INSTITUTO FEDERAL
Ceará
Programa de Pós-Graduação
em Ciência da Computação

05 Funções de Ativação e Redes Neurais de uma Camada

REDES NEURAIS ARTIFICIAIS

PPGCC – 2022.1

Prof. Saulo Oliveira <saulo.oliveira@ifce.edu.br>

FUNÇÕES DE ATIVAÇÃO

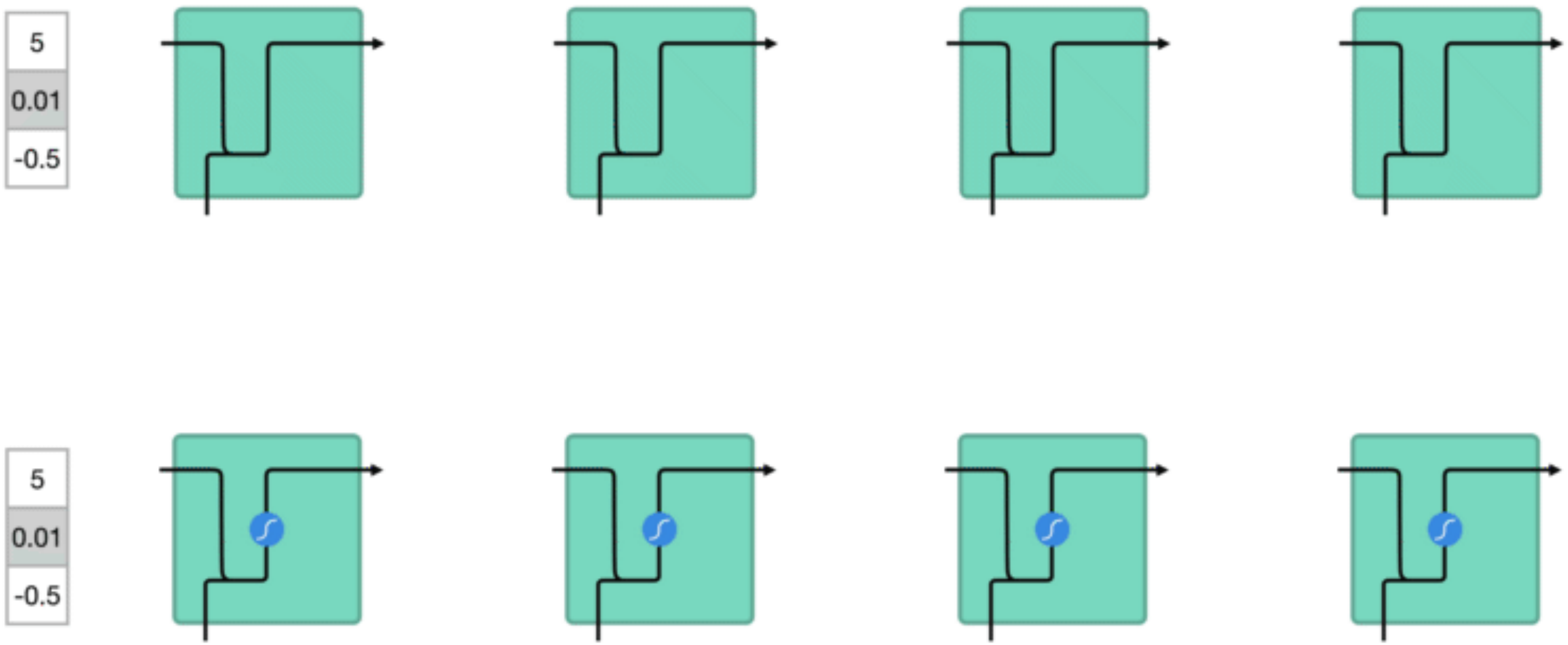
IMPORTÂNCIA

- As funções de ativação permitem que pequenas mudanças nos pesos e viés causem apenas uma pequena alteração na saída do neurônio (e da rede);

- Eles permitem a introdução de recursos não lineares na rede. Assim, as funções de ativação são cruciais para o modelo de rede neural aprender e entender funções não lineares complexas;

- Sem funções de ativação, as saída são apenas funções lineares simples (Alô, Adaline!). A complexidade das funções lineares é limitada. Logo, a capacidade de aprender mapeamentos de funções complexas a partir de dados é baixa;

