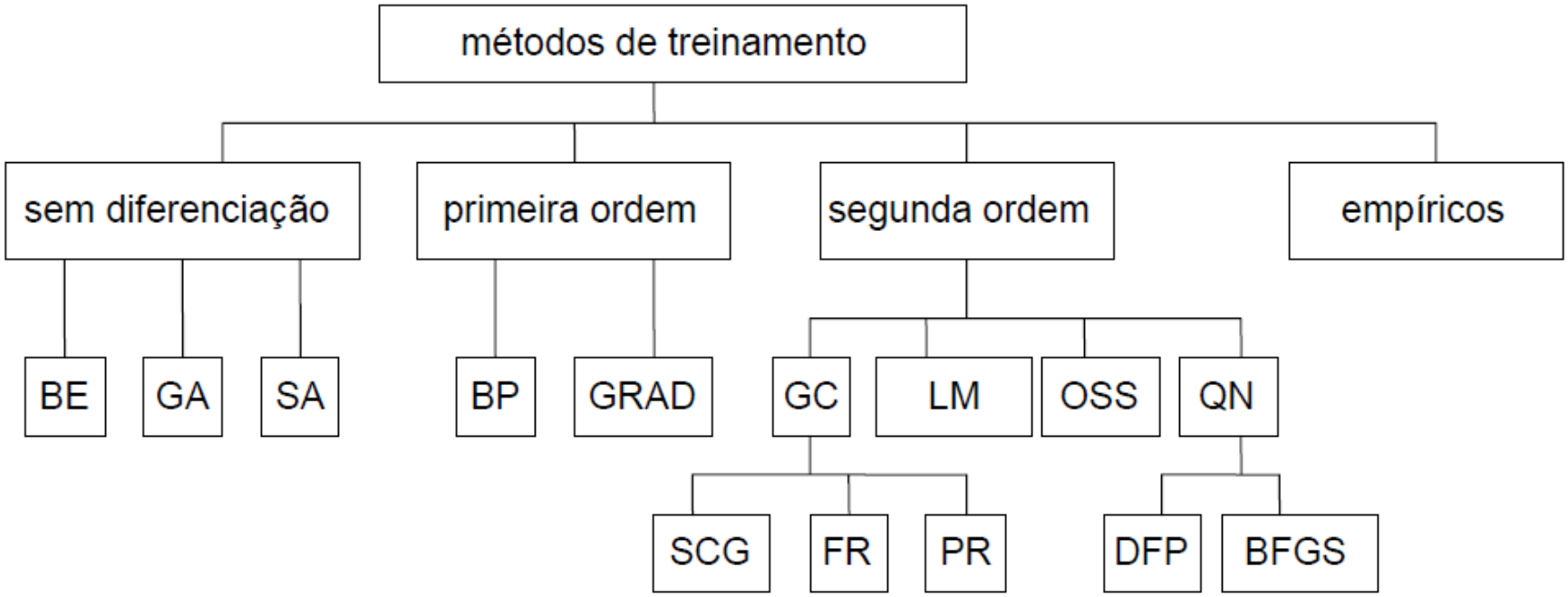
Os algoritmos de treinamento de *Perceptrons* multicamadas podem ser classificados como:

Sem diferenciação: algoritmos que não requerem derivação, apenas avaliação da função em diferentes pontos do espaço. São chamados métodos sem diferenciação.

Primeira ordem: faz uso da derivada primeira a ser minimizada. São chamados métodos de primeira ordem.

Segunda ordem: utilizam informações sobre a derivada segunda.

Uma outra classe consiste no ajuste de pesos usando o método de tentativas e erros, e é denominado método empírico.



BE-Busca Exaustiva
GA-Genetic Algorithms
SA-Simulated Annealing

GC-Gradiente Conjugado

BP –BackPropagation
GRAD- Gradiente

SCG-Scaled CG
FR-Fletcher-Reeves
PR-Polak-Ribiére

LM- Levenberg-Marquardt
OSS- One Step Secant

QN-Quasi Newton
DFP-Davidon-Fletcher-Powell

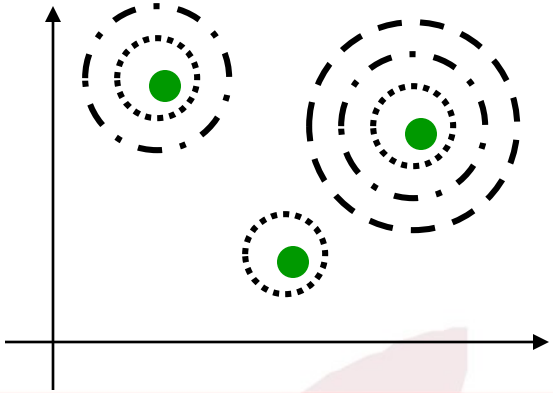
BFGS-Broyden-Fletcher-Goldfarb-Shanno

Radial-Basis Function Networks

- A função é aproximada como uma combinação linear de Funções de Base Radial *(Radial Basis Functions - **RBF**)*. As RBFs capturam o comportamento local das funções.

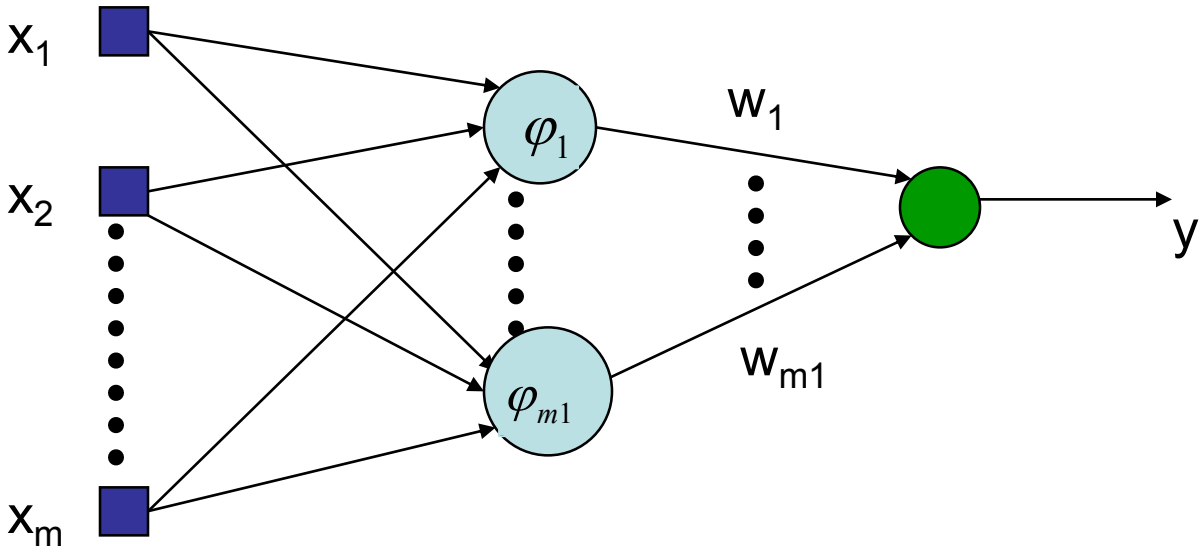
Motivação Biologica:

- RBF's representam receptores locais:



Arquitetura

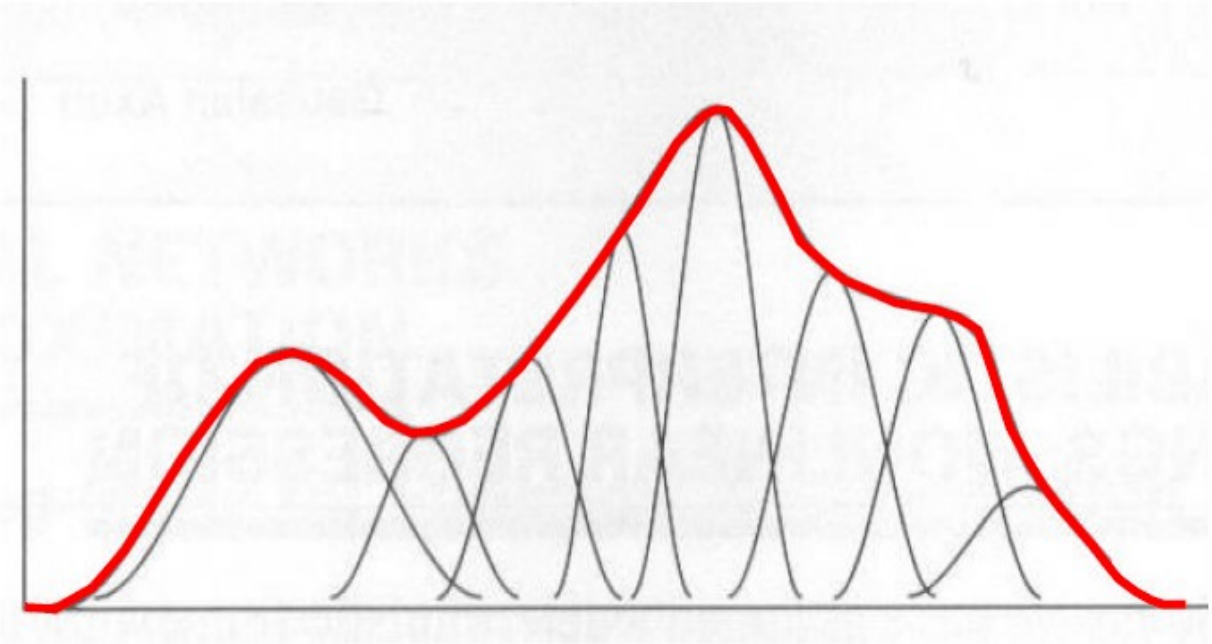
- Camanda de entrada: nodos que conectam a rede ao ambiente.



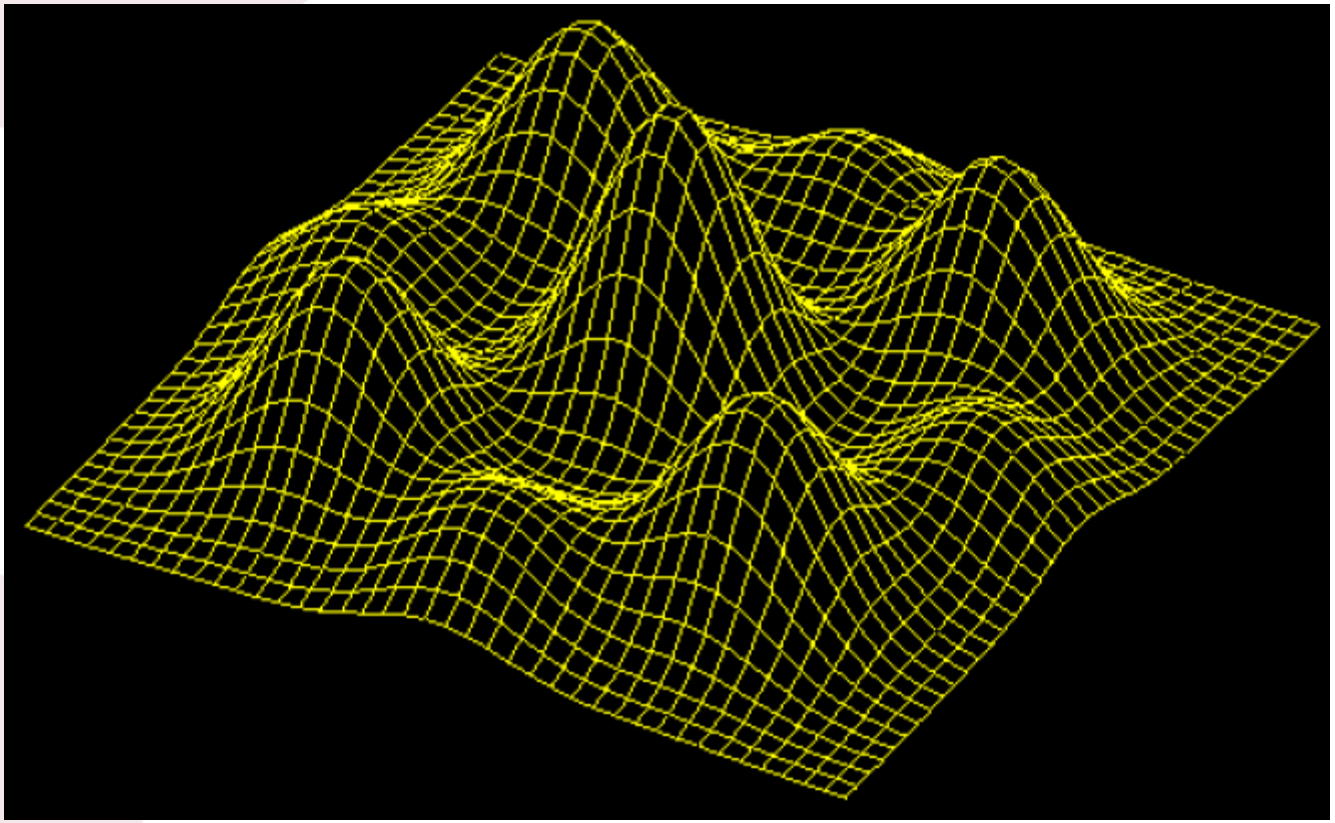
- **Camada Escondida: aplica uma transformação não-linear do espaço de entrada para uma representação no espaço gerado pelas ativações dos neurônios da camada escondida**
- **Camada de Saída: aplica uma transformação linear do espaço escondido para o espaço de saída**

