VINI CARIDÁ

Human-Centered Data & AI

Vinicius Caridá, Ph.D.

Data Science Manager, Itaú Unibanco

MBA Professor, FIAP

GDE – Machine Learning

@vinicius caridá @vfcarida @vinicius caridá @vfcarida @vinicius caridá @vfcarida

VINI CARIDÁ
Human-Centered Data & AI

VINI CARIDÁ
Human-Centered Data & AI

Redes Neurais

Recapitulando...

Regressão Linear

	Peso	Altura
 Pessoa 1	80 kg	163
 Pessoa 2	85 kg	168
 Pessoa 3	90 kg	175
 Pessoa 4	95 kg	188
 Pessoa 5	88 kg	175,4

$$\hat{y} = \beta_0 + \beta_1 X_1$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

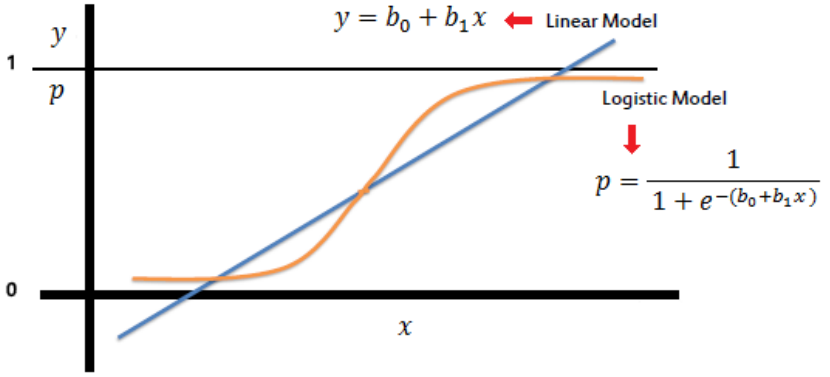
$\beta_0 = 17$

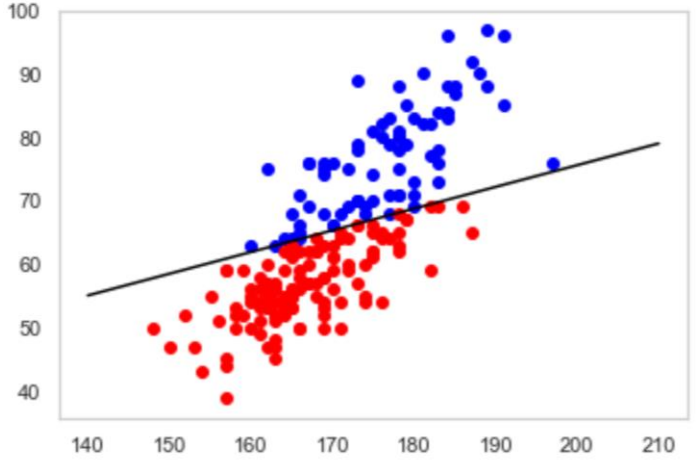
$\beta_1 = 1{,}8$

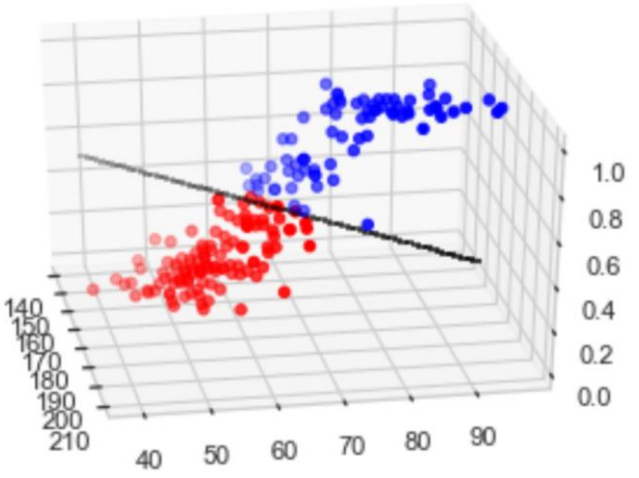
MSE = 6

$$\hat{y} = 17 + 1{,}8\, X_1$$