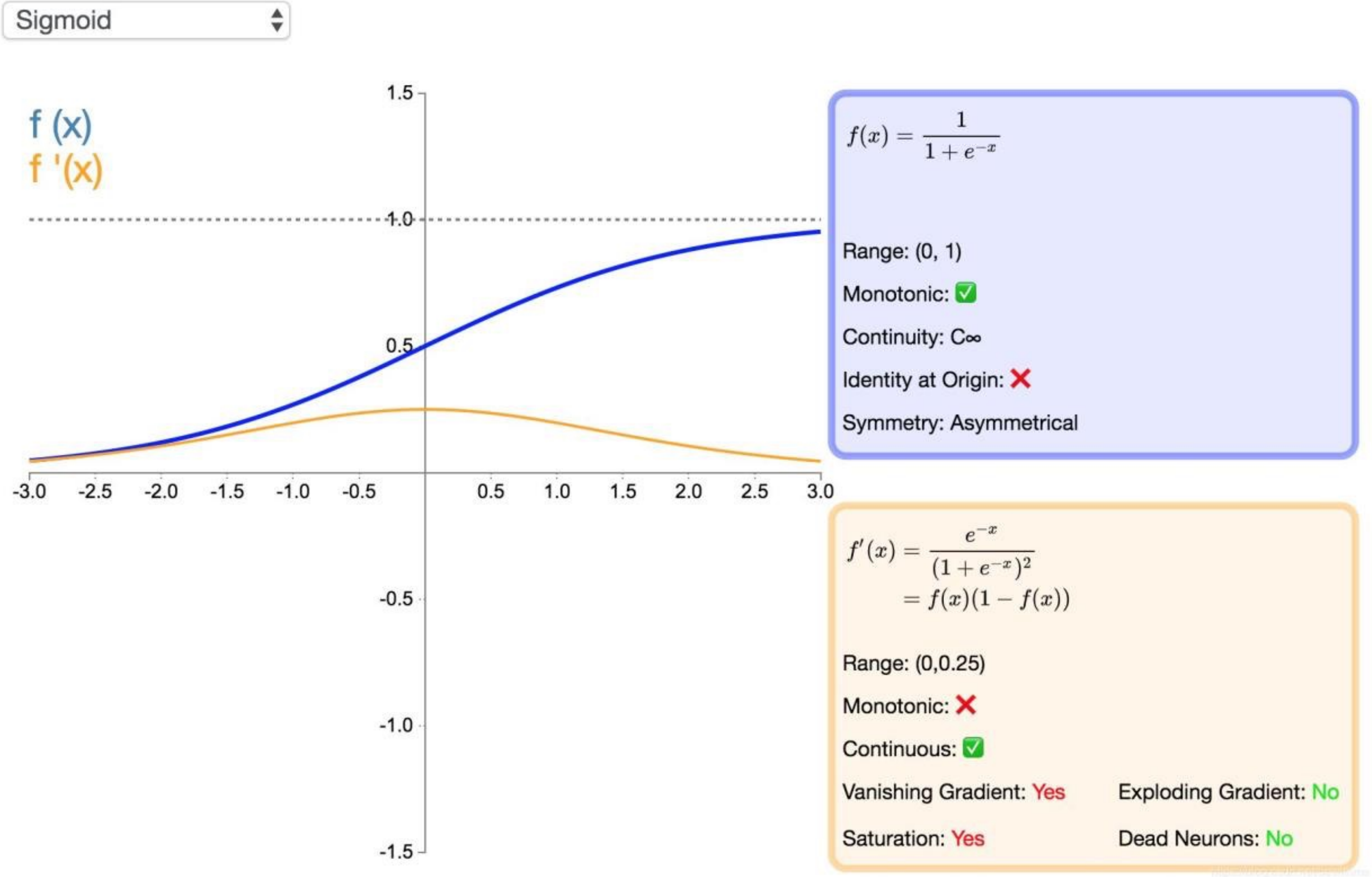
IMPORTÂNCIA



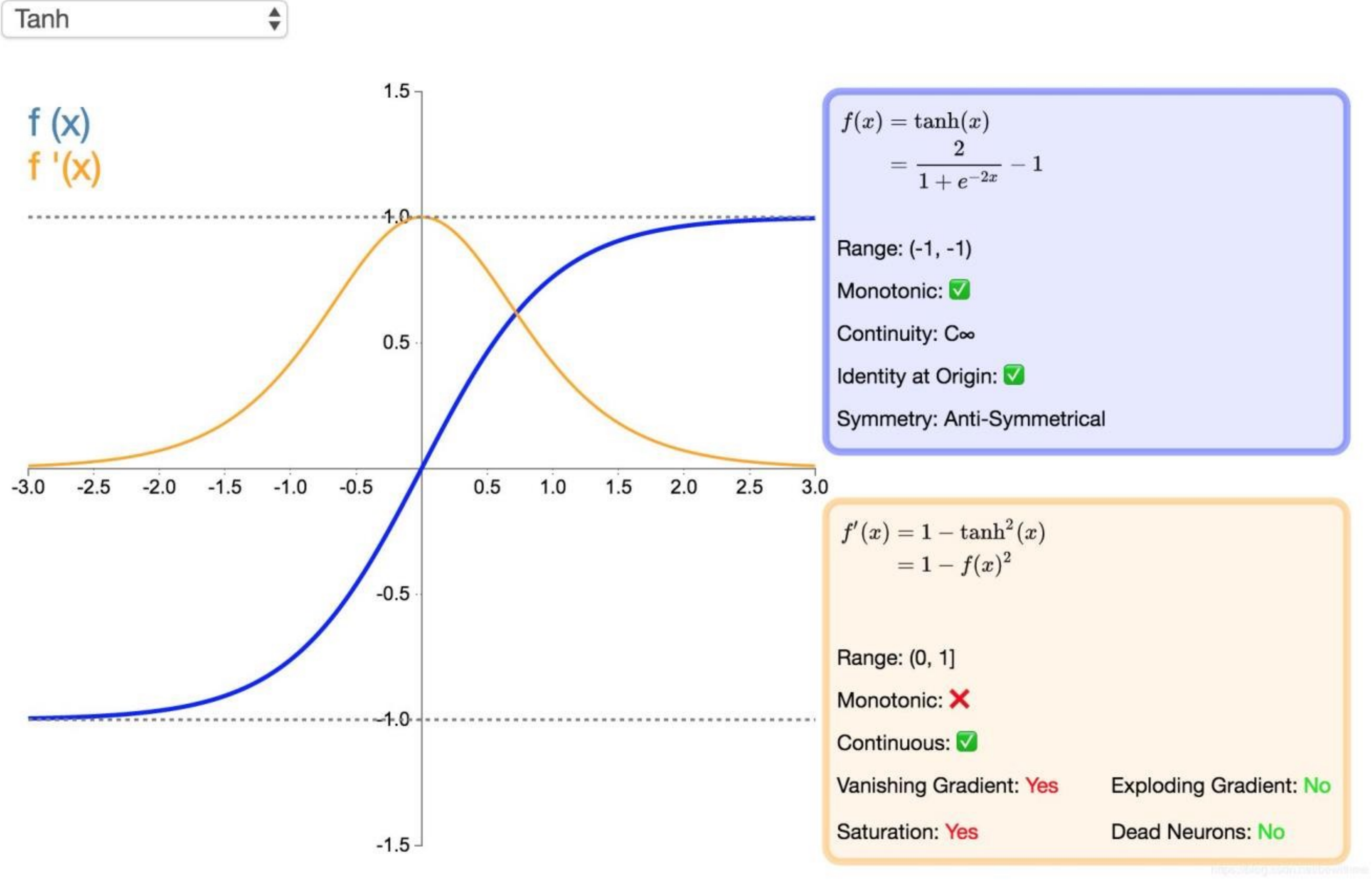
https://laptrinhx.com/introduction-to-neural-networks-and-their-key-elements-part-c-activation-functions-layers-1195957448/

AS FAMOSINHAS

Sigmoid



Tangente hiperbólica



https://laptrinhx.com/introduction-to-neural-networks-and-their-key-elements-part-c-activation-functions-layers-1195957448/

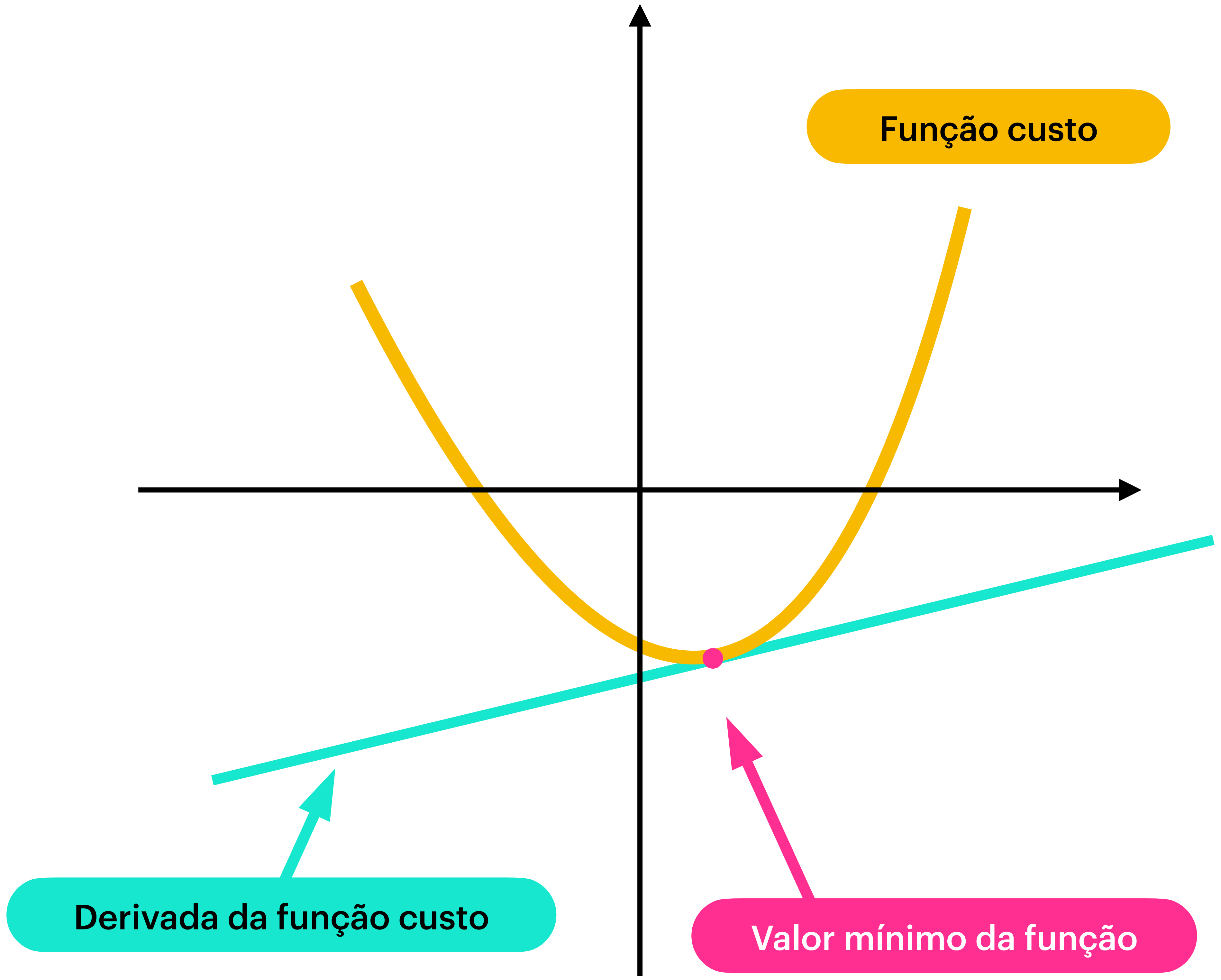
TREINANDO UM ADALINE (Recap)

- Premissa básica: achar o vetor de pesos que minimize o TODOS os erros;
- Forma de calcular o erro continua a mesma, i.e., $e = d - y = d - \varphi\left(\mathbf{x}^\intercal \mathbf{w}\right)$.
- Agregar TODOS os erros em uma função só:

$$J(\mathbf{e}) = \sum_i e_i^2.$$

$$J(\mathbf{w}) = \sum_i \left(d_i - \varphi(\mathbf{x}_i^\intercal \mathbf{w})\right)^2.$$