Aproximação de funções 1-D

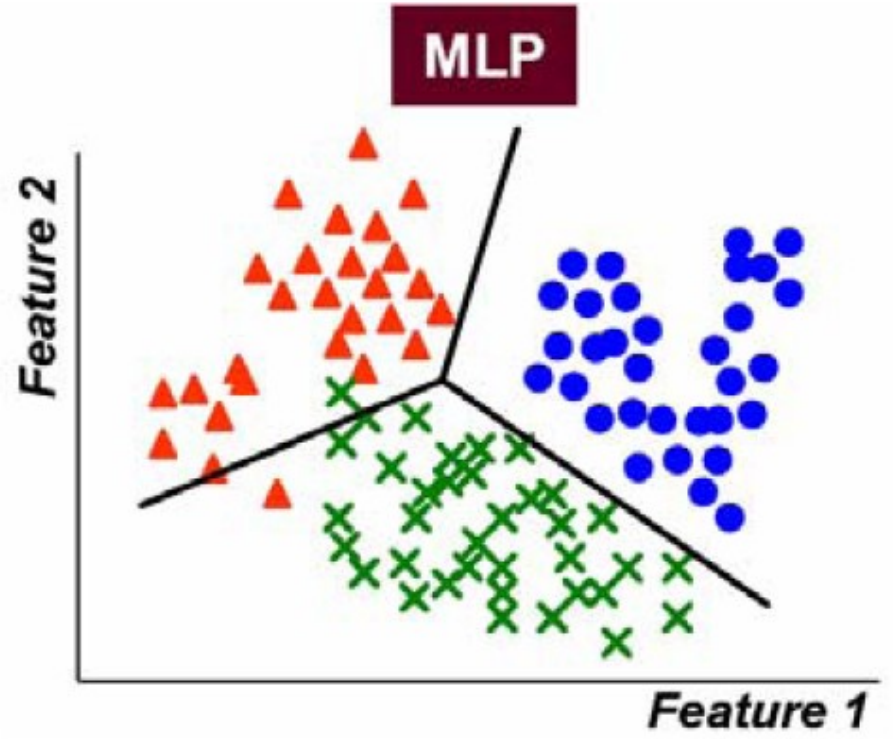
A rede RBF pode aproximar qualquer função contínua através da combinação linear de funções gaussianas com centros em diferentes posições do espaço de entrada.

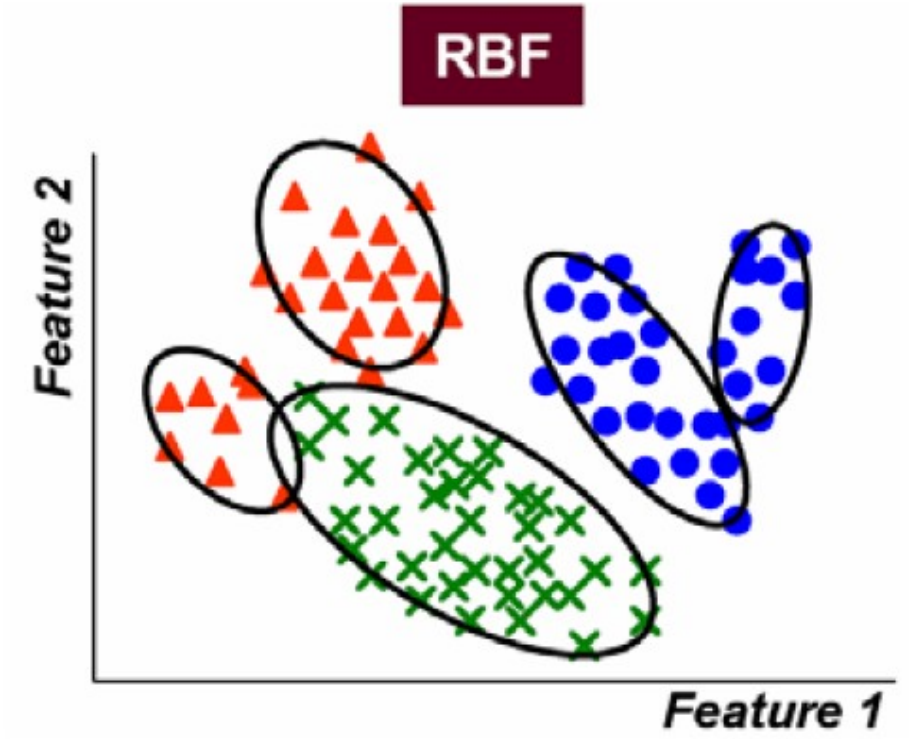


Aproximação de funções 2-D



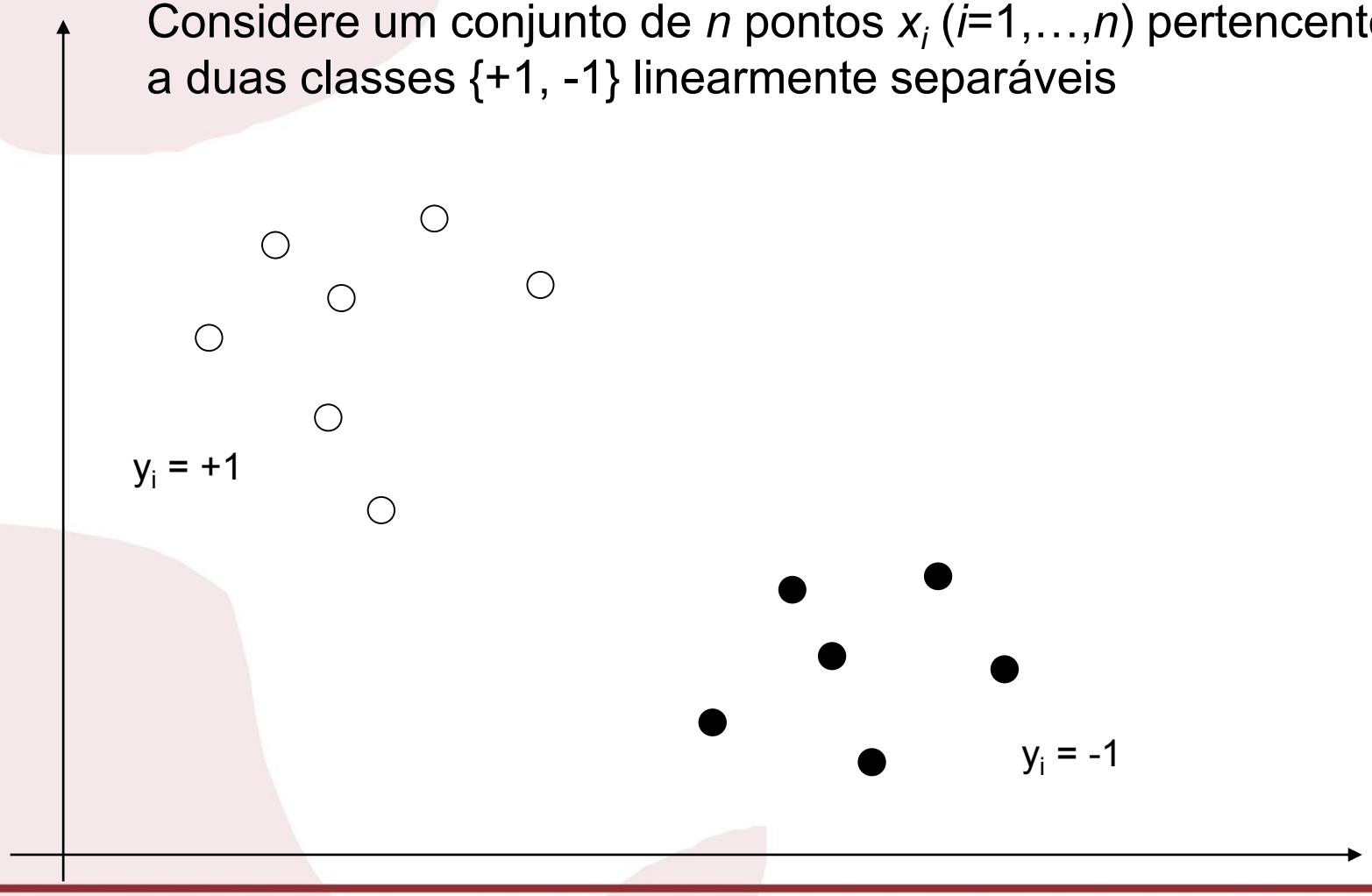
Comparação com as Redes MLP



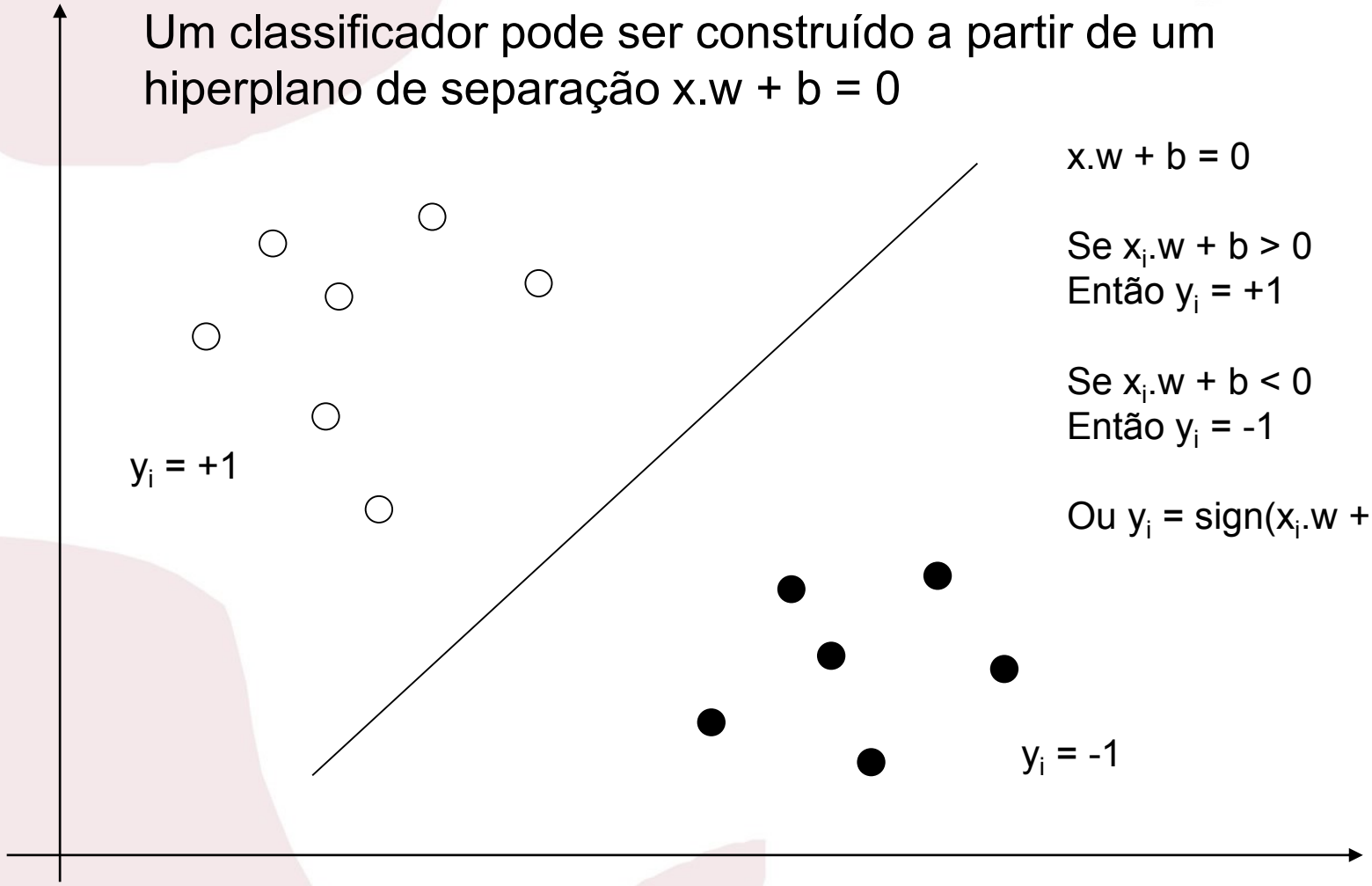


Support Vector Machines - SVM

Considere um conjunto de n pontos x_i (i=1,…,n) pertencentes a duas classes {+1, -1} linearmente separáveis



Um classificador pode ser construído a partir de um hiperplano de separação x.w + b = 0