Redes Neurais – Feedforward

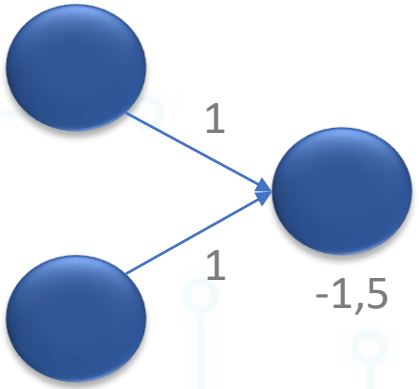




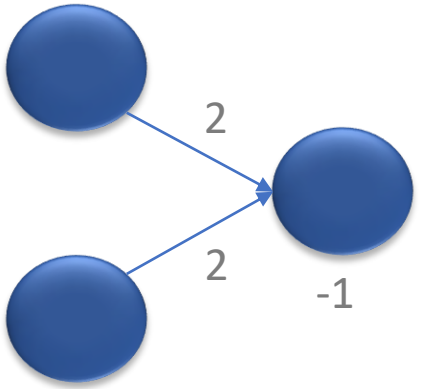


Redes Neurais

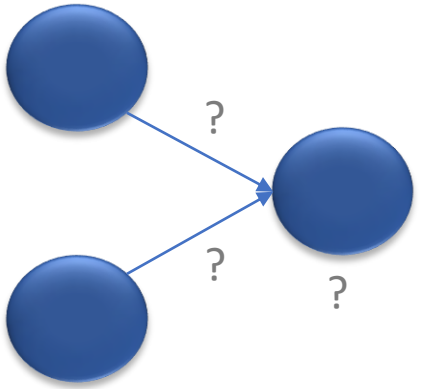
0	0	0
0	1	0
1	0	0
1	1	1



0	0	0
0	1	1
1	0	1
1	1	1



0	0	0
0	1	1
1	0	1
1	1	0



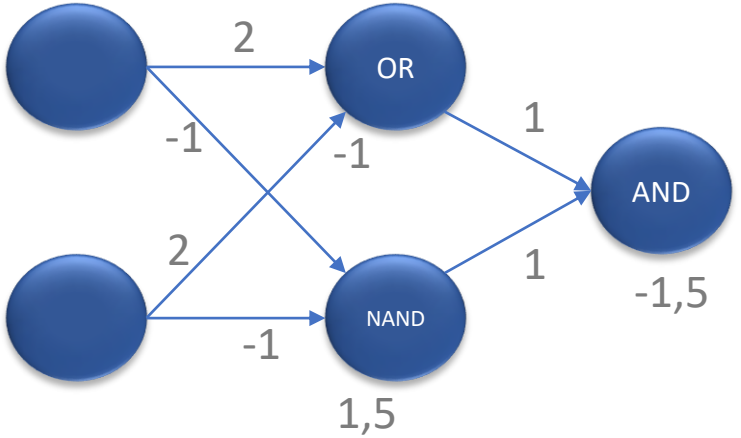
Função Degrau Padrão

Se >= 0 então 1

Se < 0 então 0

Redes Neurais

0	0	0
0	1	1
1	0	1
1	1	0

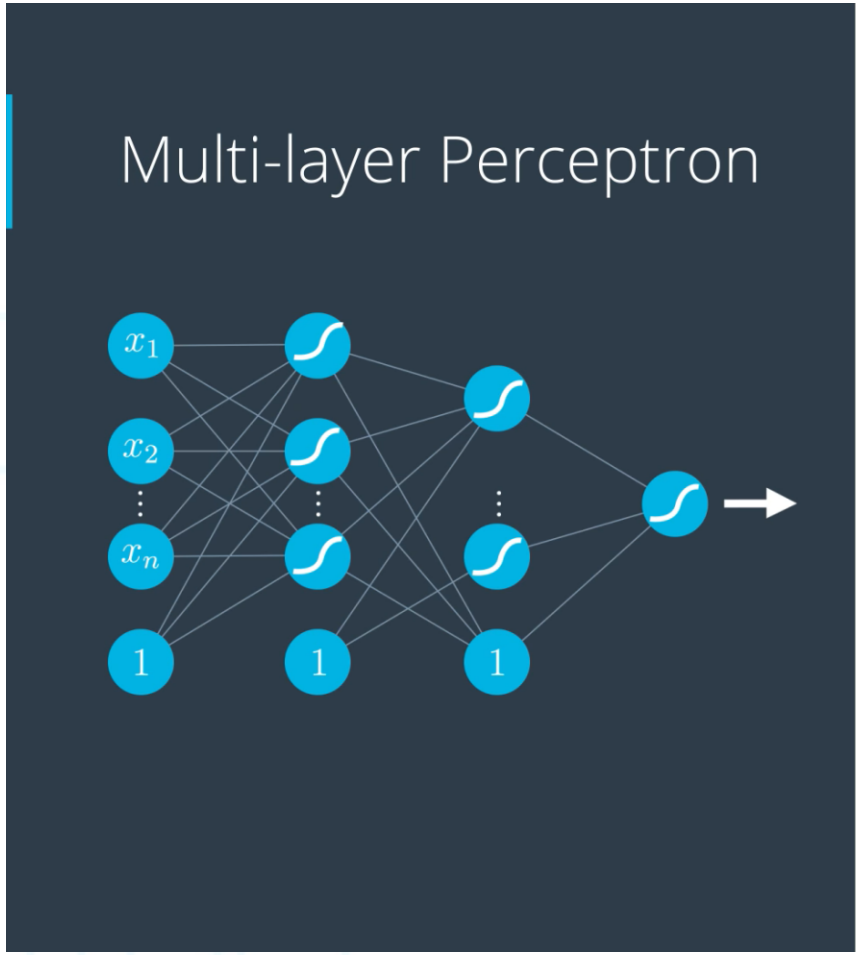


Função Degrau Padrão

Se >= 0 então 1

Se < 0 então 0

Redes Neurais – Backpropagation



PREDICTION

$$\hat{y} = \sigma W^{(3)} \circ \sigma W^{(2)} \circ \sigma \circ W^{(1)}(x)$$

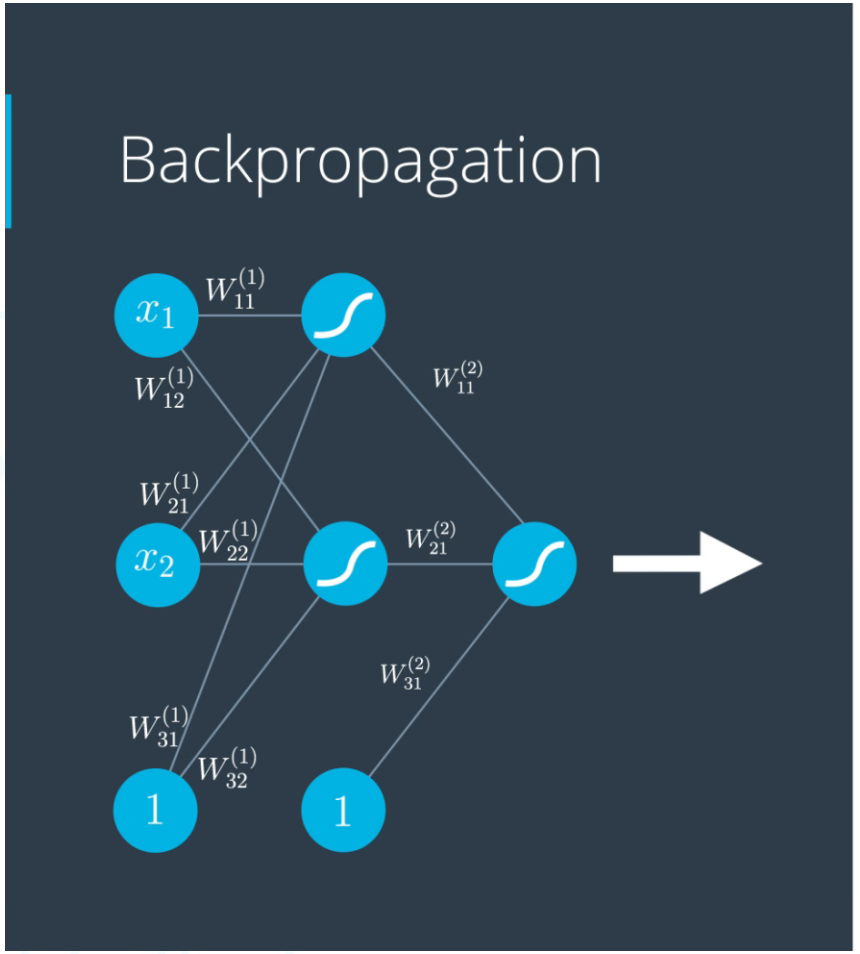
ERROR FUNCTION

$$E(W) = -\frac{1}{m}\sum_{i=1}^{m} y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)$$