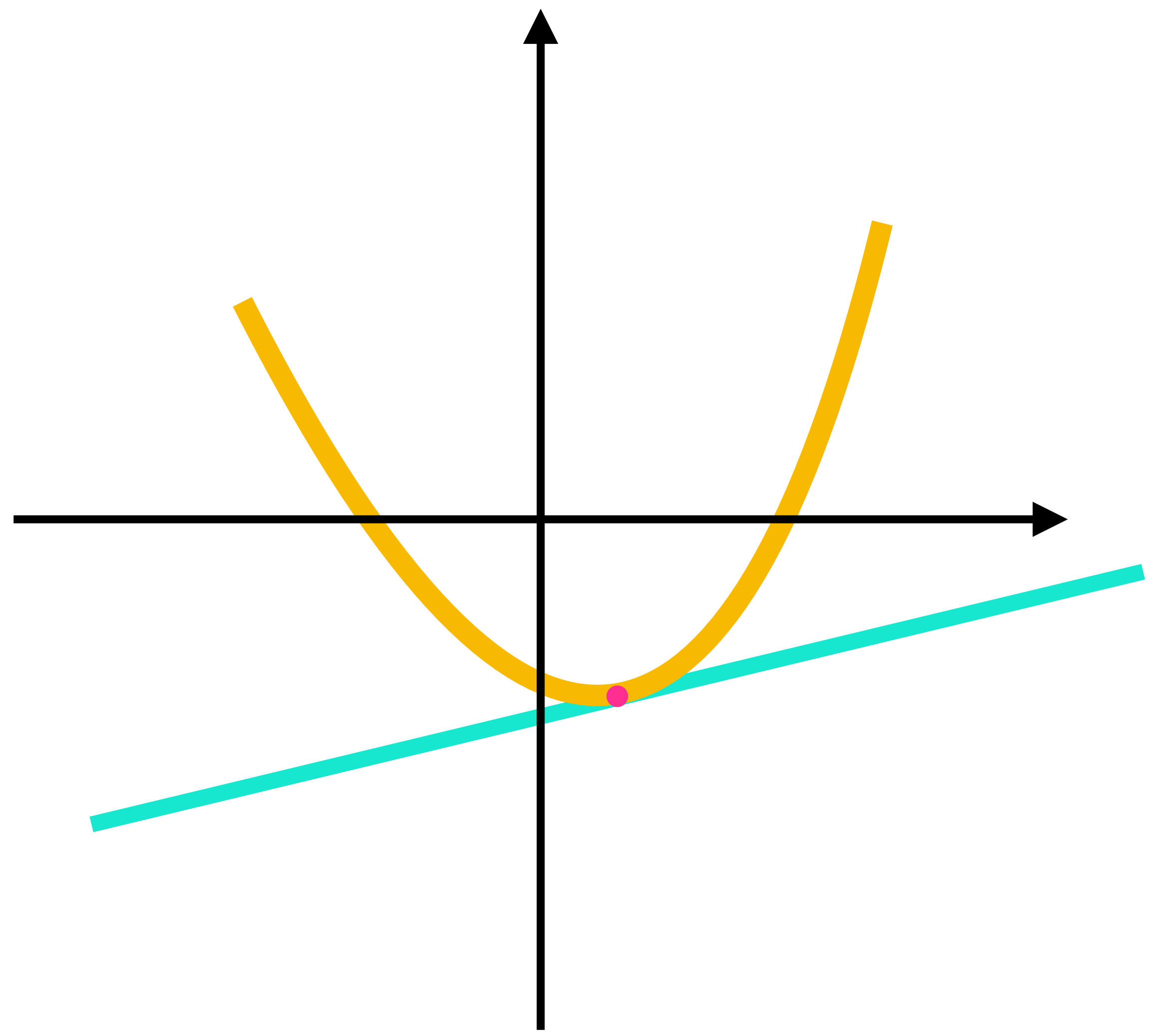
$$\boxed{\mathbf{w}^\star = \min_{\mathbf{w}} J(\mathbf{w})}$$



TREINANDO UM ADALINE (Recap)

- Com o intuito de minimizar $J(\mathbf{w})$, é necessário que seja obtido a direção do ajuste a ser aplicado no vetor de pesos para aproximar a solução do mínimo de $J(\mathbf{w})$;
- De modo iterativo, iremos realizar $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \Delta\mathbf{w}$;
- Para tal, usa-se o **gradiente** da função $J(\mathbf{w})$ no ponto $\mathbf{w}^{(t)}$;
- Sabe-se que o gradiente possui a mesma direção da maior variação do erro (aponta na direção do crescimento da função). Portanto, o ajuste deve ocorrer na direção contrária do gradiente. Logo a variação dos pesos pode ser descrita como:

$$\Delta\mathbf{w} \propto -\nabla\mathbf{w}.$$



TREINANDO UM ADALINE (Pulo do 🐈)

- Sabe-se que a função J depende de e, bem como sabe-se que e depende de y e, ainda, sabe-se que y depende de φ que depende de $\mathbf{w}$. Logo, regra da cadeia:

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial e_i}\frac{\partial e_i}{\partial y_i}\frac{\partial y_i}{\partial w_i}, \qquad \text{em que} \qquad J(\mathbf{w}) = \frac{1}{2}\sum_i \left(d_i - \varphi(\mathbf{x}_i^\intercal \mathbf{w})\right)^2.$$

- Quanto valem os termos abaixo?

a) $\dfrac{\partial J}{\partial e_i} = 2e_i$ b) $\dfrac{\partial e_i}{\partial y_i} = -\varphi'(\mathbf{x}_i^\intercal \mathbf{w})$ c) $\dfrac{\partial y_i}{\partial w_i} = \mathbf{x}_i$ d) $\dfrac{\partial J}{\partial w_i} =$

- Adicionalmente:

a) $\text{sigmoid}'(u) = \text{sigmoid}(u)(1 - \text{sigmoid}(u))$ b) $\tanh'(u) = 1 - \tanh^2(u)$

TREINANDO UM ADALINE (Pulo do 🐈)

- Sabe-se que a função J depende de e, bem como sabe-se que e depende de y e, ainda, sabe-se que y depende de φ que depende de $\mathbf{w}$. Logo, regra da cadeia:

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial e_i}\frac{\partial e_i}{\partial y_i}\frac{\partial y_i}{\partial w_i}, \qquad \text{em que} \qquad J(\mathbf{w}) = \frac{1}{2}\sum_i \left(d_i - \varphi(\mathbf{x}_i^\intercal \mathbf{w})\right)^2.$$

- Quanto valem os termos abaixo?

a) $\frac{\partial J}{\partial e_i} = 2e_i$ b) $\frac{\partial e_i}{\partial y_i} = -\,\varphi'(\mathbf{x}_i^\intercal \mathbf{w})$ c) $\frac{\partial y_i}{\partial w_i} = \mathbf{x}_i$ d) $\frac{\partial J}{\partial w_i} = -\,e_i\,\varphi'(\mathbf{x}_i^\intercal \mathbf{w})\,x_i$

- Adicionalmente:

a) $\mathrm{sigmoid}'(u) = \mathrm{sigmoid}(u)(1 - \mathrm{sigmoid}(u))$ b) $\tanh'(u) = 1 - \tanh^2(u)$

TREINANDO UM ADALINE (Pulo do 🐈)