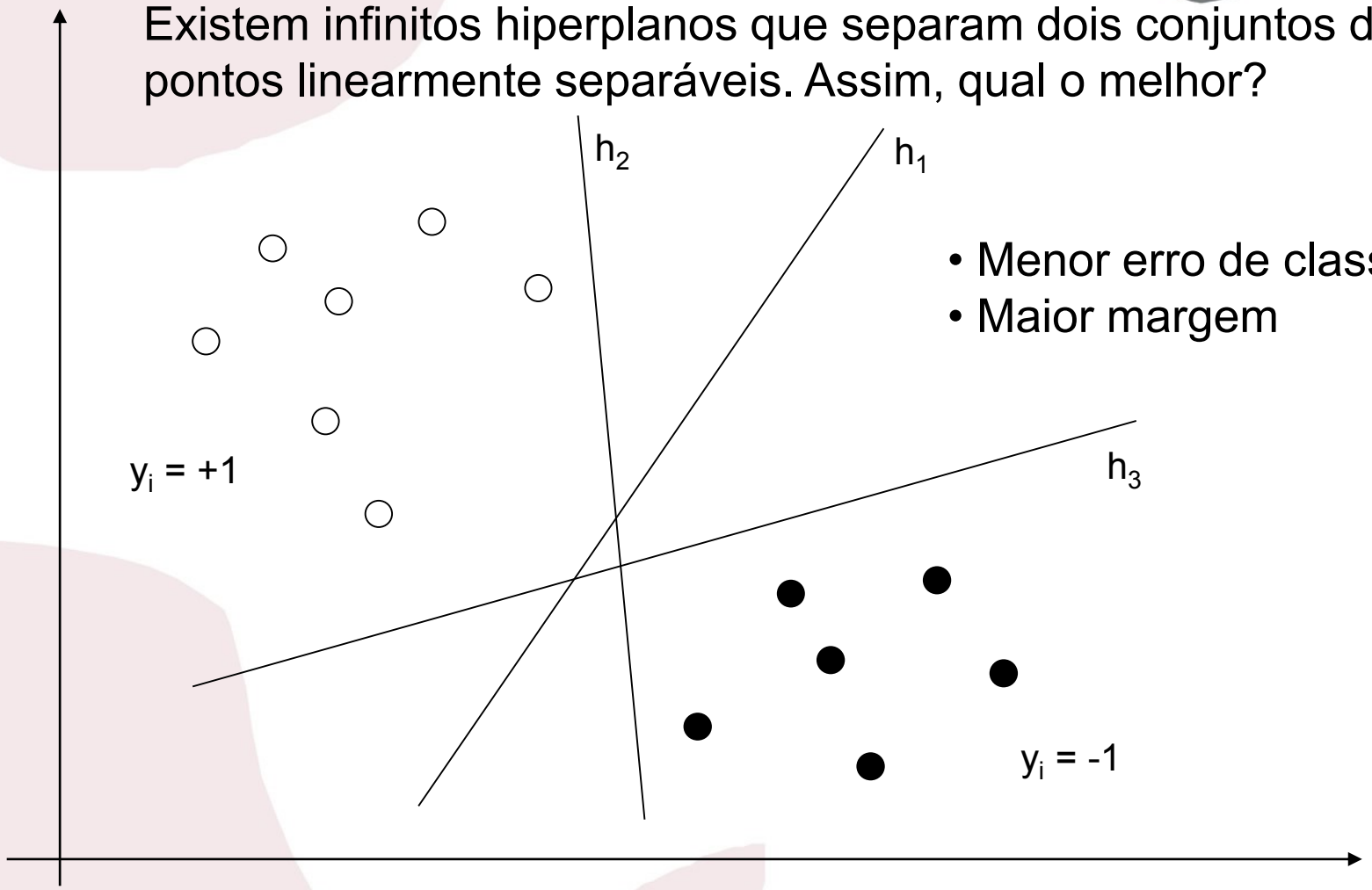
$x.w + b = 0$

Se $x_i.w + b > 0$
Então $y_i = +1$

Se $x_i.w + b < 0$
Então $y_i = -1$

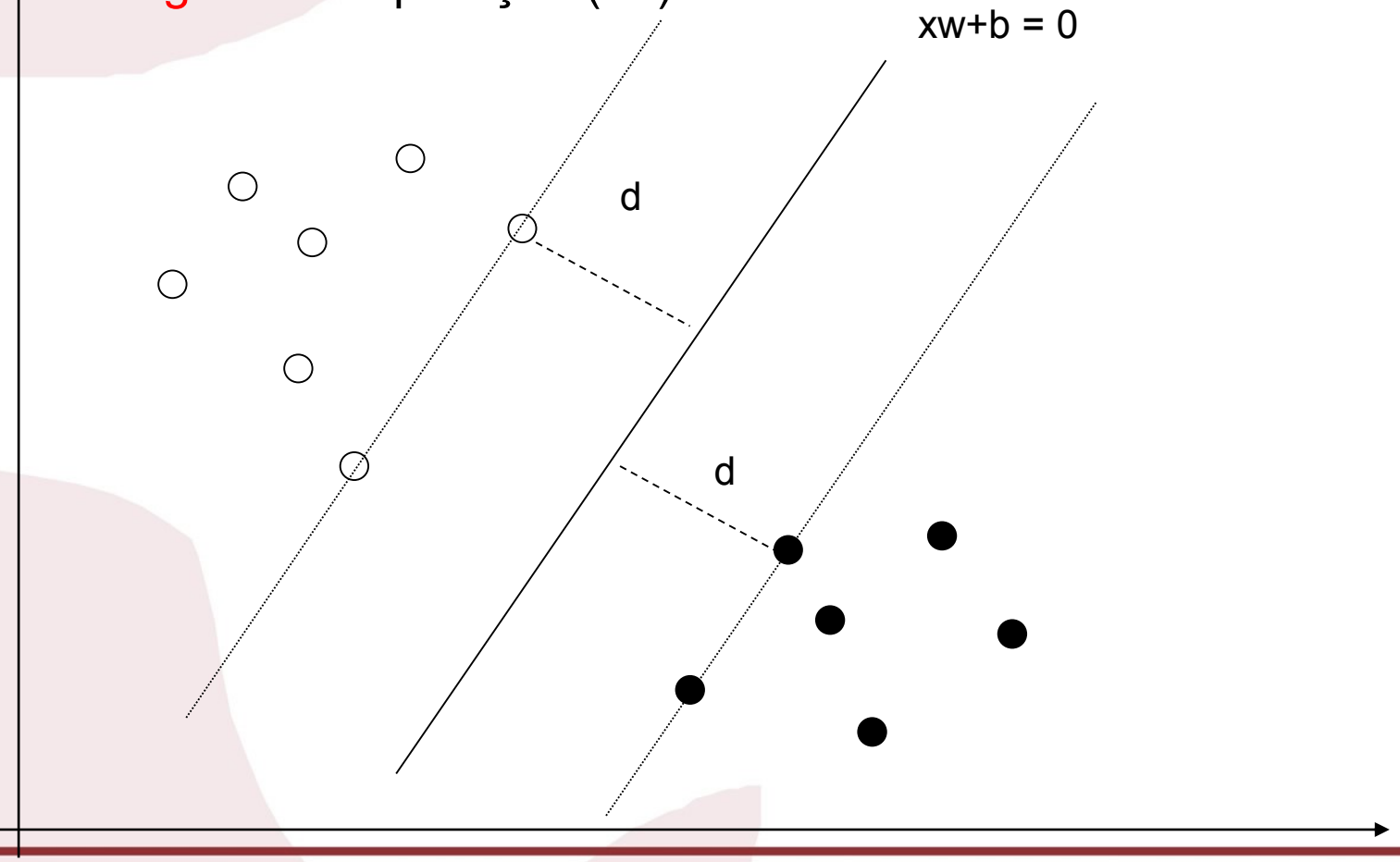
Ou $y_i = \text{sign}(x_i.w + b)$

Existem infinitos hiperplanos que separam dois conjuntos de pontos linearmente separáveis. Assim, qual o melhor?

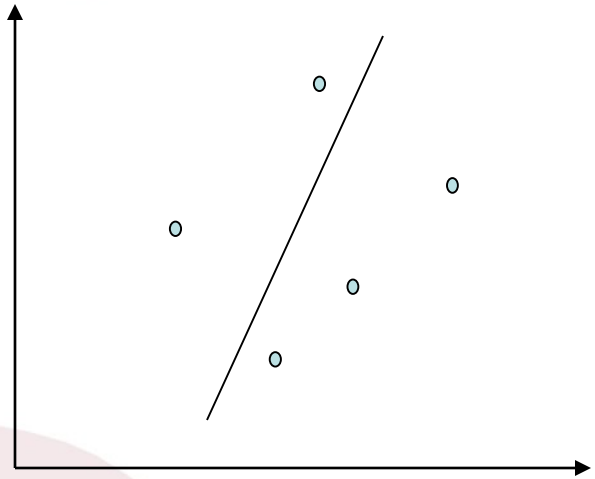


- Menor erro de classificação
- Maior margem

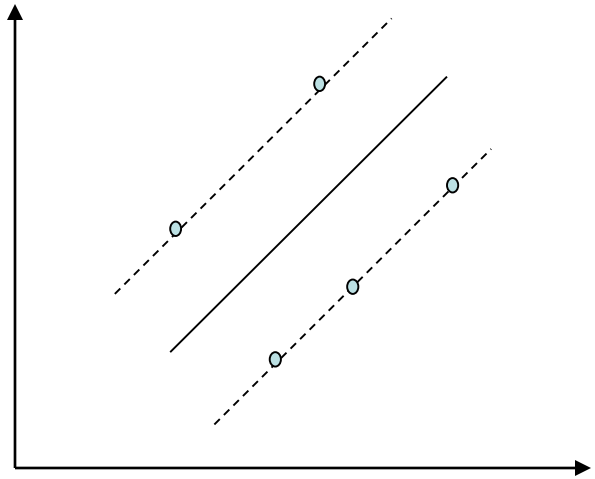
Hiperplano ótimo é equidistante às classes e maximiza a margem de separação (2d)