GRADIENT OF THE ERROR FUNCTION

$$\nabla E = (..., \frac{\partial E}{\partial w_j^{(i)}}, ...)$$

Redes Neurais – Backpropagation



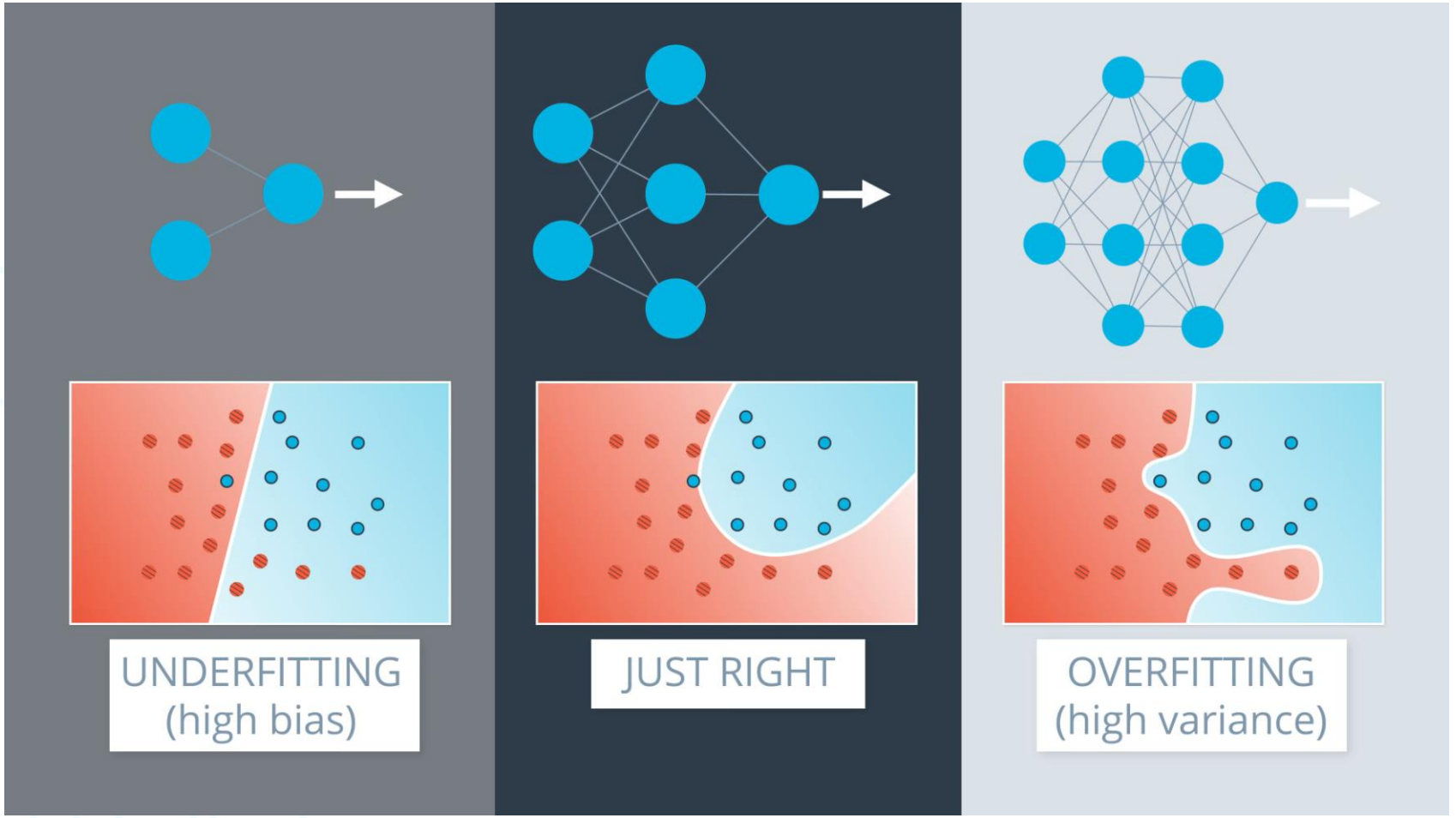
$$\hat{y} = \sigma W^{(2)} \circ \sigma \circ W^{(1)}(x)$$

$$W^{(1)} = \begin{pmatrix} W_{11}^{(1)} & W_{12}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} \\ W_{31}^{(1)} & W_{32}^{(1)} \end{pmatrix} \quad W^{(2)} = \begin{pmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{pmatrix}$$

$$\nabla E = \begin{pmatrix} \frac{\partial E}{\partial W_{11}^{(1)}} & \frac{\partial E}{\partial W_{12}^{(1)}} & \frac{\partial E}{\partial W_{11}^{(2)}} \\ \frac{\partial E}{\partial W_{21}^{(1)}} & \frac{\partial E}{\partial W_{22}^{(1)}} & \frac{\partial E}{\partial W_{21}^{(2)}} \\ \frac{\partial E}{\partial W_{31}^{(1)}} & \frac{\partial E}{\partial W_{32}^{(1)}} & \frac{\partial E}{\partial W_{31}^{(2)}} \end{pmatrix}$$

$$W_{ij}^{\prime(k)} \leftarrow W_{ij}^{(k)} - \alpha \frac{\partial E}{\partial W_{ij}^{(k)}}$$