- Considerando que

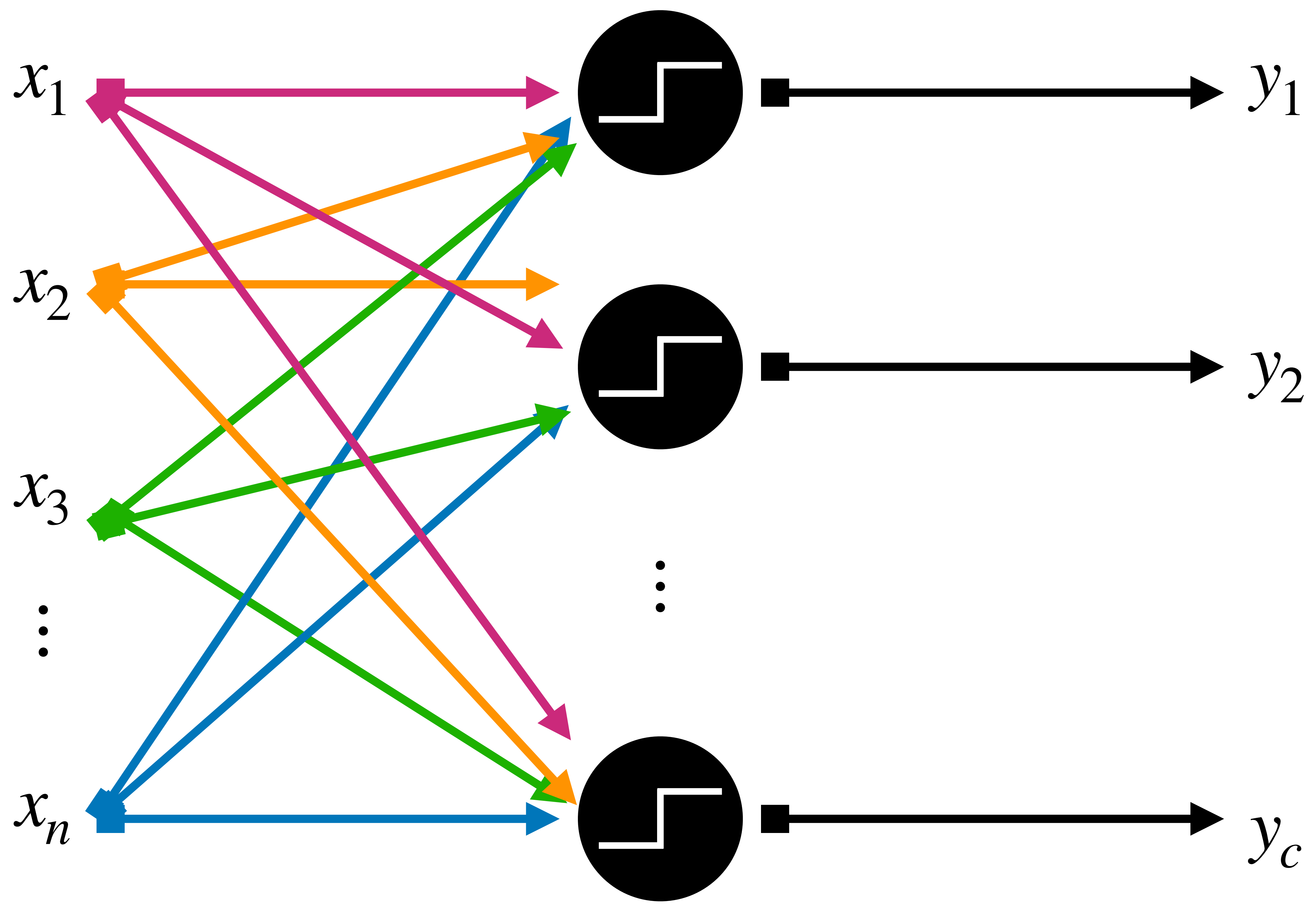
a) $\dfrac{\partial J}{\partial e_i} = 2e_i$ b) $\dfrac{\partial e_i}{\partial y_i} = -\,\varphi'(\mathbf{x}_i^\intercal \mathbf{w})$ c) $\dfrac{\partial y_i}{\partial w_i} = \mathbf{x}_i$ d) $\dfrac{\partial J}{\partial w_i} = -\,e_i\,\varphi'(\mathbf{x}_i^\intercal \mathbf{w})\,x_i$

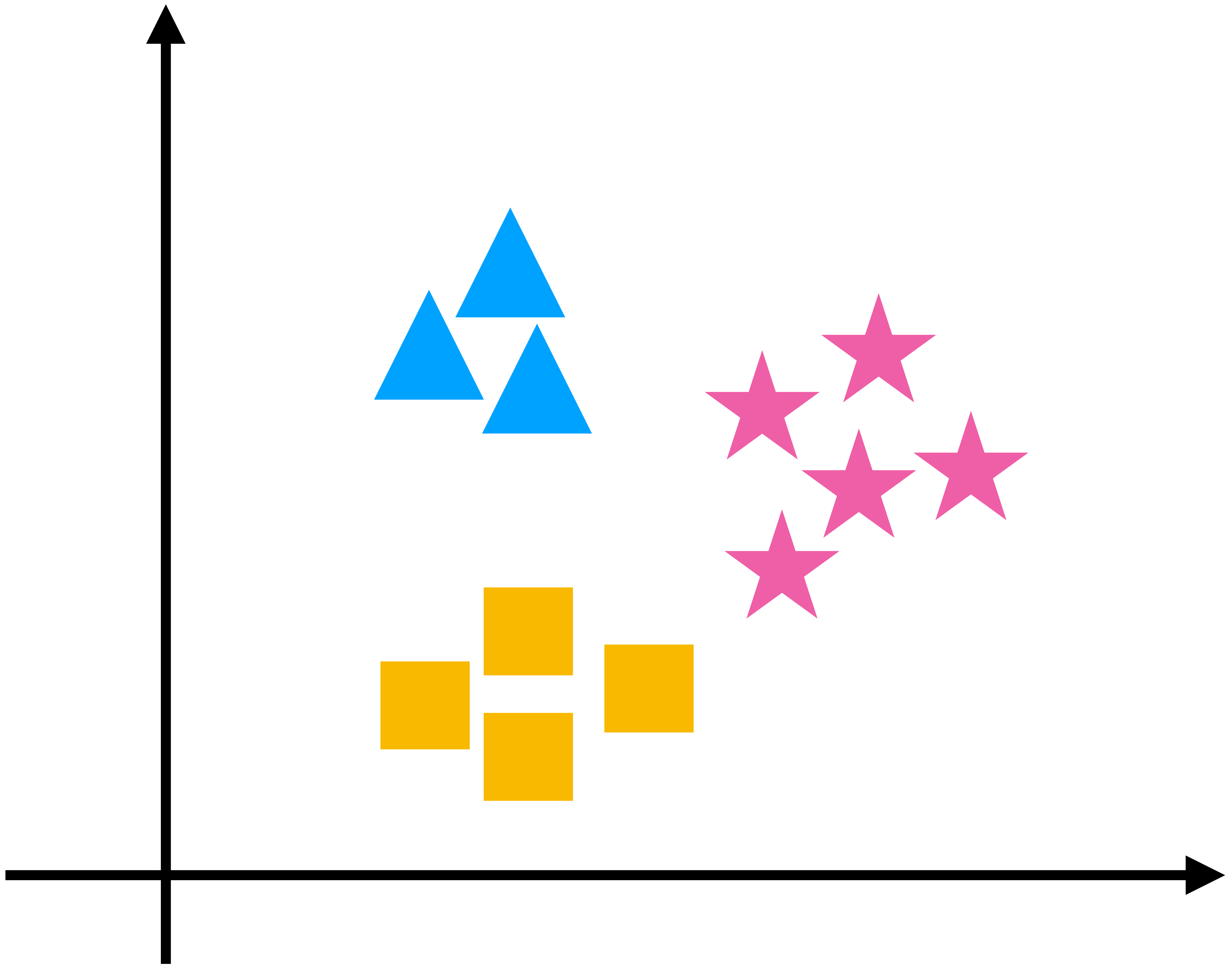
- Considerando também a regra de aprendizagem $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \Delta\mathbf{w}$;
- Considerando que $\Delta\mathbf{w} \propto \nabla J$ e que $\dfrac{\partial J}{\partial \mathbf{w}} = -\,e\,\varphi'(\mathbf{x}^\intercal \mathbf{w})\,\mathbf{x}$, podemos reescrever a regra de atualização da seguinte forma:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta e^{(t)} \varphi'\left(\mathbf{x}^\intercal \mathbf{w}^{(t)}\right)\mathbf{x}.$$