Comparação

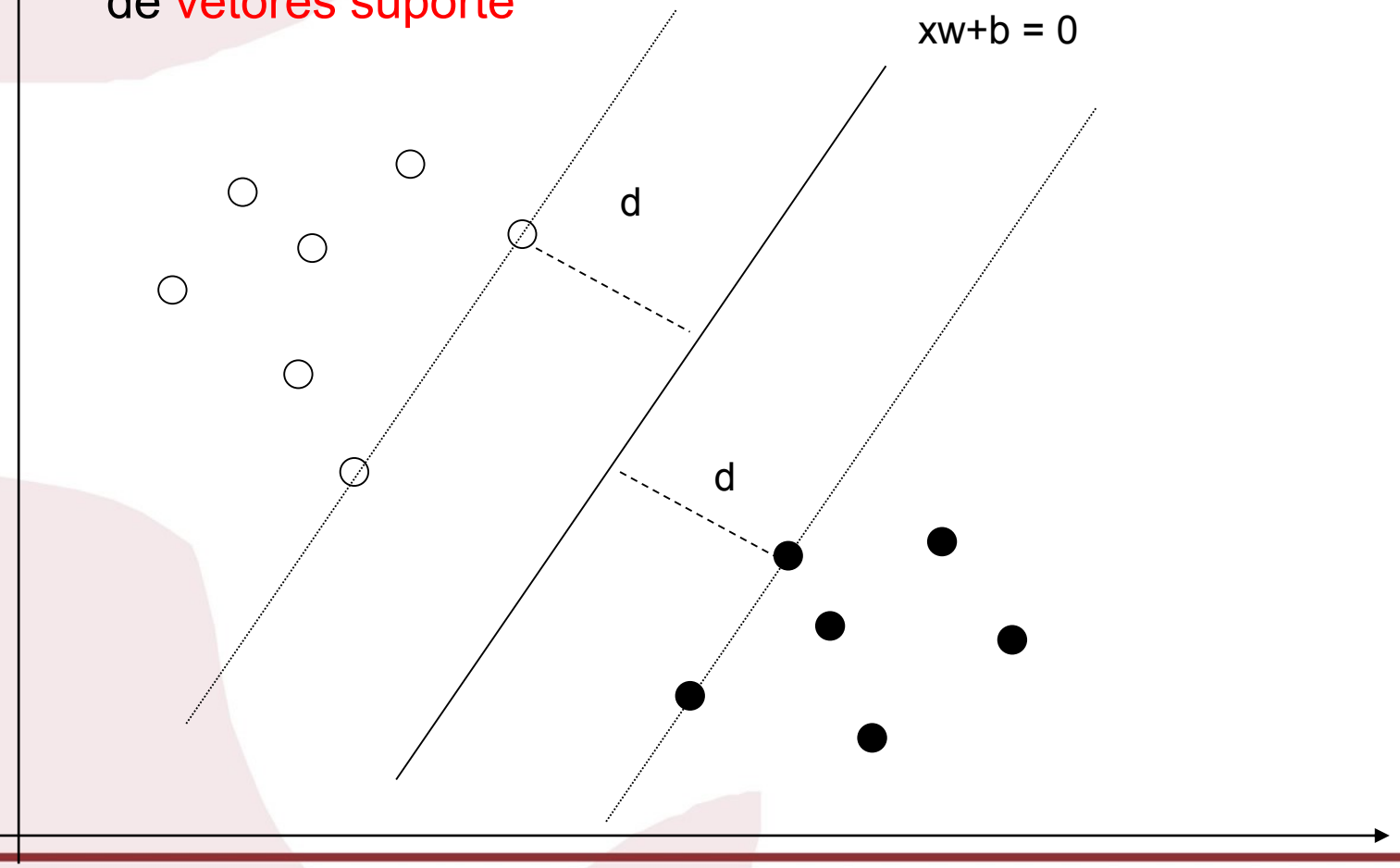


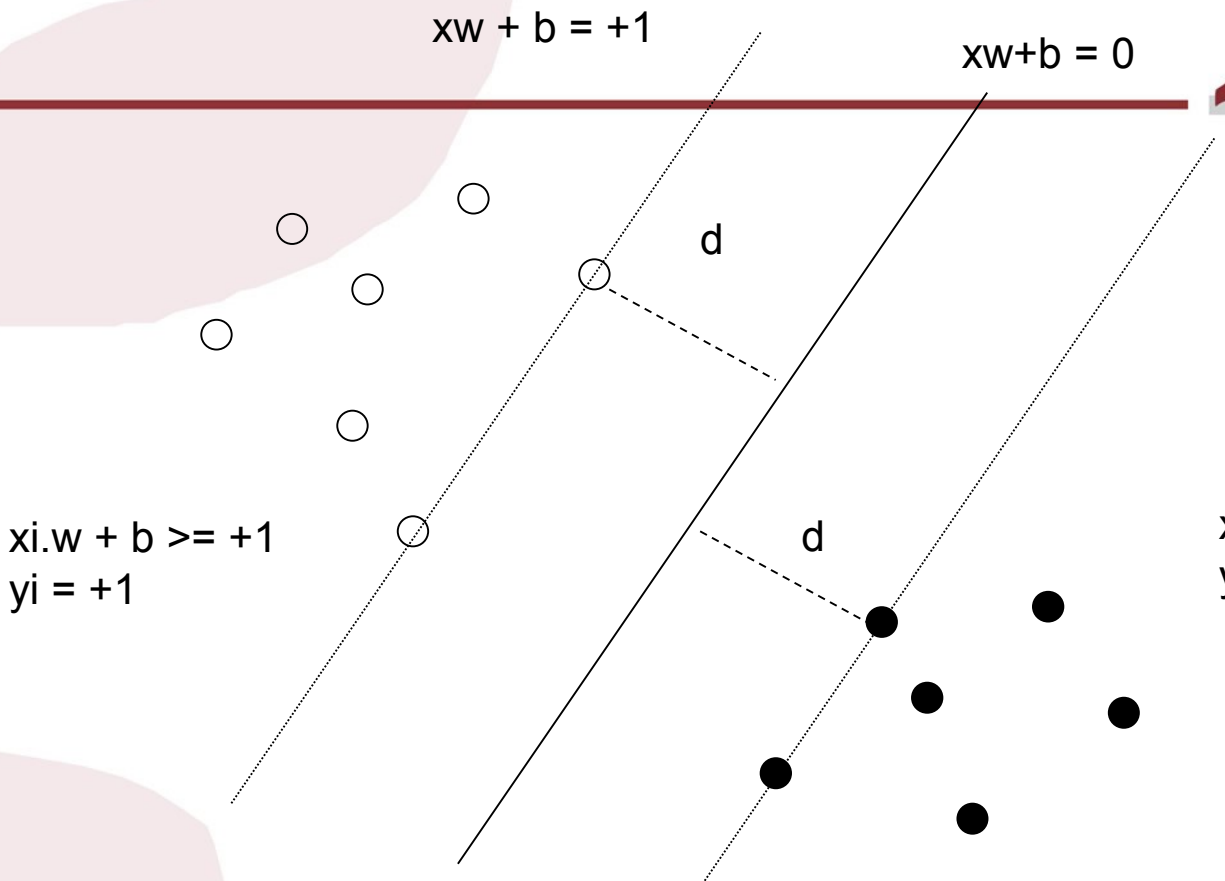
Rede Neural Linear



SVM Linear

Pontos mais próximos do hiperplano ótimo são chamados de vetores suporte

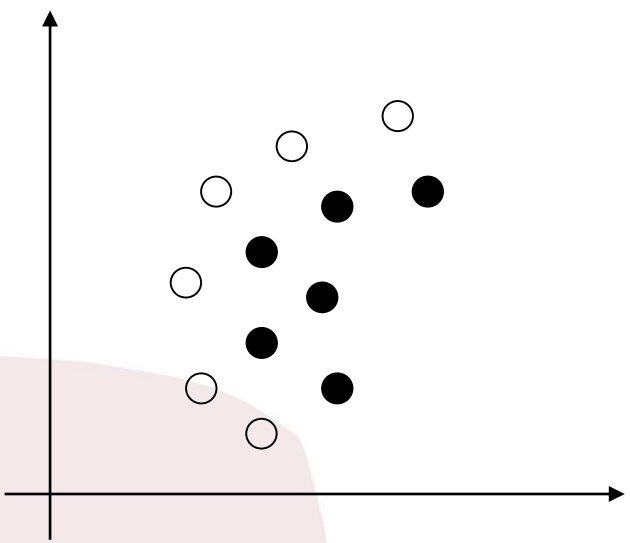


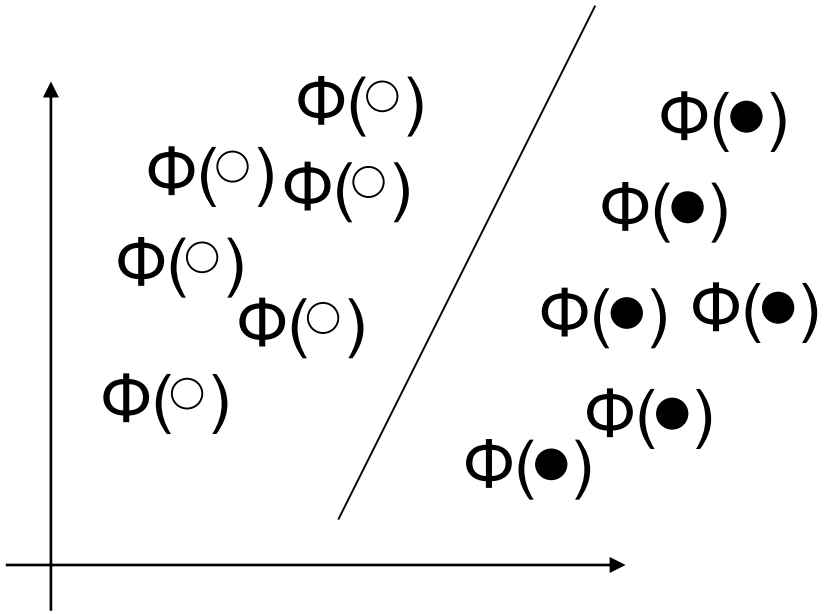


Hiperplanos superior e inferior podem ser reescalonados para:
xw + b = +1 e xw + b = -1

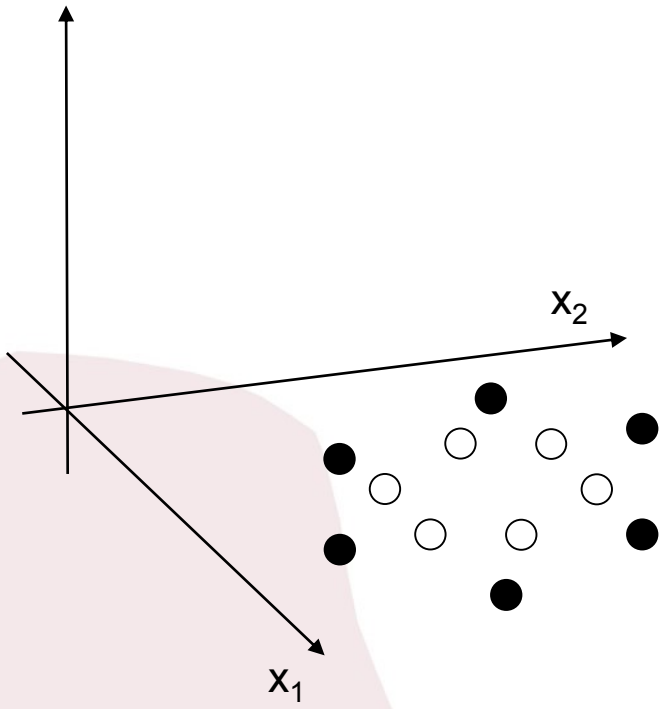
Margem 2d é calculada como: $\frac{2}{||w||}$

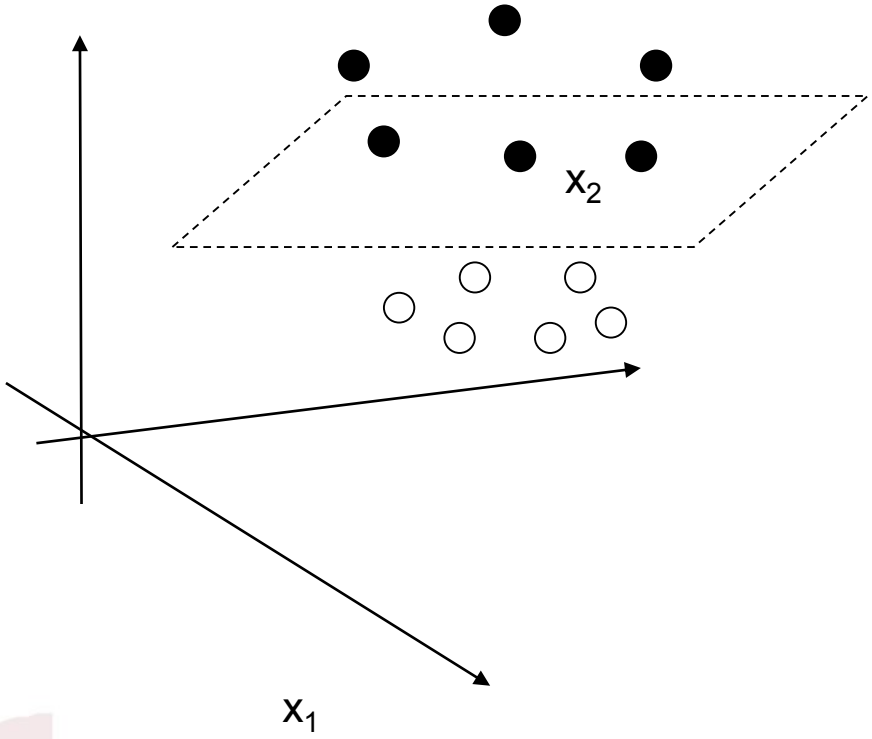
SVM Não-Linear





SVM Não-Linear

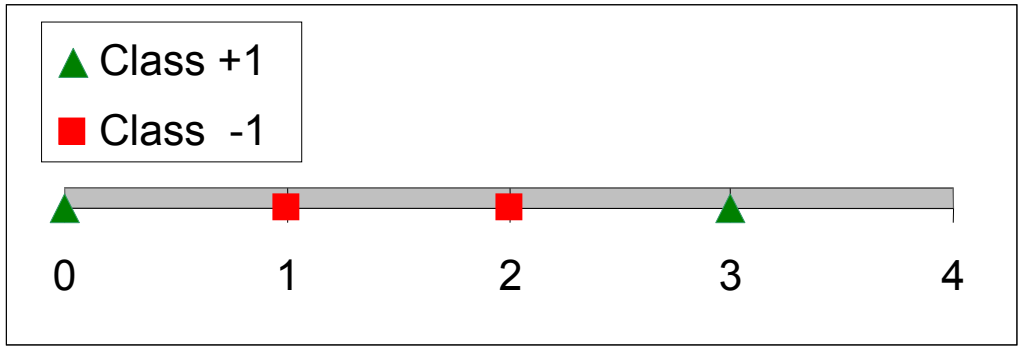




Exemplo

Como separar as duas classes com apenas um ponto?

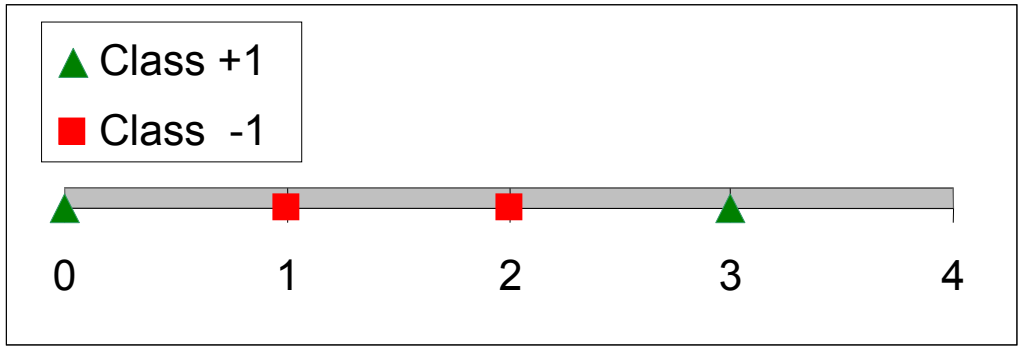
X_1	Class
0	+1
1	-1
2	-1
3	+1



Exemplo

SVM usa uma função não linear sobre os atributos do espaço de características inicial

X_1	Class
0	+1
1	-1
2	-1
3	+1



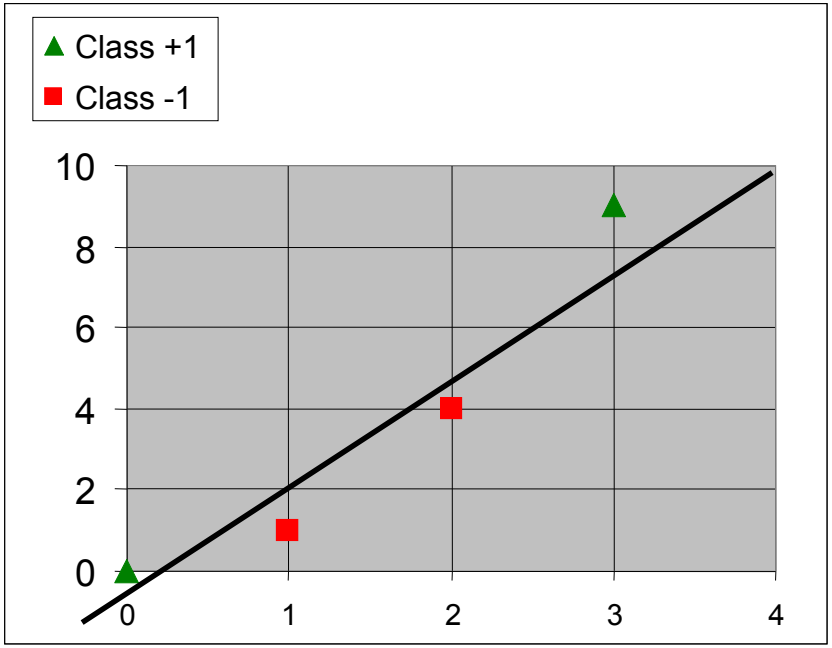
$\Phi(X_1) = (X_1, X_1^2)$

Esta função torna o problema bidimensional

Exemplo

SVM usa uma função não linear sobre os atributos do espaço de características inicial