Redes Neurais – Overfitting and Underfitting

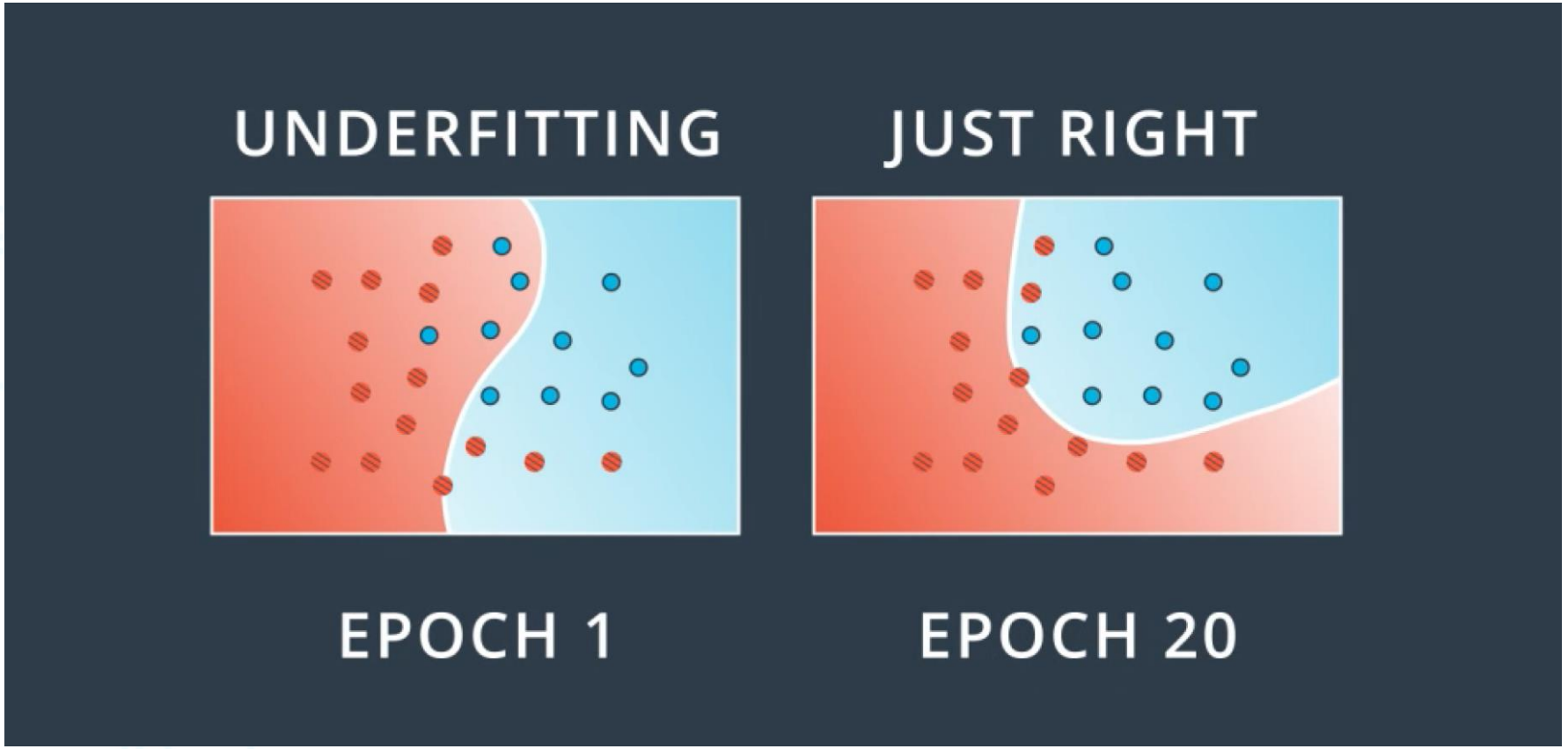


VINI CARIDÁ
Human-Centered Data & AI

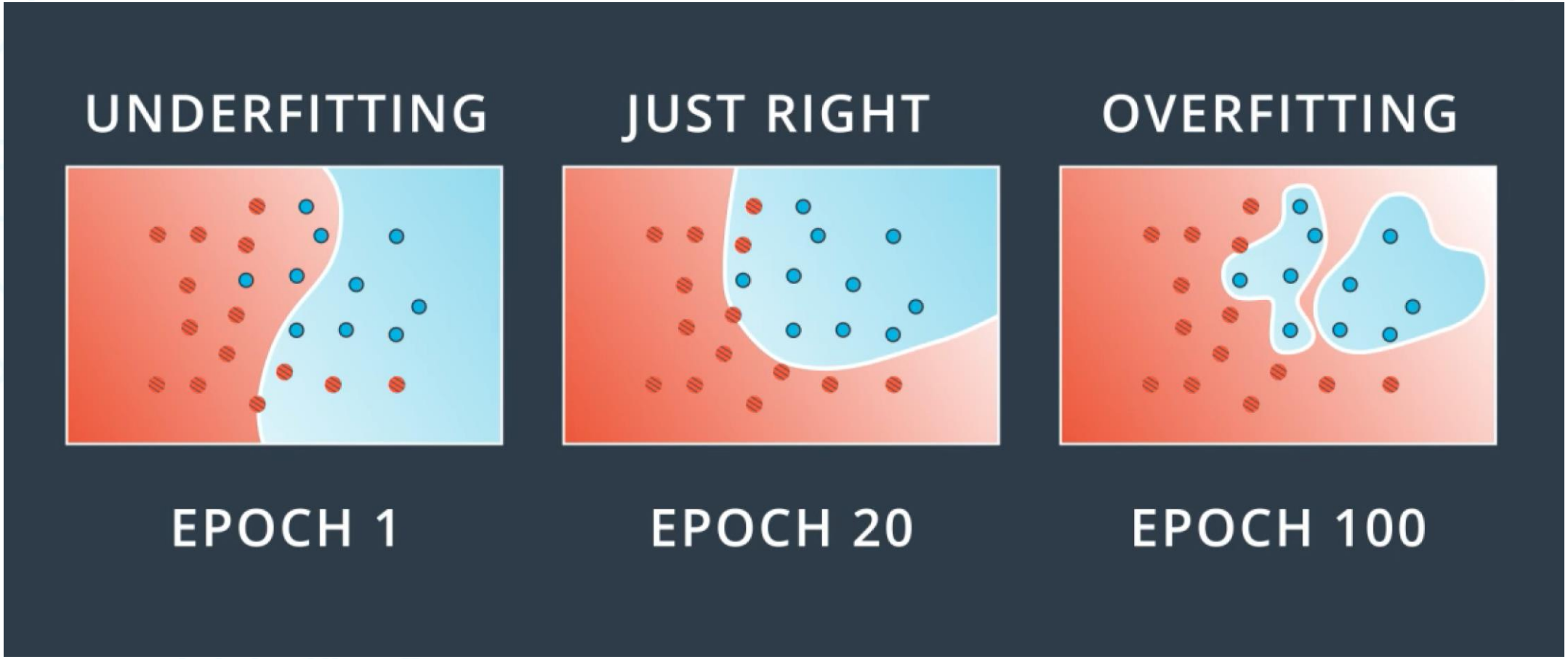