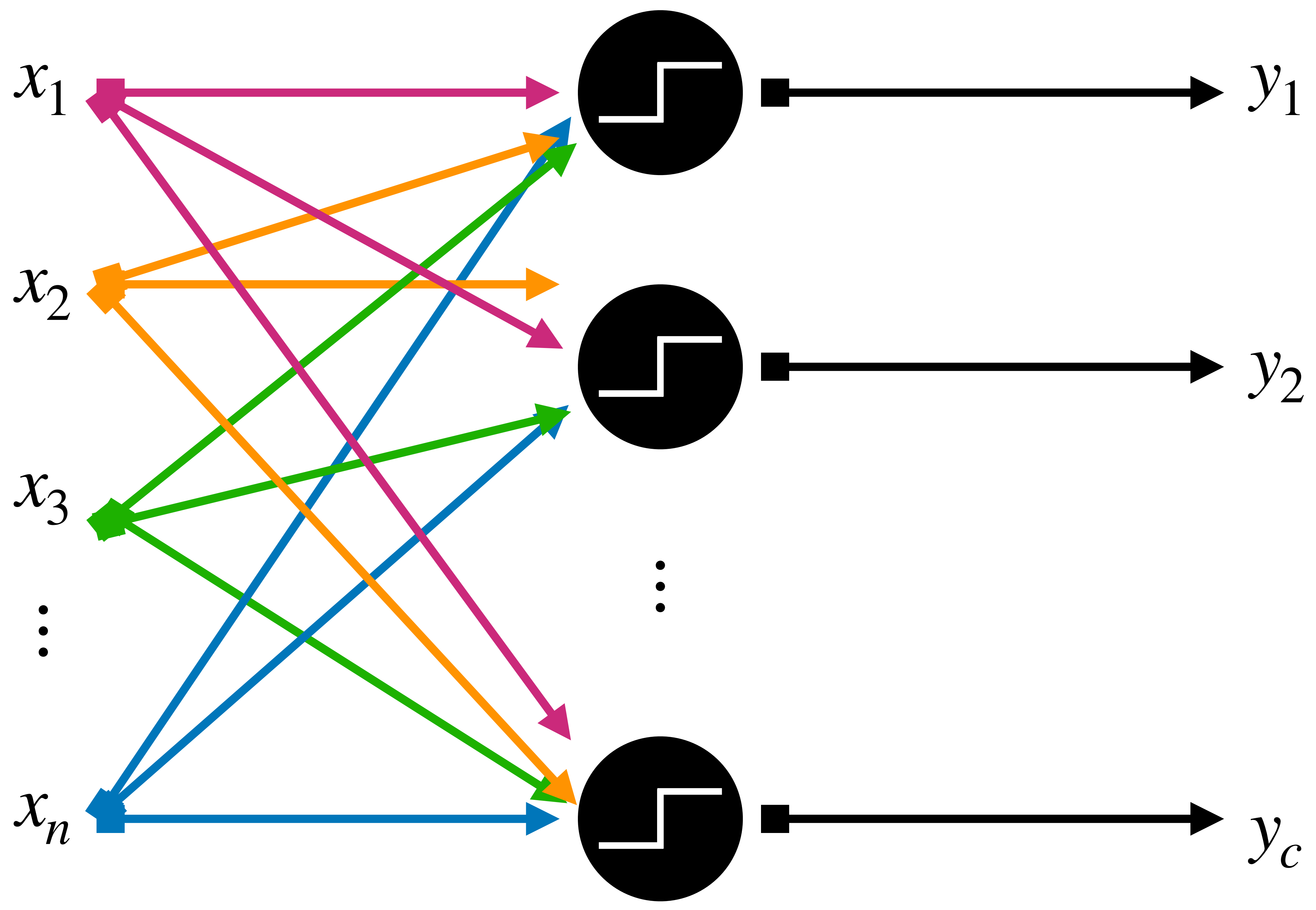
COMO FAZER O PERCEPTRON CLASSIFICAR > 2 CLASSES?