X_1	X_1^2	Class
0	0	+1
1	1	-1
2	4	-1
3	9	+1



$\Phi(X_1) = (X_1, X_1^2)$

Esta função torna o problema bidimensional e os dados linearmente separáveis

Exemplo

X_1	X_1^2	Class
0	0	+1
1	1	-1
2	4	-1
3	9	+1

- $w \cdot x + b = +1$

  $w_1x_1 + w_2x_2 + b = +1$

  $0w_1 + 0w_2 + b = +1$ → $b = 1$

  $3w_1 + 9w_2 + b = +1$

- $w \cdot x + b = -1$

  $w_1x_1 + w_2x_2 + b = -1$ substituindo b e após w_1

  $1w_1 + 1w_2 + b = -1$ → $w_1 = -2 - w_2$

  $2w_1 + 4w_2 + b = -1$ → $-4 -2w_2 + 4w2 + 1 = -1$

- $w \cdot x + b = 0$ $w_2 = 1$ e $w_1 = -3$

  $w_1x_1 + w_2x_2 + b = 0$ → $\mathbf{-3x_1 + x_2 + 1 = 0}$

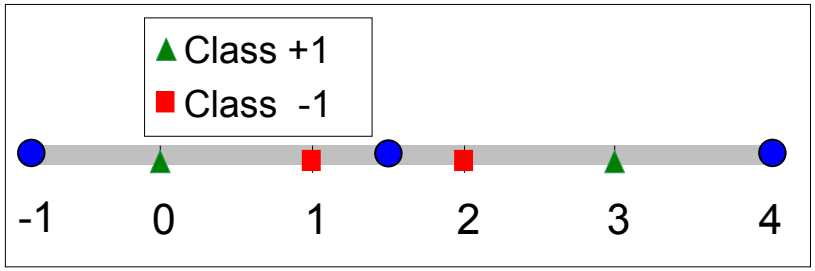
Exemplo

H: $-3x_1 + x_2 + 1 = 0$
Dados de Teste (1.5), (-1), (4)

(1.5) mapear para (1.5, 2.25)
-3 . 1.5 + 2.25 + 1 = - 1.15 [-1]

(-1) mapear para (-1,1)
-3 . -1 + 1 + 1 = 5 [+1]

(4) mapear para (4,16)
-3 . 4 + 16 + 1 = 5 [+1]



Problema

- Como escolher a função $\Phi(x)$ tal que o espaço de características transformado seja eficiente para classificação e não possua custo computacional alto demais?
  - Com uma função especial, chamada **função kernel** é possível calcular o produto escalar $\Phi(x_i).\ \Phi(x_j)$ sem mesmo conhecer o mapeamento Φ!

Demo

MLP

Applet: http://lcn.epfl.ch/tutorial/english/mlp/html/index.html
http://freeisms.com/MLPAppletItself.html

RBF

Applet: http://lcn.epfl.ch/tutorial/english/rbf/html/index.html
http://www.cvlibs.net/projects/gausspro.html

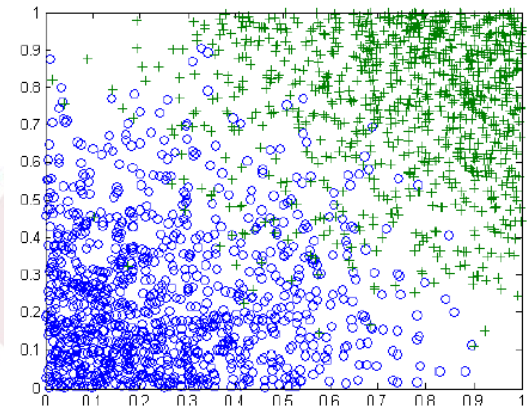
SVM

Applet: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
http://www.cs.jhu.edu/~jason/tutorials/SVMApplet/

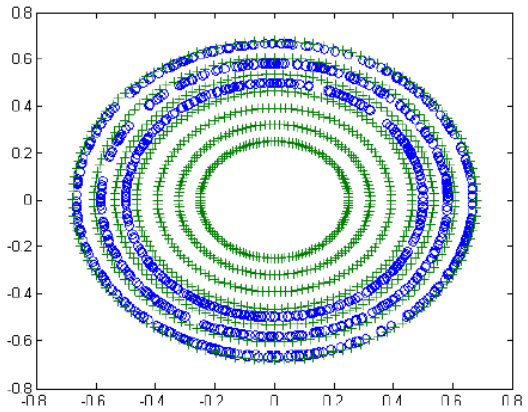
Vários classificadores

Applet: http://www.cs.technion.ac.il/~rani/LocBoost/

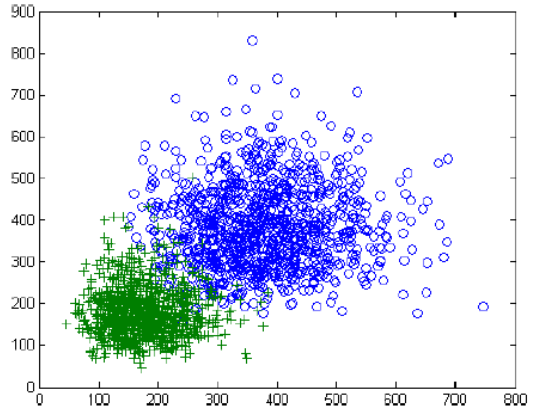
Toy problems



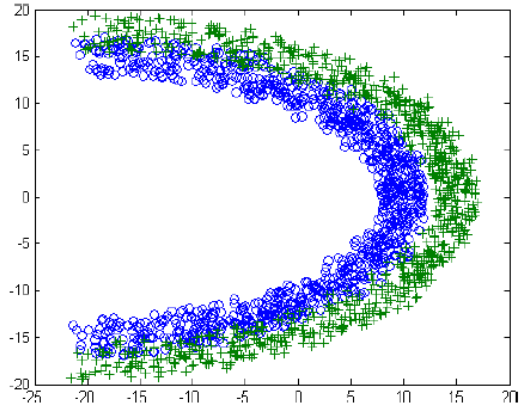
(a) Beta Distribution



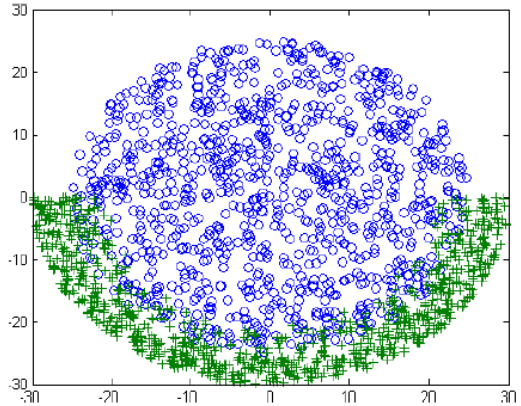
(b) Cluster in Cluster



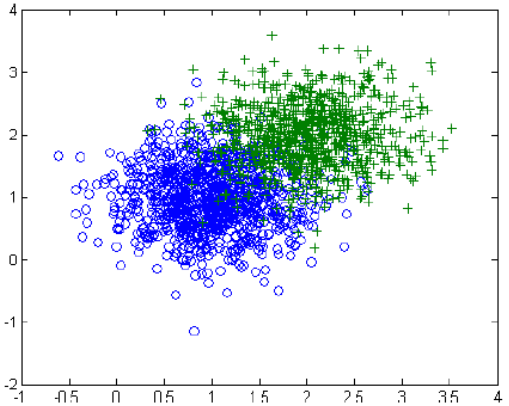
(c) Gamma Distribution



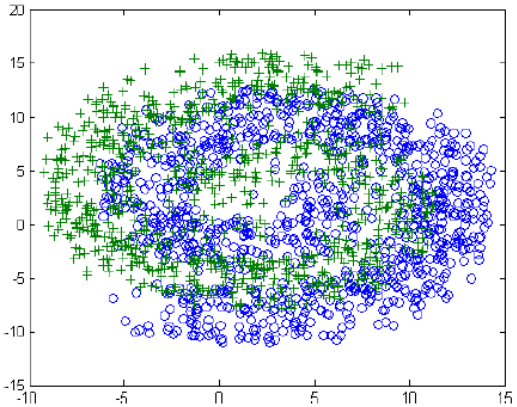
(d) Half Kernel



(e) Moon



(f) Normal Distribution



(g) Spirals

Aplicação prática

Problema Abordado

- Os problemas poderão ser de dois tipos:
  - **classificação** ou
  - **aproximação**.
- 1) Problemas de Classificação: Dado um padrão (exemplo), a rede deve dar como resposta a classe à qual ele pertence.

Idade	**Renda**	…	**Profissão**	**Classe**	
24	1070	…	Engenheiro	Bom pagador	← Padrão 1
…	…	…	…	…	…
41	4700	…	Professor	Mau pagador	← Padrão N

Atributos numéricos (ou quantitativos)

Atributo categórico (ou qualitativo)

Problema Abordado

– 2) Problemas de **Aproximação**: Dado um padrão, a rede deve **gerar saídas que se aproximem das saídas verdadeiras**.

Temperatura	Umidade	…	Dir. dos Ventos	Quant. chuva
27	0.28	…	Norte	0.12
…	…	…	…	…
21	0.67	…	Sudeste	1.32

– Em ambos os casos, deseja-se **generalização**, ou seja, que a rede seja capaz de gerar as saídas mais corretas possíveis não apenas para os **padrões apresentados no treinamento**, mas também para **padrões novos**.

Pré-processamento

- 1) Para os atributos **numéricos**: Fazer **normalização** (escalonamento):
  - Valores devem ser normalizados para o **intervalo [0,1] (depende da função de ativação a ser utilizada – sigmóide logística**).
  - Expressão de normalização:

$$x_{norm} = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})}$$

  - onde x_{norm} é o **valor normalizado** correspondente ao valor **original** x, e x_{min} e x_{max} são os valores **mínimo** e **máximo** entre todos os valores da base de dados.
  - Pode ser feito separadamente por atributo.
  - Este processo não é necessariamente o melhor **(o objetivo é apenas didático).**

Pré-processamento

- 2) Para os atributos **categóricos** e saídas: Fazer **codificação ortogonal**.
  - Atribui-se uma **seqüência de bits** ao atributo, sendo que apenas um dos bits vale 1, indicando a categoria.
  - Ex.: Se um atributo tiver como valores possíveis **A**, **B** e **C**, então:
    - a categoria **A** fica codificada como **1 0 0**,
    - a categoria **B** fica codificada como **0 1 0**,
    - a categoria **C** fica codificada como **0 0 1**.

Pré-processamento

- 3) No caso de **missing data** (valores faltando):
  - a) **Eliminação de padrões**
    - Quando há padrões suficientes, eliminamos aqueles com valores faltando.
    - Obs.: É difícil definir quantos padrões são “suficientes” (neste projeto, isto não é cobrado rigorosamente).
  - b) **Substituição de valores**
    - i) Atributos **numéricos**: o valor faltando pode ser **substituído pela média** dos valores nos demais padrões.
    - ii) Atributos **categóricos**:
      - » pode ser criada uma **nova categoria** (“ausente”),
      - » ou o valor faltando pode ser **substituído por alguma categoria existente** (observando a semântica).

Particionamento dos Dados

– O particionamento adotado é o sugerido pelo conhecido relatório *Proben1*:

  - **50%** dos padrões de cada classe escolhidos aleatoriamente para **treinamento**,
  - **25%** para **validação**,
  - **25%** para **teste**.

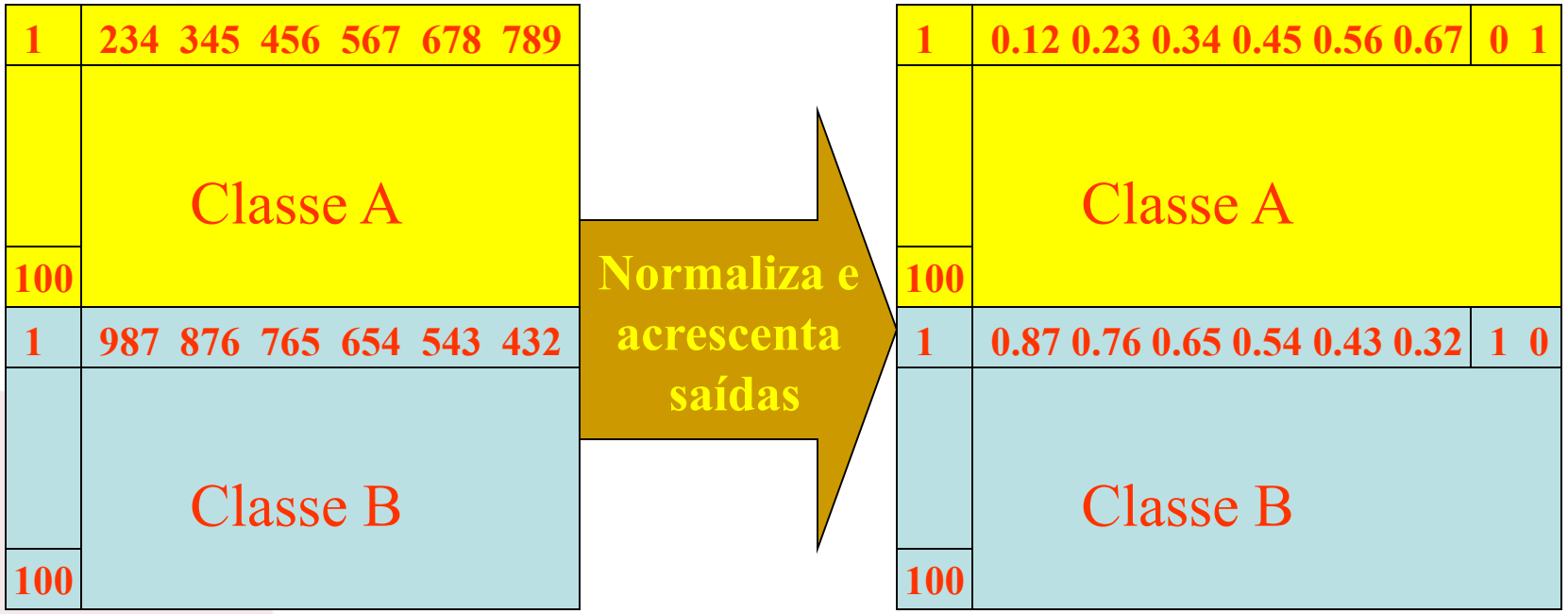
– É importante que as **proporções entre as classes** no conjunto completo sejam mantidas nos conjuntos de treinamento, validação e teste.