Redes Neurais

Redes Neurais – Model Complexity Graph

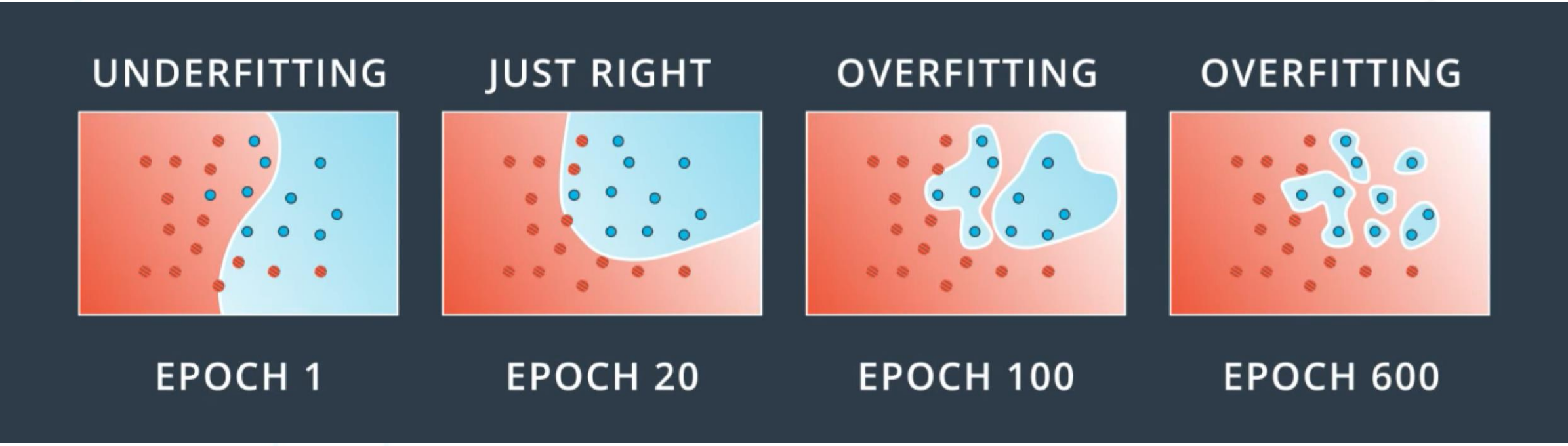
Redes Neurais – Model Complexity Graph



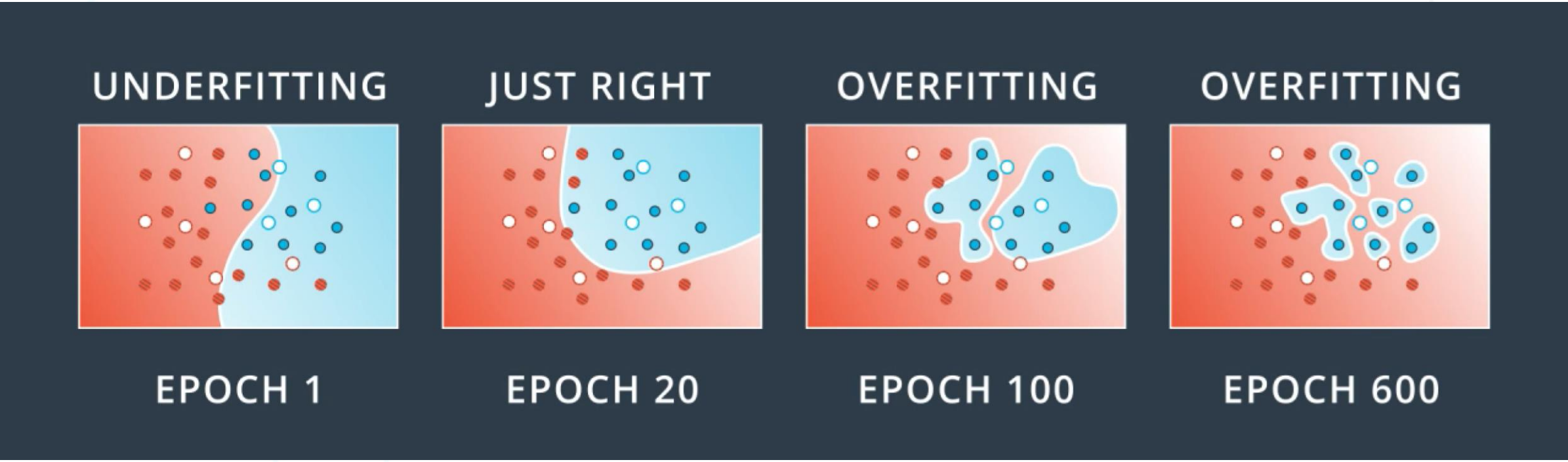