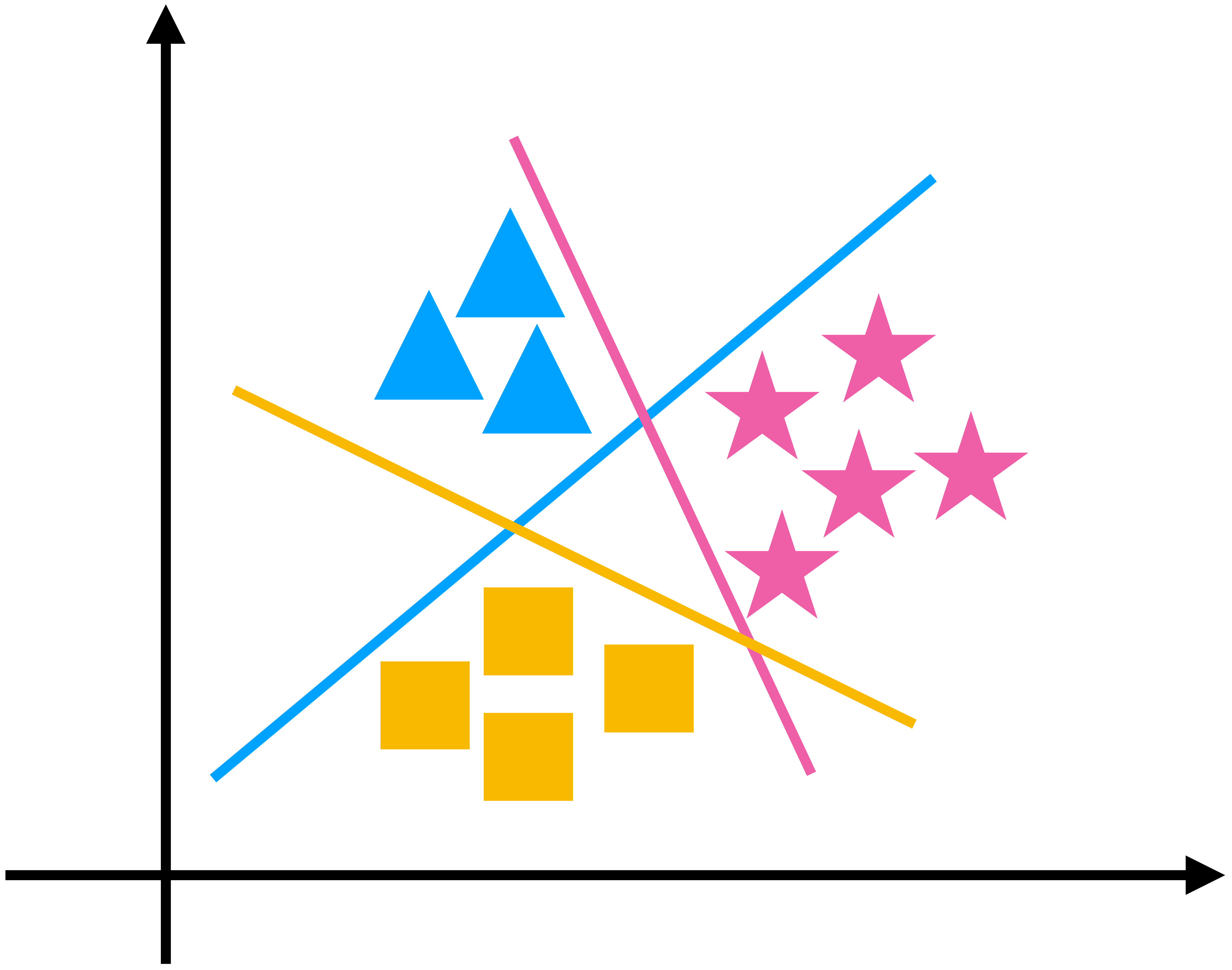
REDE PERCEPTRON



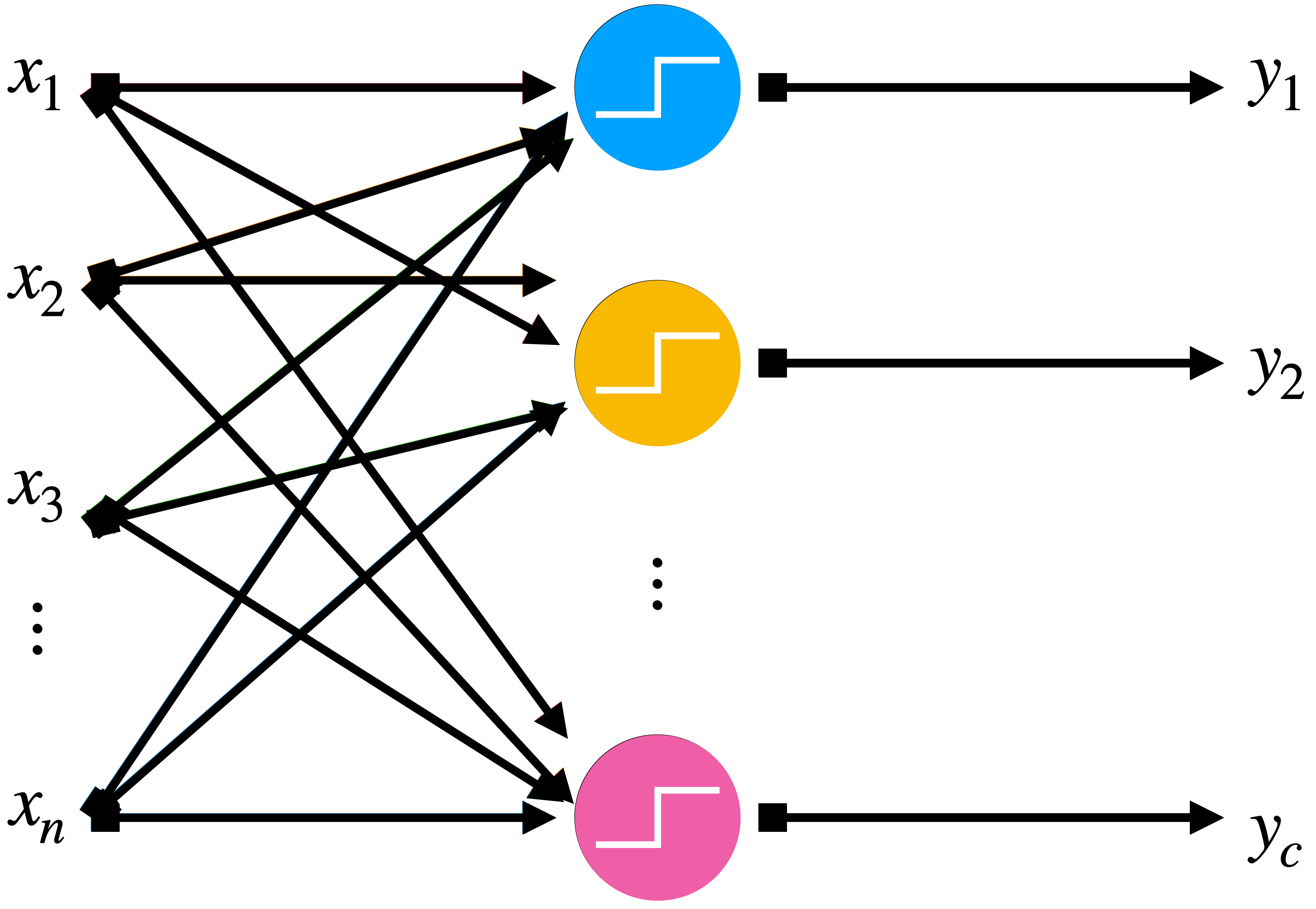


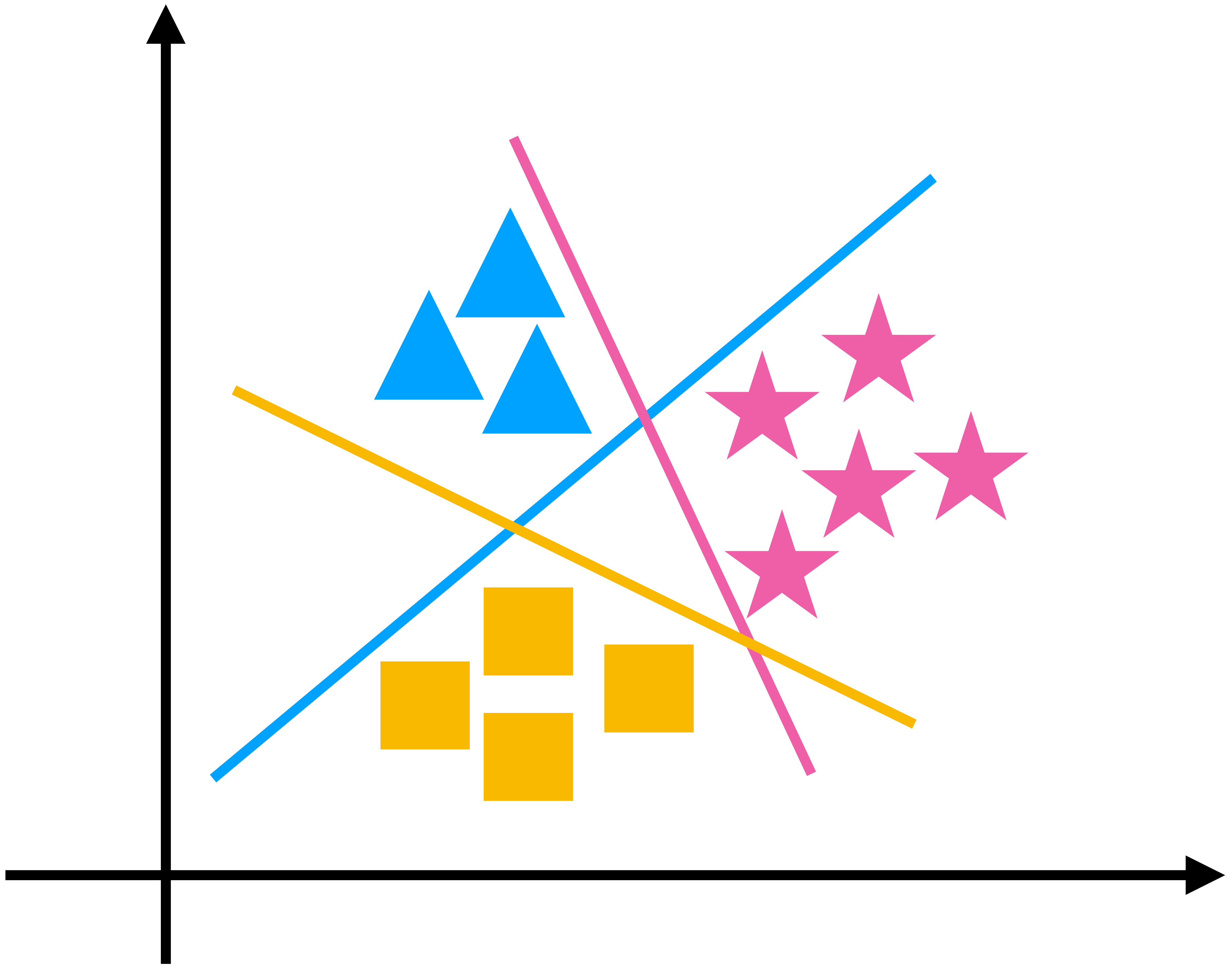
REDE PERCEPTRON



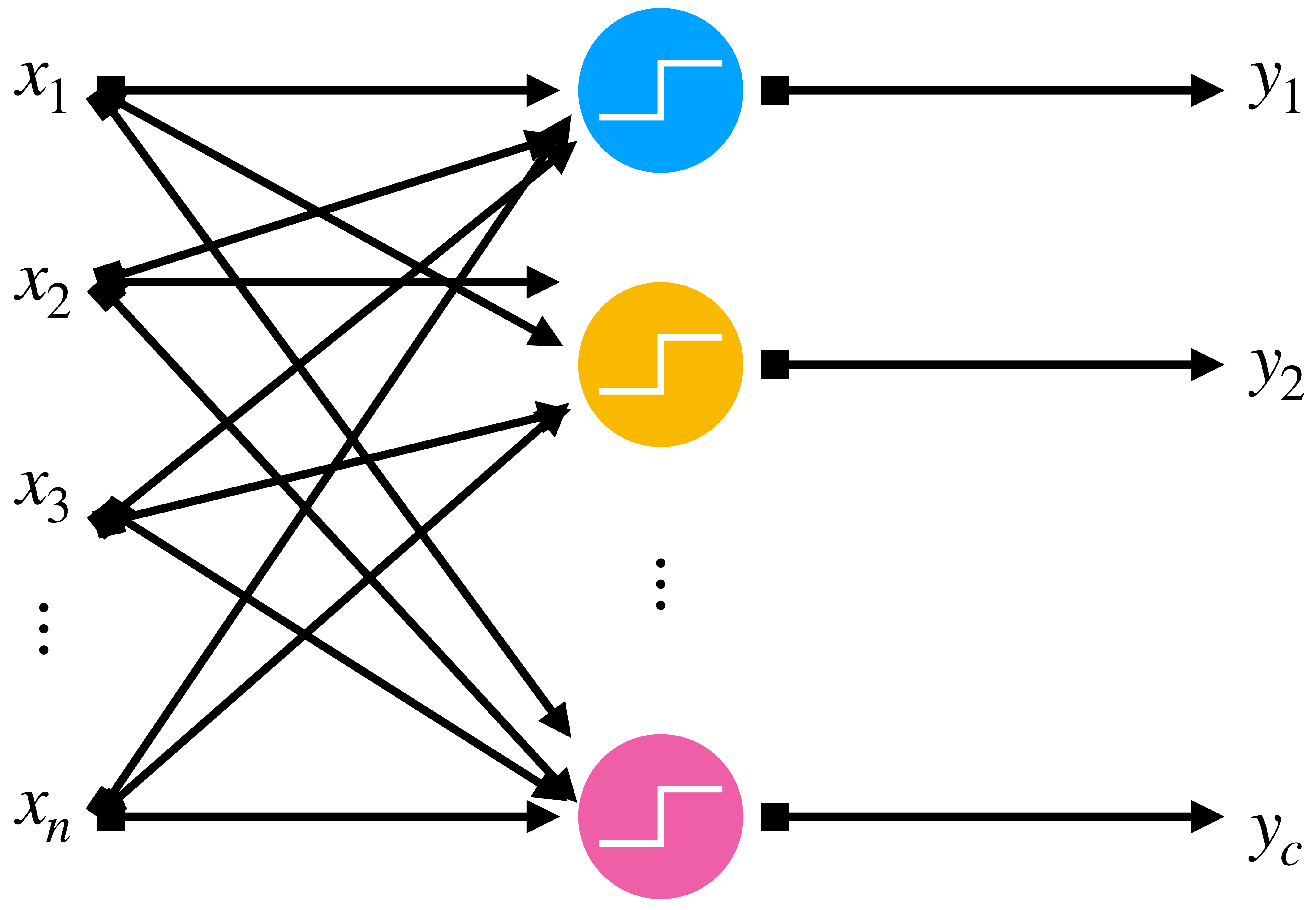


REDE PERCEPTRON





REDE PERCEPTRON

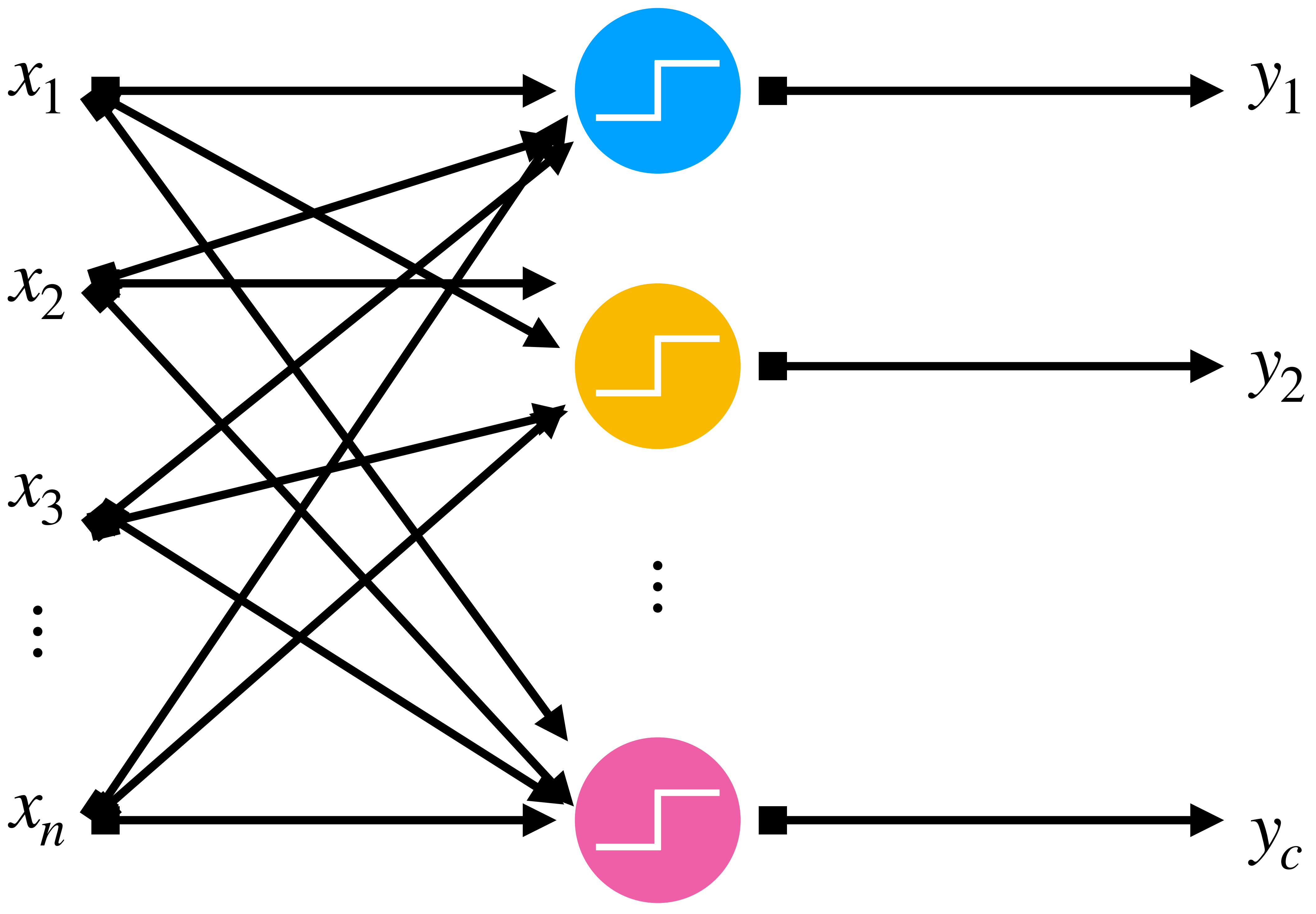


- Nessa arquitetura, o número de neurônios é igual ao número de classes;
- A classificação de cada padrão é o resultado da combinação de todos os neurônios. No entanto, só um neurônio deve responder;