Particionamento dos Dados

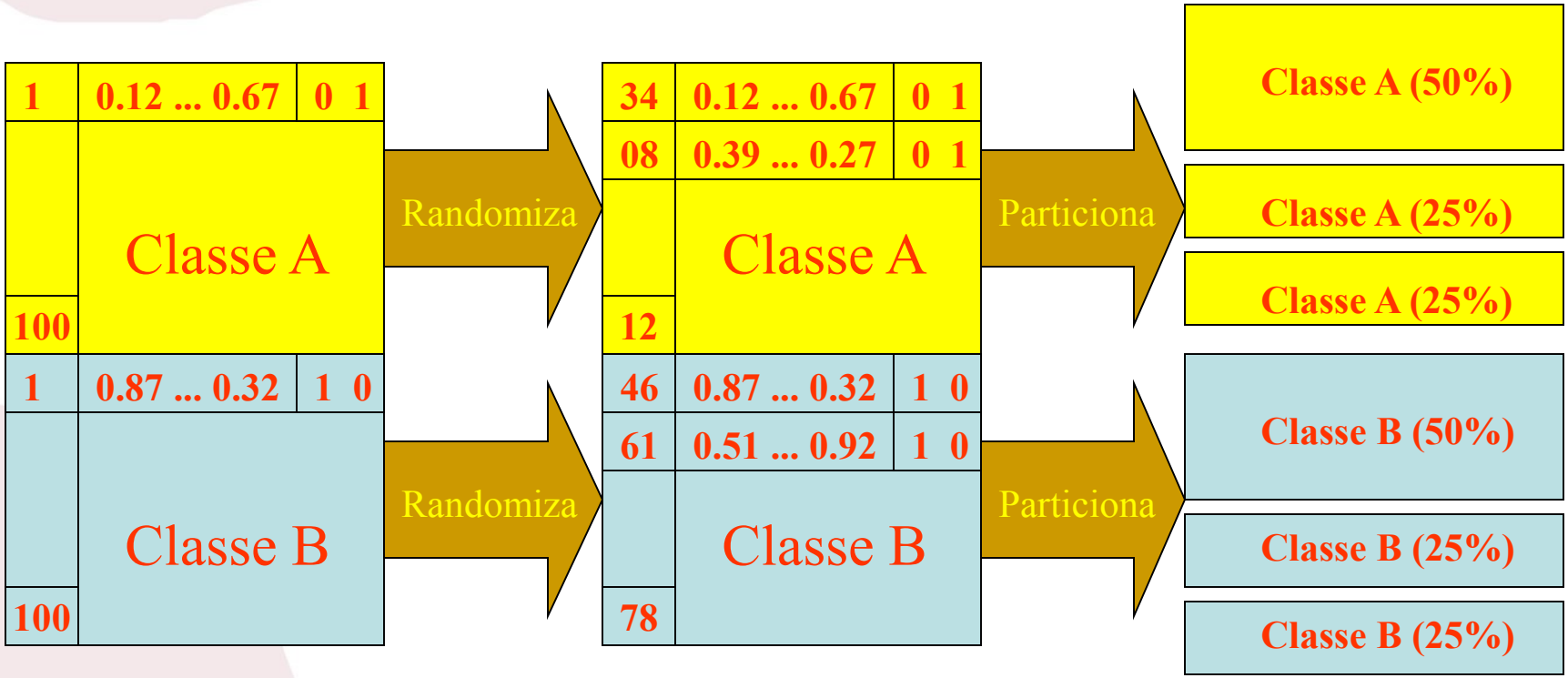
- Se houver **quantidades diferentes de padrões** nas classes, a equipe pode escolher entre:
  - a) **Usar a mesma quantidade de padrões para todas as classes.**
    - Ex.: Classe A com 200 padrões e classe B com 100 padrões.
      - » Usa todos da classe B e escolhe aleatoriamente 100 padrões da classe A.
    - Obs.: Pode ser que a base fique com poucos padrões (é difícil definir a quantidade certa; não será cobrado com rigor).
  - b) **Usar todos os dados, mantendo a proporção existente.**
    - Pode ser que as classes com mais padrões sejam “muito mais aprendidas” do que as outras.

Particionamento dos Dados

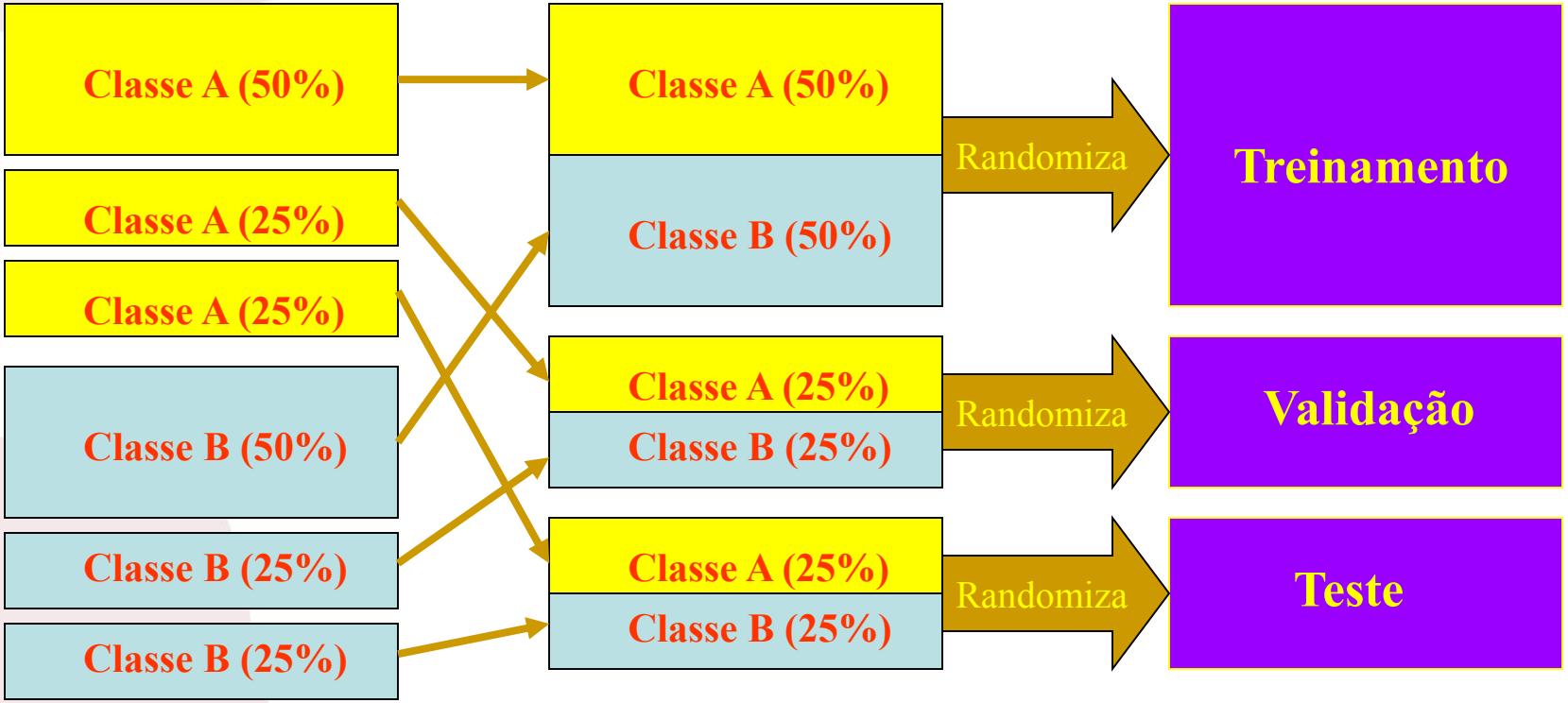
Exemplo:



Particionamento dos Dados



Particionamento dos Dados



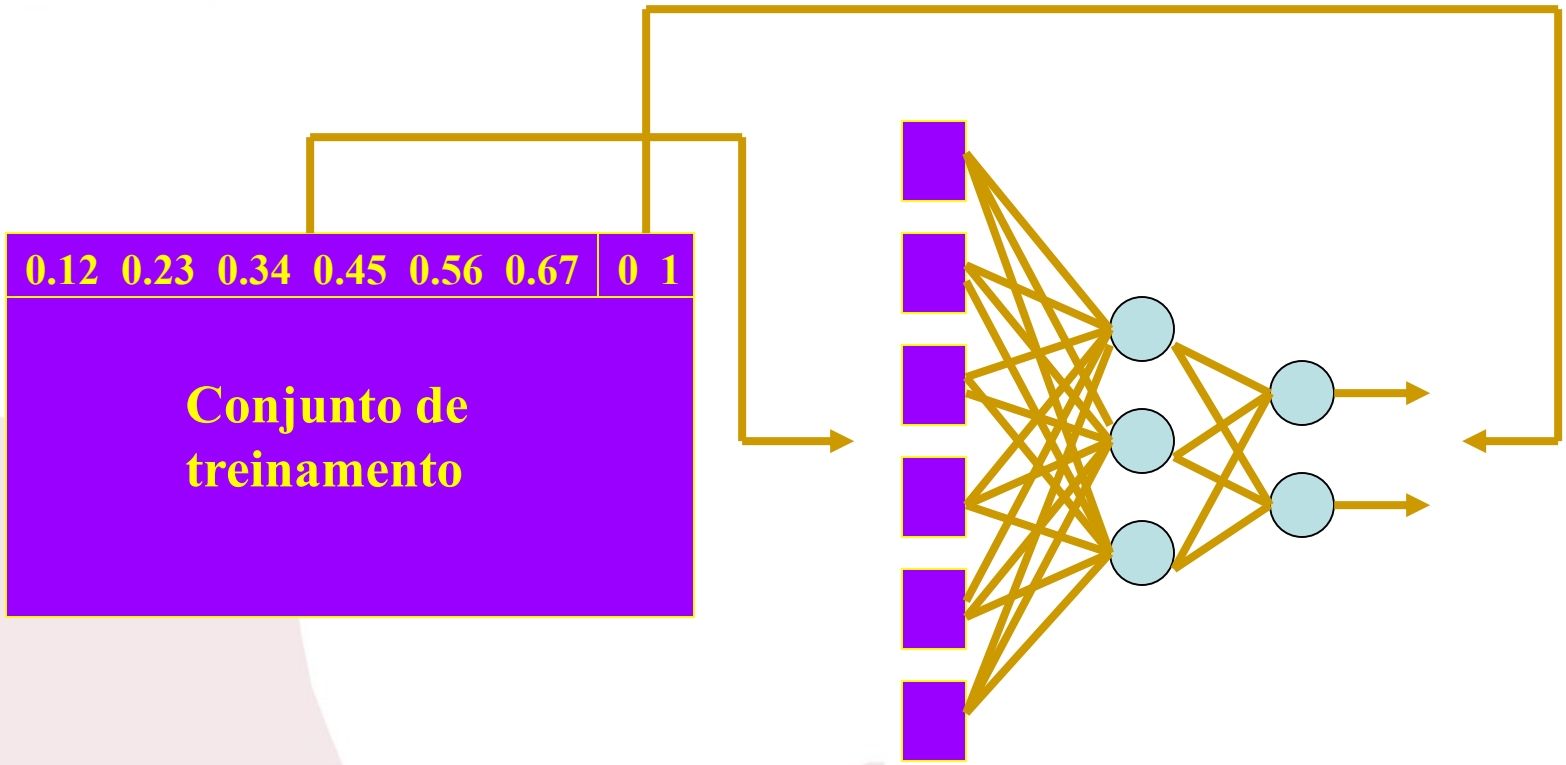
Definição da Topologia MLP

– Aspectos de treinamento:

- Nº de nodos de **entrada**: Quantidade de atributos de entrada.
- Nº de nodos de **saída**:
  - Em problemas de **classificação**, é a quantidade de classes.
    - » Regra de classificação ***winner-takes-all***: o nodo de saída que gerar a maior saída define a classe do padrão.
  - Em problemas de **aproximação**, é a quantidade de variáveis de saída.
- **Uma única camada escondida**.
- **Função de ativação** dos neurônios: sigmóide logística.
- Todas as possíveis conexões entre camadas adjacentes, sem conexões entre camadas não-adjacentes.

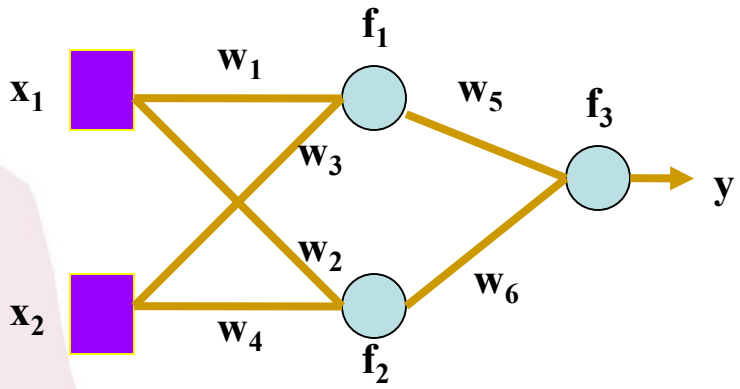
Definição da Topologia MLP

Exemplo: 6 entradas e 2 saídas.