Redes Neurais – Model Complexity Graph



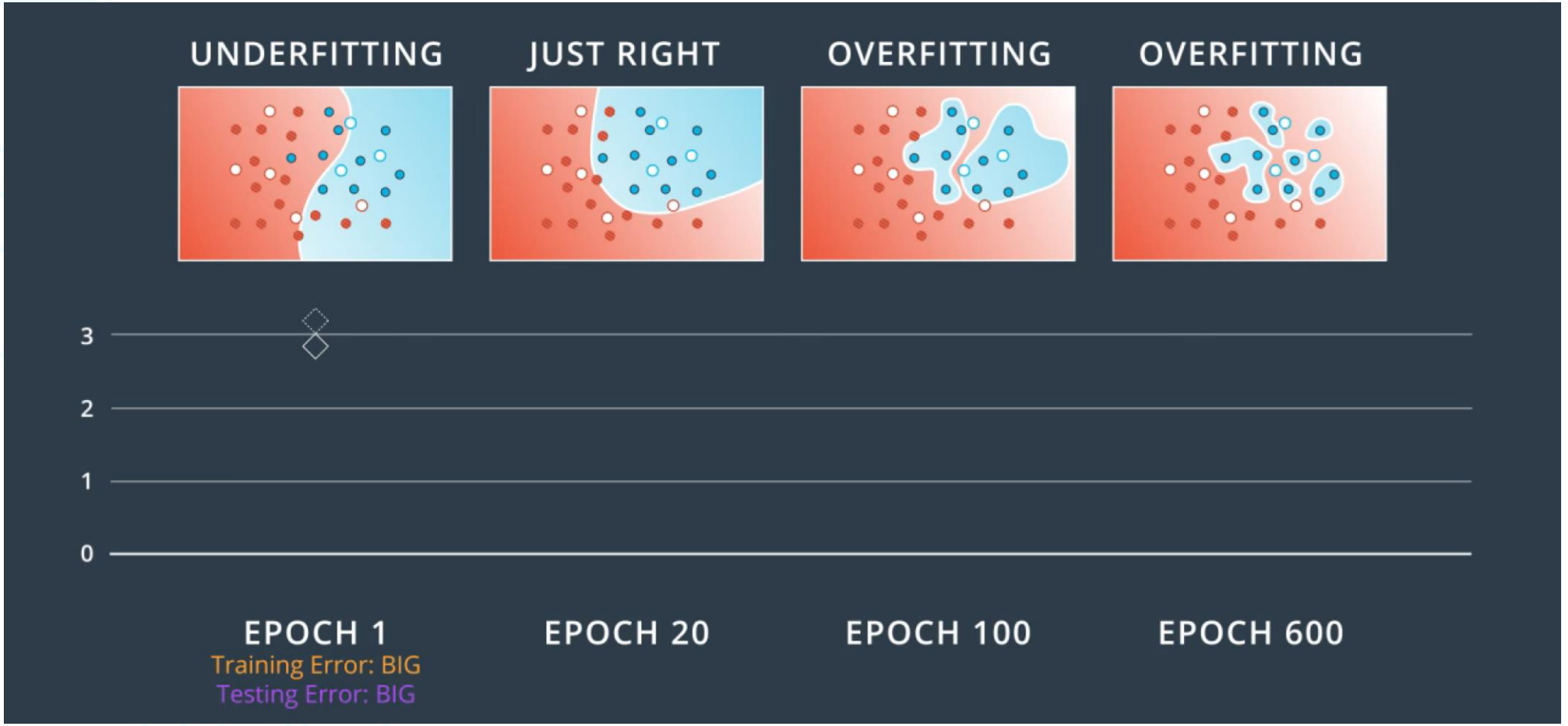
Redes Neurais – Model Complexity Graph



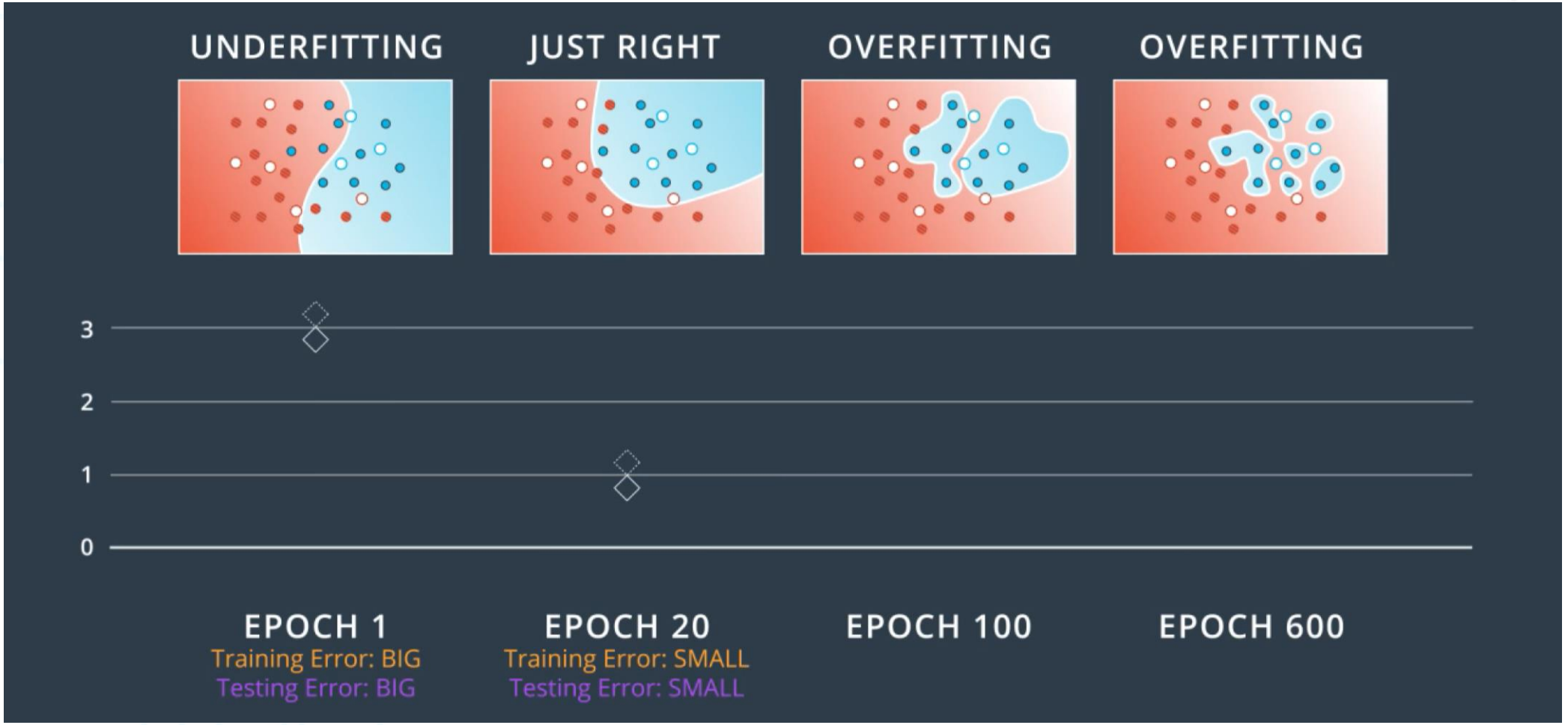