$$y_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}, y_2 = \begin{bmatrix}0\\1\\0\end{bmatrix} \text{ e } y_3 = \begin{bmatrix}0\\0\\1\end{bmatrix}.$$

- A quantidade de pesos aumenta :);
- Rede Adaline usa a mesma estrutura.

REDE PERCEPTRON



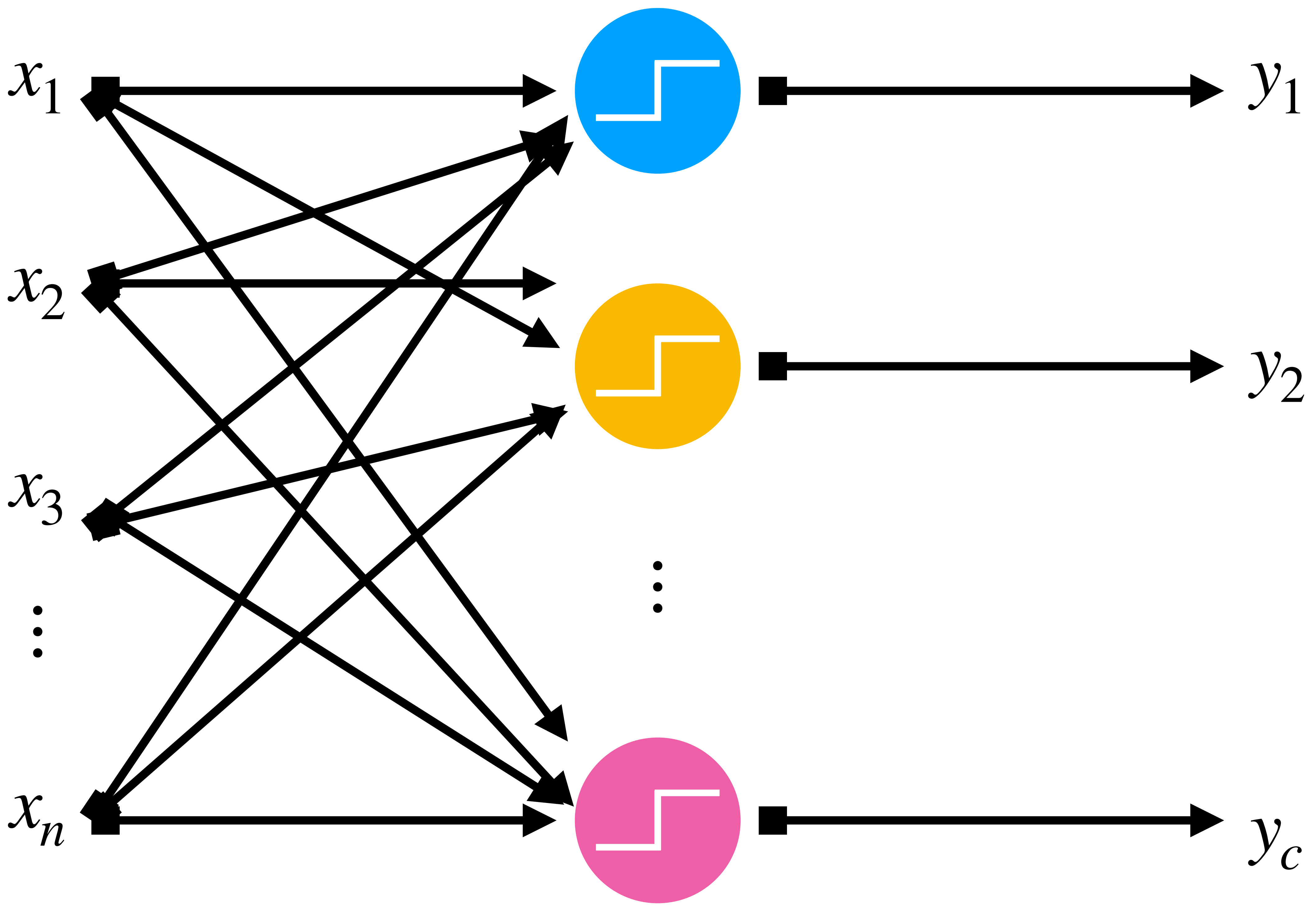
- Nessa arquitetura, o número de neurônios é igual ao número de classes;
- A classificação de cada padrão é o resultado da combinação de todos os neurônios. No entanto, só um neurônio deve responder;