Definição da Topologia MLP

- Aspecto que será **variável** neste projeto:
  - **Nº de neurônios escondidos** (projetista).
- Variando o nº de neurônios escondidos, estamos variando a quantidade de pesos da rede.
- Explicação:Uma rede neural **implementa uma função**.



As funções f_i são do tipo sigmóide logística.

$$y = f_3(w_5 f_1 (w_1 x_1 + w_3 x_2) + w_6 f_2 (w_2 x_1 + w_4 x_2)).$$

Definição da Topologia MLP

- Os pesos da rede são os **parâmetros da função**.
- Dessa forma, aumentar a quantidade de pesos da rede significa **aumentar a complexidade da função** implementada.
  - Se a quantidade de pesos for **pequena demais**, pode haver **underfitting**.
    - A função implementada **não tem complexidade suficiente** para resolver o problema abordado.
  - Se a quantidade de pesos for **grande demais**, pode haver **overfitting**.
    - A função implementada tem **complexidade demais para o problema**, sendo capaz de **modelar detalhes** demais dos dados de treinamento.
    - Dessa forma, a rede **não generaliza** bem.

Medidas de Erro

– Para ambos os tipos de problema, será usado o **erro SSE** (*sum squared error* - soma dos erros quadráticos).

– Ex.:

	Saídas da rede			Saídas desejadas		
Padrão	1	...	N	1	...	N
Nodo 1	0.98	...	0.12	1.00	...	0.00
Nodo 2	0.02	...	0.96	0.00	...	1.00

– Soma dos erros quadráticos (SSE):

SSE = $(0.98 - 1.00)^2 + ... + (0.12 - 0.00)^2 +$
$(0.02 - 0.00)^2 + ... + (0.96 - 1.00)^2$.

$$SSE = \sum_{i=1}^{p} \left\| t^{(i)} - y^{(i)} \right\|^2$$

Medidas de Erro

- Para problemas de **classificação**, também será calculado o **erro de classificação** (neste projeto, só para o conjunto de teste).
- Regra de classificação ***winner-takes-all***:
  - O neurônio de saída que apresentar o maior valor de saída determina a classe do padrão.
  - Ex.:

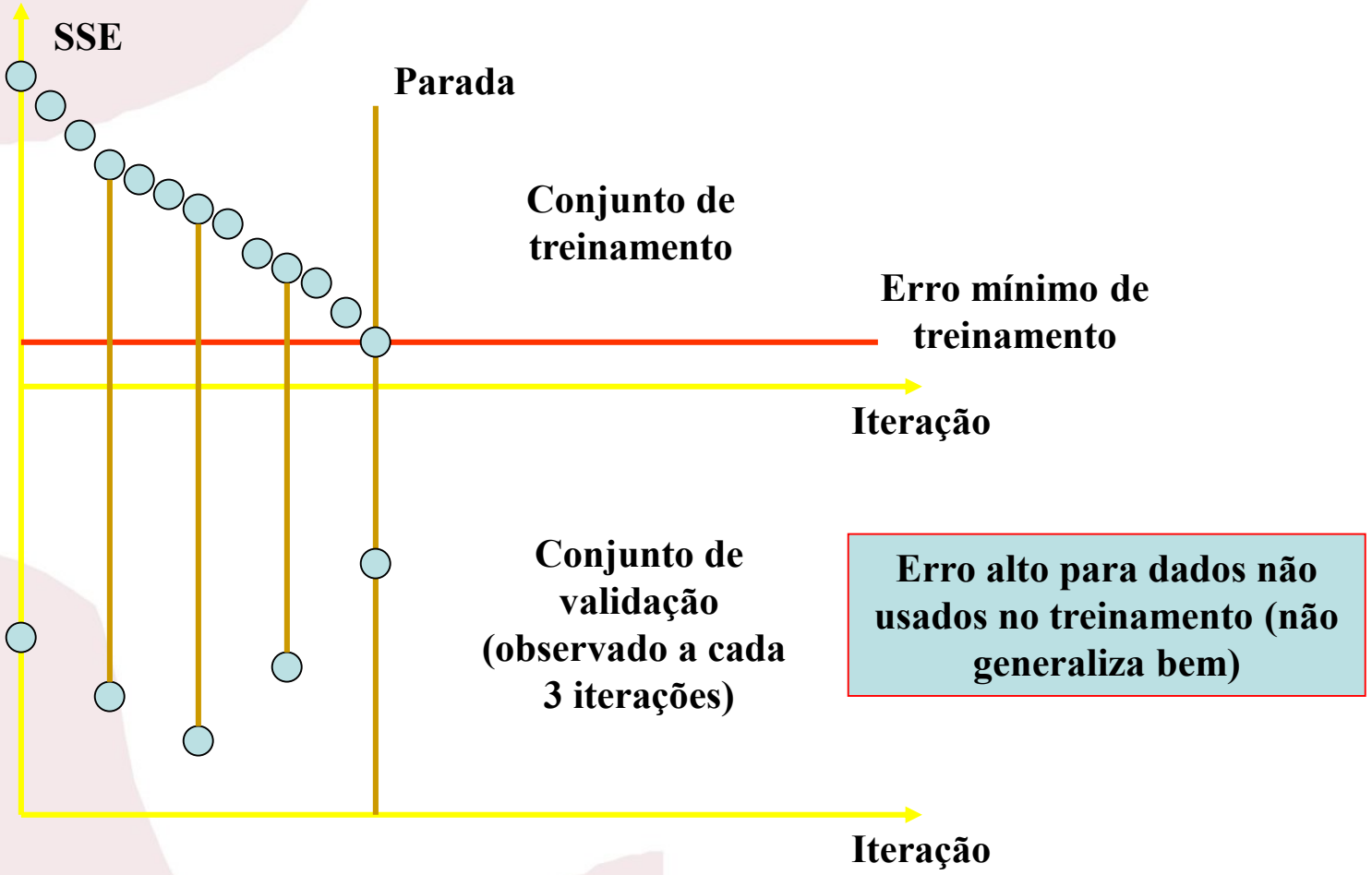
	Saídas da rede			Saídas desejadas		
Padrão	1	...	N	1	...	N
Nodo 1	0.98	...	0.12	1.00	...	0.00
Nodo 2	0.02	...	0.96	0.00	...	1.00
Classe	1	...	2	1	...	2

Erro Classif. = 100 x $\frac{\text{Quant. de padrões classificados erradamente}}{\text{Quant. total de padrões}}$

Treinamento com Backpropagation

- Será usado o algoritmo **backpropagation** padrão.
  - É um algoritmo de **gradiente descendente**, ou seja, utiliza informações de **derivada**.
  - Por isso, as funções de ativação devem ser **contínuas e diferenciáveis** (é o caso da sigmóide logística).
- **Objetivo**: Fazer "ajuste de pesos", ou seja, **escolher os pesos que geram as saídas mais corretas possíveis** (menor erro) de forma iterativa.
- **Idéia geral**: A cada iteração, obter um **erro cada vez menor** para os dados de treinamento.
- **Cuidado**: Não permitir que a rede aprenda **detalhes demais** do conjunto de treinamento (**overfitting**).

Parada por Erro Mínimo de Treinamento



Parada por Erro Mínimo de Validação

– É recomendável que o treinamento seja interrompido quando o **erro no conjunto de validação atingir um mínimo**.

  • A partir deste ponto, supõe-se que a rede só aprenderia **detalhes irrelevantes** do conjunto de treinamento.

  • O erro para dados de treinamento seria cada vez menor, mas o **erro para dados novos** (validação) seria cada vez mais alto.

– Neste projeto, será usado o seguinte **critério de parada**:

  • Interromper o treinamento quando o **erro de validação subir por 5 iterações consecutivas**.

  • É o critério implementado no Matlab (parâmetro “max_fail = 5”).

Parâmetros de Treinamento

- Os parâmetros **variáveis** do treinamento serão:
  - **Taxa de aprendizado,**
  - **Máximo de iterações permitidas** (é outro critério de parada).
- A equipe deve escolher **3 valores** para cada um deles.
- Usando **taxa de aprendizado muito baixa**, cada iteração faz um **ajuste muito pequeno** nos pesos (passo muito pequeno).
  - Pode precisar de **muitas iterações** para convergir para o ponto de mínimo desejado na superfície de busca.
- Usando **taxa de aprendizado muito alta**, cada iteração faz um **ajuste muito grande** nos pesos (passo muito grande).
  - Pode causar **oscilações** em torno de um ponto de mínimo.

Análise de Resultados

- Serão usadas:
  - 3 quantidades de neurônios escondidos,
  - 3 taxas de aprendizado,
  - 3 quantidades máximas de iterações permitidas.
- Temos um total de a serem testadas.
- Para cada configuração, será realizado um treinamento.
- A melhor configuração a ser escolhida é a de .

Config.	SSE de Treinamento	SSE de Validação
1	2.13	3.45
2	1.44	0.71
...	...	...
27	4.43	5.18

Melhor configuração

Análise de Resultados

- Para a **melhor configuração** escolhida, devem ser feitos **30 treinamentos com diferentes inicializações de pesos**.
- O objetivo é verificar como a melhor rede se comporta quando variamos os pesos iniciais.

Inicialização	SSE de Treinamento	SSE de Validação	SSE de Teste	E.Class. de Teste
1	1.12	0.66	0.79	12.08
2	1.44	0.71	0.88	13.32
...	...	...	...	...
30	1.23	0.66	0.90	09.87
Média	1.15	0.70	0.85	11.24
Desv-pad	0.07	0.11	0.10	02.35

Análise de Resultados

- Deve ser escrito um **relatório** com a análise dos resultados.
- Os **gráficos** a serem incluídos ficam a critério da equipe.
- **Dicas**:
  - Matriz de confusão

	Classe verdadeira		
Classe prevista	**1**	**2**	**3**
1	40	5	0
2	0	45	0
3	10	1	39

  - Matriz de custo

	Classe verdadeira		
Classe prevista	**1**	**2**	**3**
1	0	2	5
2	1	0	2
3	3	1	0

  - Gráficos de desempenho

Aspectos do treinamento de redes MLP

O aprendizado é resultado de **apresentação repetitiva** de todas as amostras do **conjunto de treinamento**.

Cada apresentação de todo o conjunto de treinamento é denominada **época ou iteração**.

O processo de aprendizagem é **repetido** época após época, até que um **critério de parada** seja satisfeito.

É recomendável que a **ordem de apresentação** das amostras seja **aleatória** de uma época para outra. Isso tende a fazer com que o **ajuste de pesos** tenha um caráter **estocástico** ao longo do treinamento.

Atualização local ou por lote.

Atualização pode ser de duas maneiras básicas: **local e por lote**.

Local: a atualização é feita imediatamente após a apresentação de cada amostra de treinamento.

- é também chamado de método de atualização **on-line** ou padrão a padrão.
- requer um menor armazenamento para cada conexão, e apresenta **menos possibilidade de convergência para um mínimo local**.

Lote: a atualização dos pesos só é feita após a apresentação de todas as amostras de treinamento que constituem uma época.

- é também conhecido como método de atualização **off-line** ou **batch**.
- o **ajuste** relativo a cada apresentação de uma amostra é **acumulado**.
- fornece uma melhor estimativa do vetor gradiente.

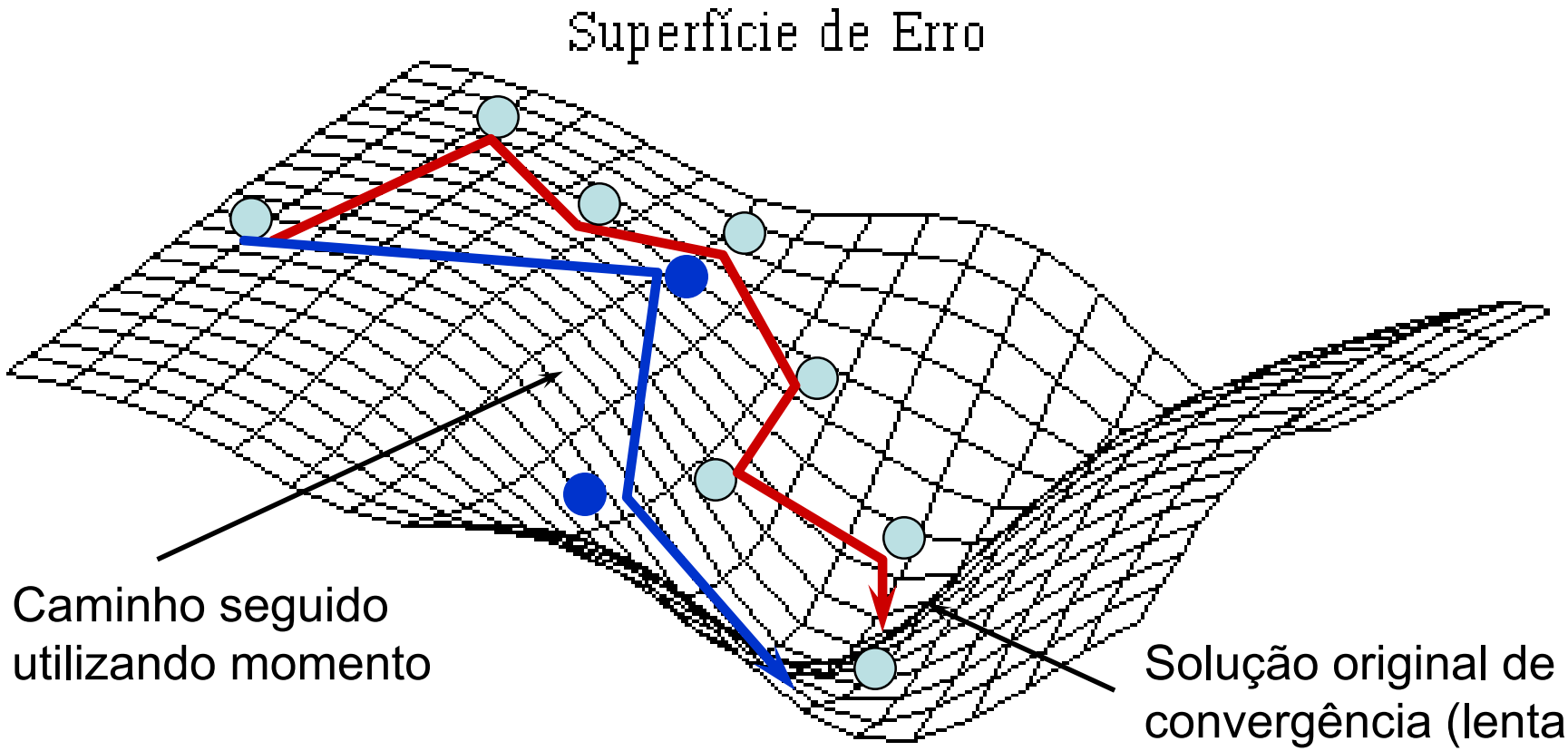
Atualização dos pesos

- 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
  - Normalmente, α é ajustada entre 0,5 e 0,9

Momento


Superfície de Erro
Caminho seguido utilizando momento
Solução original de convergência (lenta)

Critérios de parada

O processo de minimização do MSE (função custo) não apresenta convergência garantida e não possui um critério de parada bem definido.