Redes Neurais – Model Complexity Graph



Redes Neurais – Model Complexity Graph

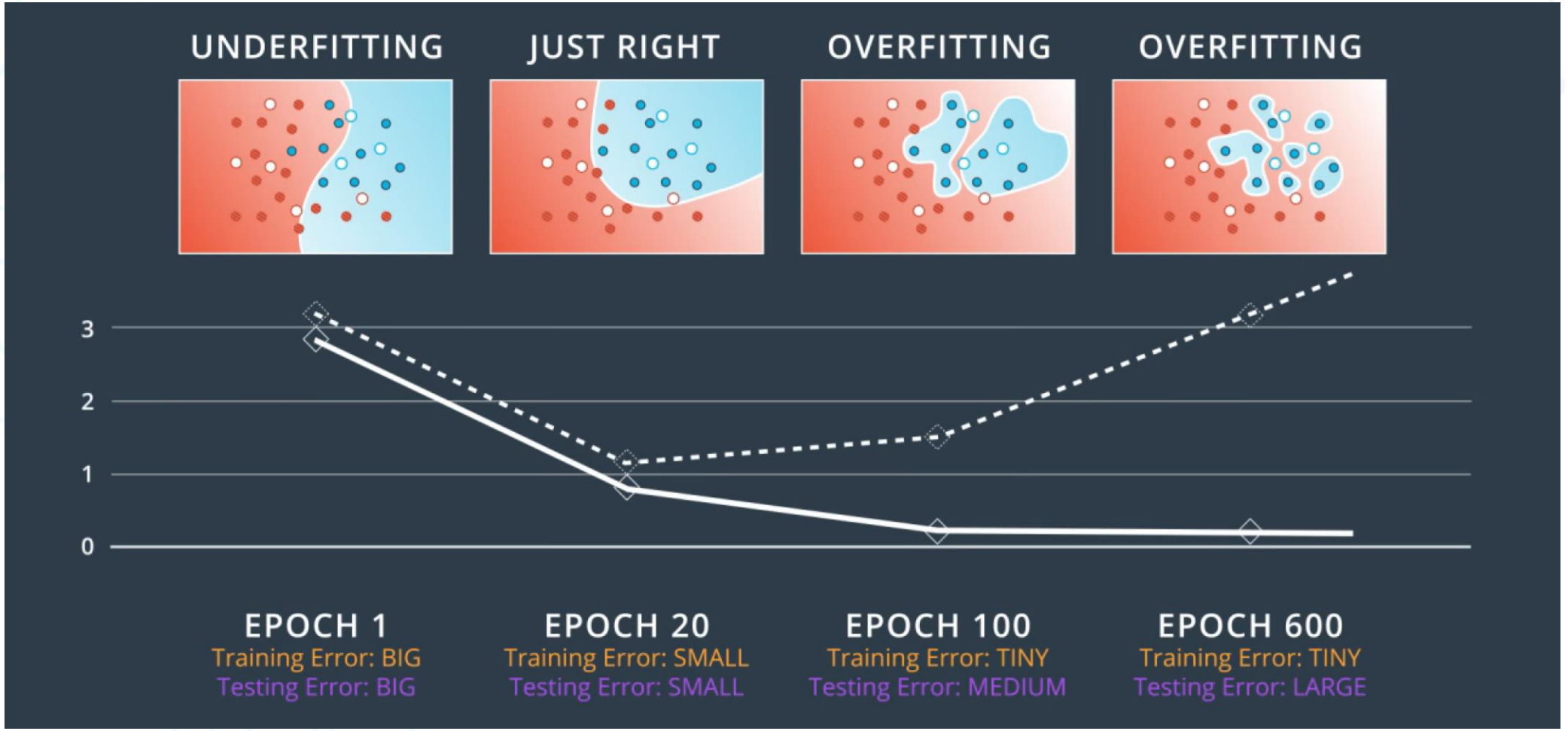


Redes Neurais – Model Complexity Graph