$$y_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, y_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ e } y_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

- A quantidade de pesos aumenta :);
- Rede Adaline usa a mesma estrutura.

- $y_1 = \varphi\left(\mathbf{x}^\intercal \mathbf{w}_1\right) = \varphi\left(\sum_{i=1}^{n} x_i w_{1,i}\right)$

REDE PERCEPTRON



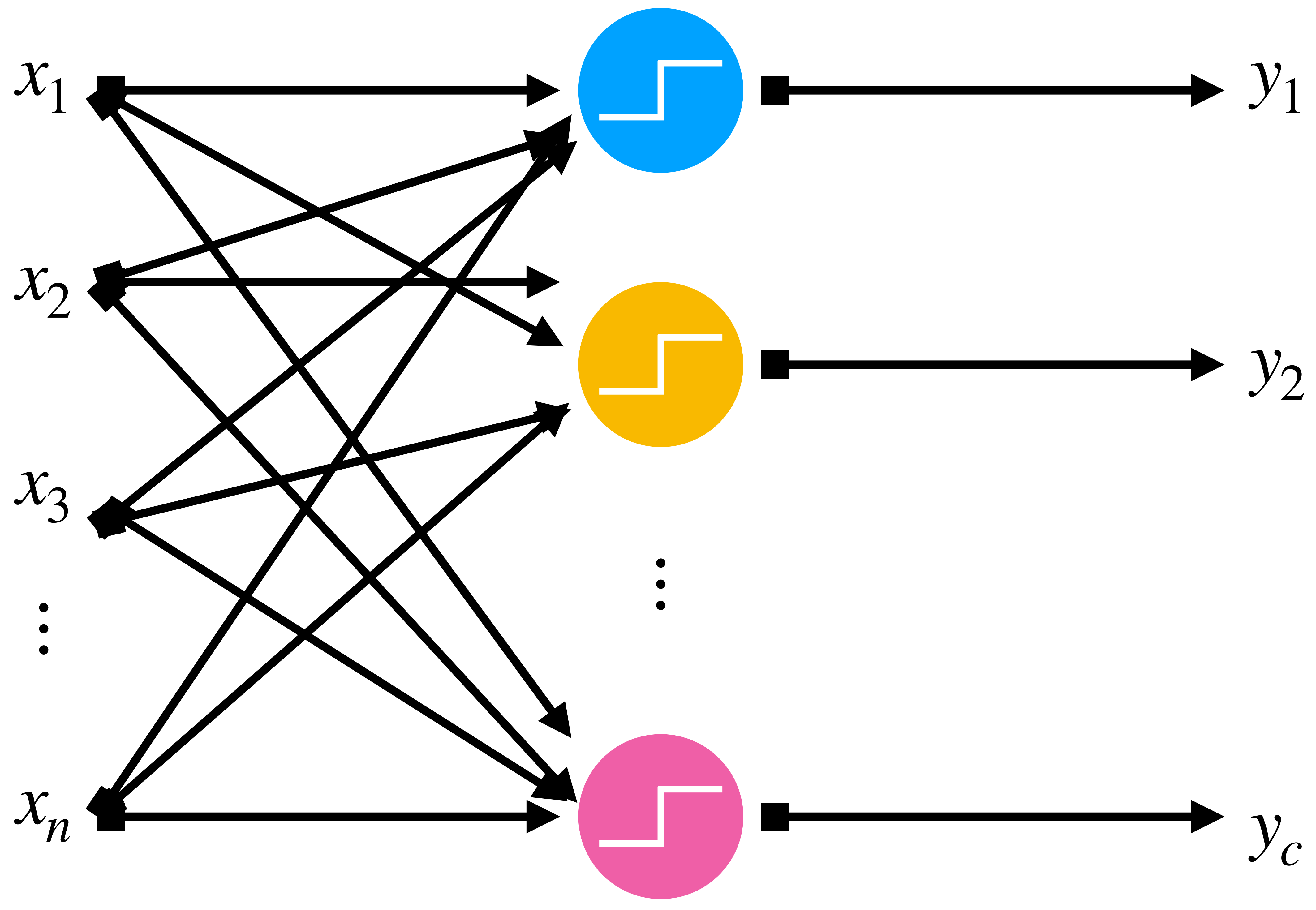
- Nessa arquitetura, o número de neurônios é igual ao número de classes;
- A classificação de cada padrão é o resultado da combinação de todos os neurônios. No entanto, só um neurônio deve responder;

$$y_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, y_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ e } y_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

- A quantidade de pesos aumenta :);
- Rede Adaline usa a mesma estrutura.

- $y_1 = \varphi\left(\mathbf{x}^\intercal \mathbf{w}_1\right) = \varphi\left(\sum_{i=1}^{n} x_i w_{1,i}\right)$
- $y_2 = \varphi\left(\mathbf{x}^\intercal \mathbf{w}_2\right) = \varphi\left(\sum_{i=1}^{n} x_i w_{2,i}\right)$

REDE PERCEPTRON



- Nessa arquitetura, o número de neurônios é igual ao número de classes;
- A classificação de cada padrão é o resultado da combinação de todos os neurônios. No entanto, só um neurônio deve responder;

$$y_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}, y_2 = \begin{bmatrix}0\\1\\0\end{bmatrix} \text{ e } y_3 = \begin{bmatrix}0\\0\\1\end{bmatrix}.$$

- A quantidade de pesos aumenta :);
- Rede Adaline usa a mesma estrutura.

- $y_1 = \varphi\left(\mathbf{x}^\intercal \mathbf{w}_1\right) = \varphi\left(\sum_{i=1}^{n} x_i w_{1,i}\right)$
- $y_2 = \varphi\left(\mathbf{x}^\intercal \mathbf{w}_2\right) = \varphi\left(\sum_{i=1}^{n} x_i w_{2,i}\right)$
- $y_3 = \varphi\left(\mathbf{x}^\intercal \mathbf{w}_3\right) = \varphi\left(\sum_{i=1}^{n} x_i w_{3,i}\right)$

Referências

- Richard O. Duda, Peter E. Hart, David G. Stork. **Pattern Classification**. John Wiley & Sons, 2012.
- Guilherme A. Barreto. **Introdução à Classificação de Padrões**. Grupo de Aprendizado de Máquinas – GRAMA, 2021.
- KDNugets. **Neural Networks with Numpy for Absolute Beginners — Part 2: Linear Regression**. https://www.kdnuggets.com/2019/03/neural-networks-numpy-absolute-beginners-part-2-linear-regression.html/2, março de 2019. Acessado em Março de 2022.