Um critério de parada não muito recomendável, que não leva em conta o estado do processo iterativo é o da **pré-definição do número total de iterações**.

Apresenta-se a seguir um critério de parada que leva em conta o processo iterativo.

Critérios de parada (cont.)

Consideremos um critério que leva em conta informações a respeito do estado iterativo. Considera-se nesse caso a **possibilidade de existência de mínimos locais.**
Seja θ^* o **vetor de pesos** que denota um ponto **mínimo**, local ou global.

Uma **condição** para que θ^* seja um **mínimo** é que o gradiente $\nabla \Im(\theta)$, da função custo, seja zero em $\theta = \theta^*$.

Como critério tem-se as **seguintes alternativas** de parada:
1 - quando a norma euclidiana da estimativa do **vetor gradiente** $\|\nabla \Im(\theta)\|$ atinge um valor suficientemente **pequeno**.

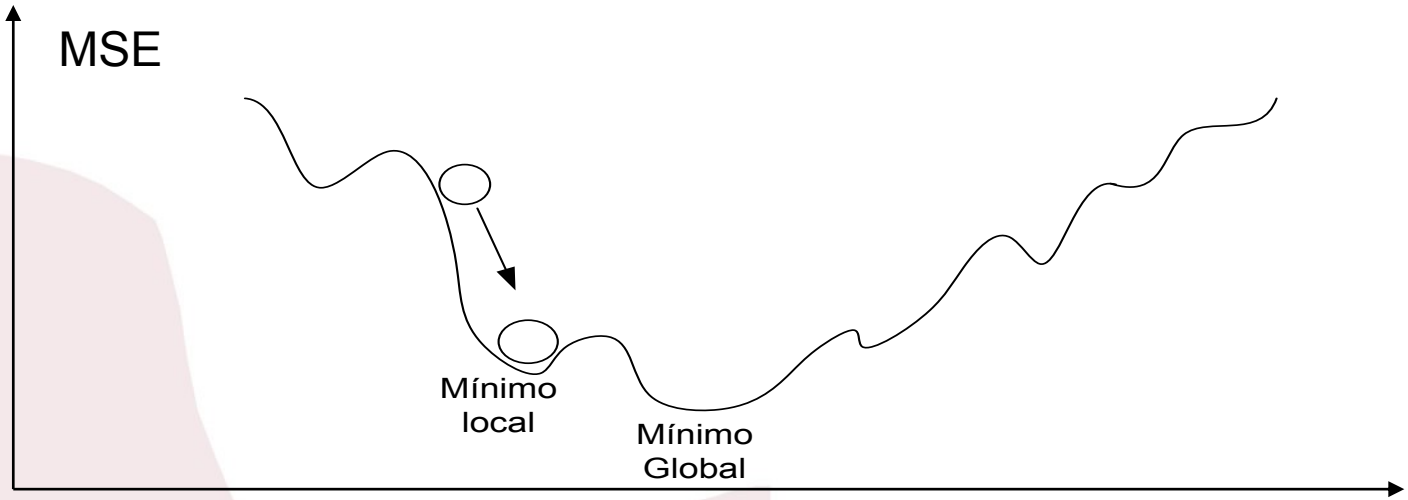
2 - quando a **variação** do **erro quadrático médio** (MSE) de uma época para outra atingir um valor suficientemente **pequeno**.

3 - quando o erro **quadrático médio** atingir um valor suficientemente **pequeno** ou seja, $\Im_{med}(\theta) \leq \varepsilon$ onde ε é um valor suficientemente pequeno.

Critérios de parada (cont.)

- Nota-se que se o critério é de valor **mínimo de MSE** então não se garante que o algoritmo irá atingir esse valor.

- Por outro lado, se o critério é o mínimo valor do **vetor gradiente** deve-se considerar que o algoritmo **termina no mínimo local** mais próximo.



Critérios de parada (cont.)

Outro critério de parada que pode ser usado em conjunto com um dos critérios anteriores é a **avaliação da capacidade de generalização** da rede após cada época de treinamento.

O processo de treinamento é interrompido antes que a capacidade de generalização da rede fique restrita.

Arquitetura da rede

A quantidade de neurônios na **camada de entrada** é dada pelo **problema** a ser abordado.

No entanto, a quantidade de neurônios **nas camadas de processamento** são características do **projeto**.

Aumentando-se o número de neurônios na **camada escondida** aumenta-se a capacidade de mapeamento não-linear da rede.

No entanto, quando esse número for **muito grande**, o modelo pode se sobre-ajustar aos dados, na presença de ruído nas amostras de treinamento. Diz-se que a rede está sujeita ao sobre-treinamento (***overfitting***).

Arquitetura da rede

Por outro lado, uma rede com **poucos neurônios** na camada escondida pode não ser capaz de realizar o mapeamento desejado, o que é denominado de ***underfitting***.

O *underfitting* também pode ser causado quando o **treinamento é interropido** de forma prematura.

Unidades intermediárias

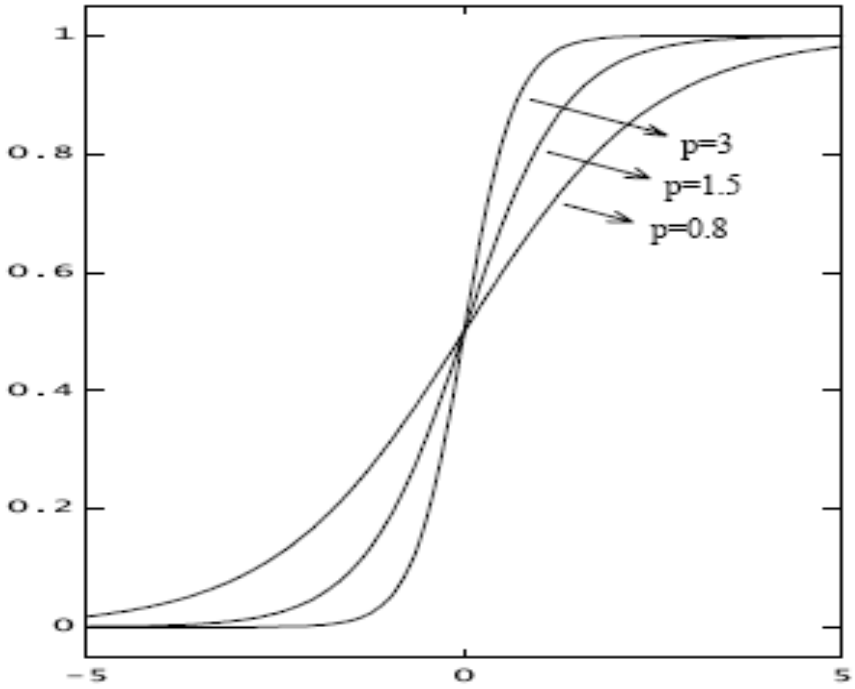
- Número de neurônios nas camadas intermediárias (cont.)
  - Depende de:
    - Número de exemplos de treinamento
    - Quantidade de ruído
    - Complexidade da função a ser aprendida
      - Distribuição estatística

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 de forma polinomial em relação ao número de entradas

Normalização dos dados de entrada

Uma característica das funções sigmoidais é a **saturação**, ou seja, para valores grandes de argumento, a função opera numa região de saturação.



É importante portanto trabalhar com valores de entrada que estejam contidos num intervalo que **não atinjam a saturação**, por exemplo, [0,1] ou [-1,1].

Inicialização dos vetores de pesos e bias

A eficiência do aprendizado em redes multicamadas depende da:

- especificação de arquitetura da rede,
- função de ativação,
- regra de aprendizagem, e
- valores iniciais dos vetores de pesos e bias.

Considerando-se que os três primeiros itens já foram definidos, verifica-se agora a **inicialização dos vetores de pesos e bias.**

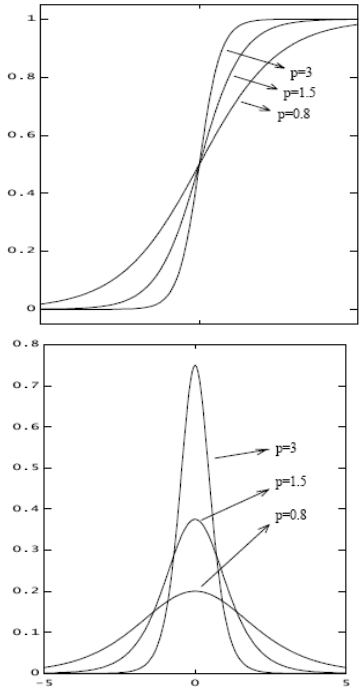
Inicialização aleatória dos pesos

A atualização de um peso entre duas unidades depende da derivada da função de ativação da unidade posterior e função de ativação da unidade anterior.

Por esta razão, é importante evitar escolhas de pesos iniciais que tornem as funções de ativação ou suas derivadas iguais a zero.

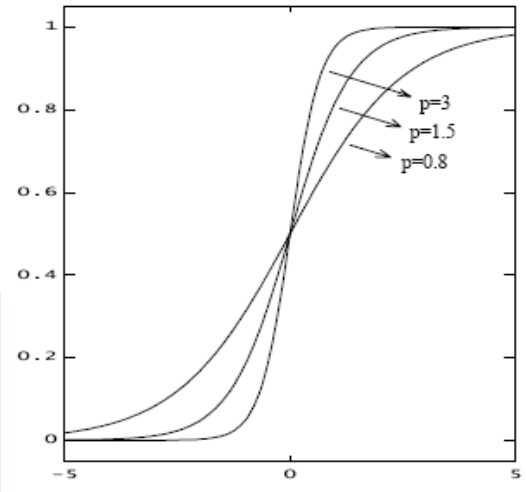
Os valores para os pesos iniciais não devem ser muito grandes, tal que as **derivadas** das **funções de ativação** tenham valores **muito pequenos** (região de saturação).

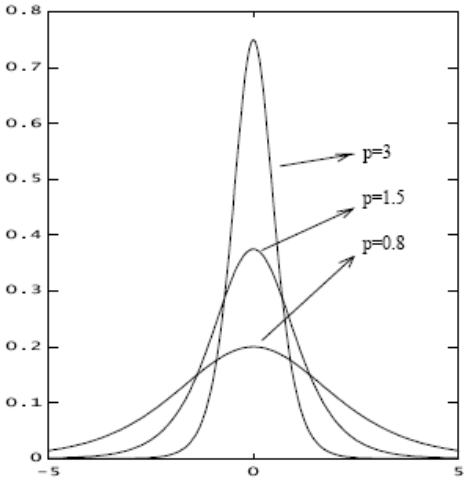
Por outro lado, se os **pesos iniciais** são **muito pequenos**, a soma pode cair perto de zero, onde o **aprendizado** é **muito lento**.



Inicialização aleatória dos pesos

Um **procedimento comum** é inicializar os pesos e *bias* a valores randômicos entre -0.5 e 0.5, ou entre -1 e 1. Os valores podem ser positivos ou negativos porque os pesos finais após o treinamento também podem ser positivos ou negativos.





Bibiografia básica

Simon Haykin – Redes Neurais – Princípios e Prática, 2a. Edição, Ed. Artmed: Bookman, Porto Alegre, 1999.

Braga, A. P., Carvalho, A. C. P. L., Ludermir, T. B. Redes Neurais Artificiais: teoria e aplicações. LTC - Livros Técnicos e Científico, 2ª edição, 2007 p.260.

Fontes

Site da Profa. **Teresa Ludermir** / CIN - UFPE:
http://www.cin.ufpe.br/~tbl

Site do Prof. **José Hiroki Saito** – DC / UFSCAR:
http://www.dc.ufscar.br/~saito/download/topicos-pis/

Site do Prof. **André de Carvalho** – ICMC / USP:
http://www.icmc.usp.br/~andre/research/neural/

Site do Prof. **Yaser Abu-Mostafa**
http://work.caltech.edu/telecourse.html