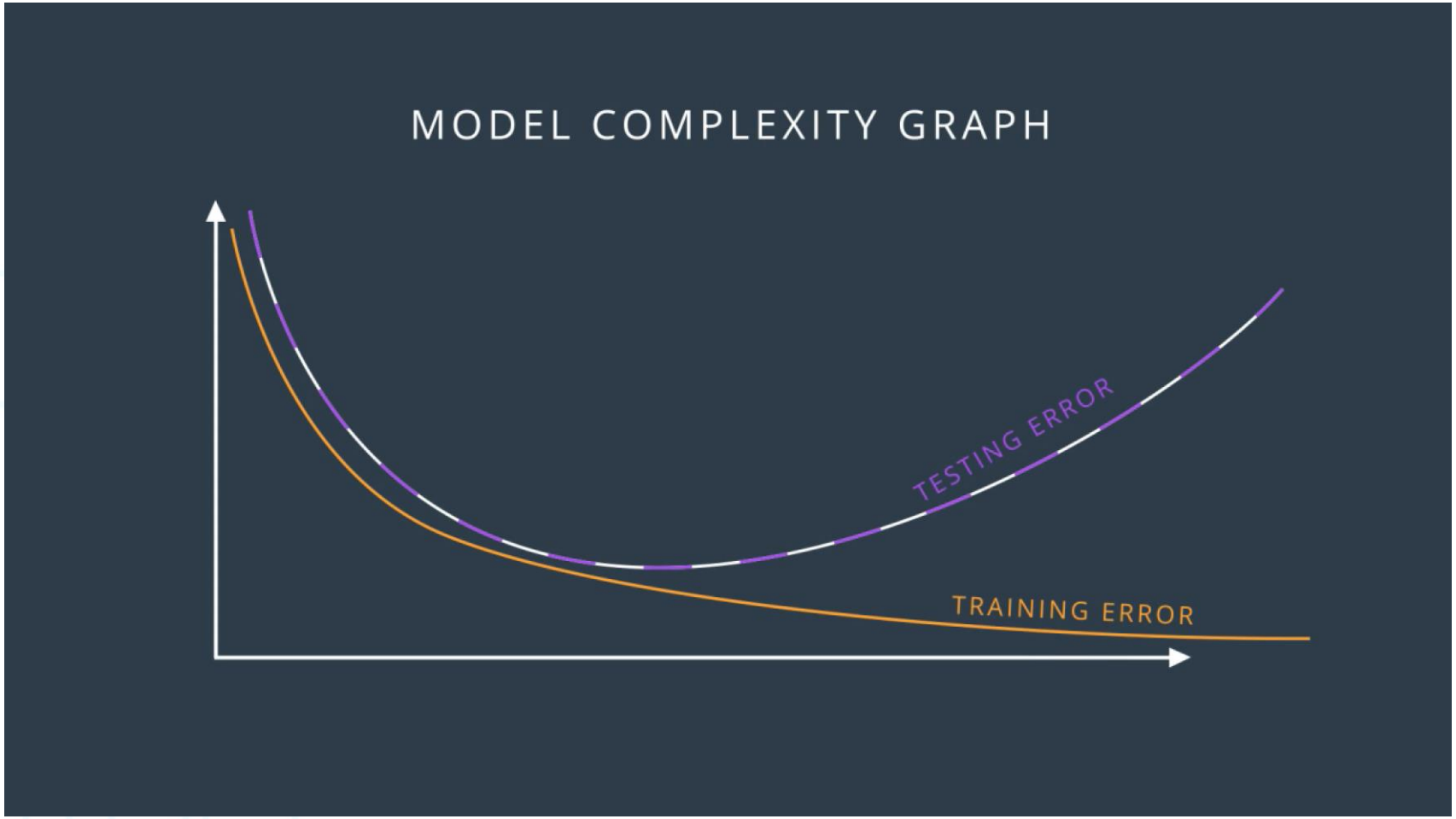


Redes Neurais – Model Complexity Graph

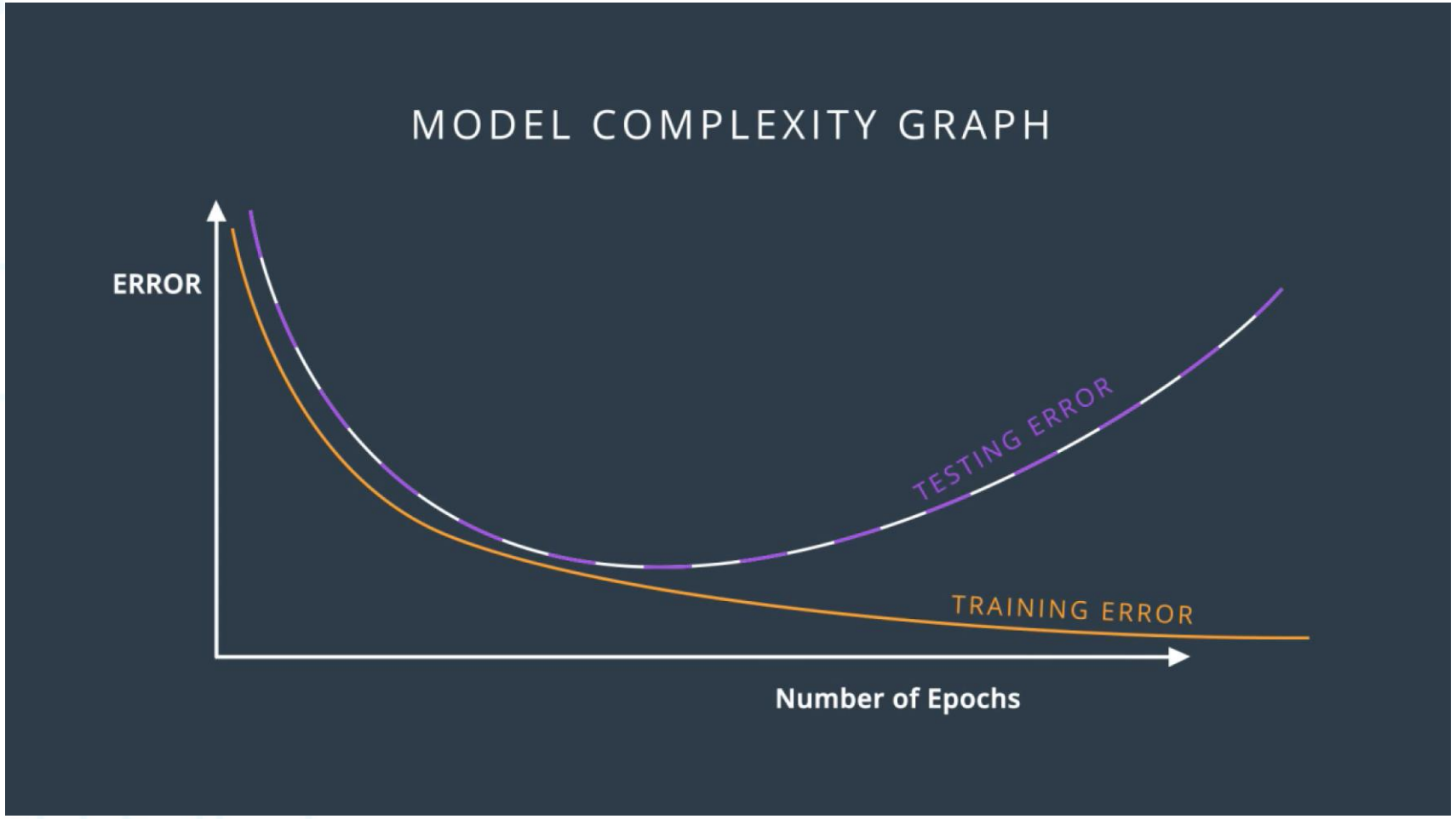
Redes Neurais – Model Complexity Graph



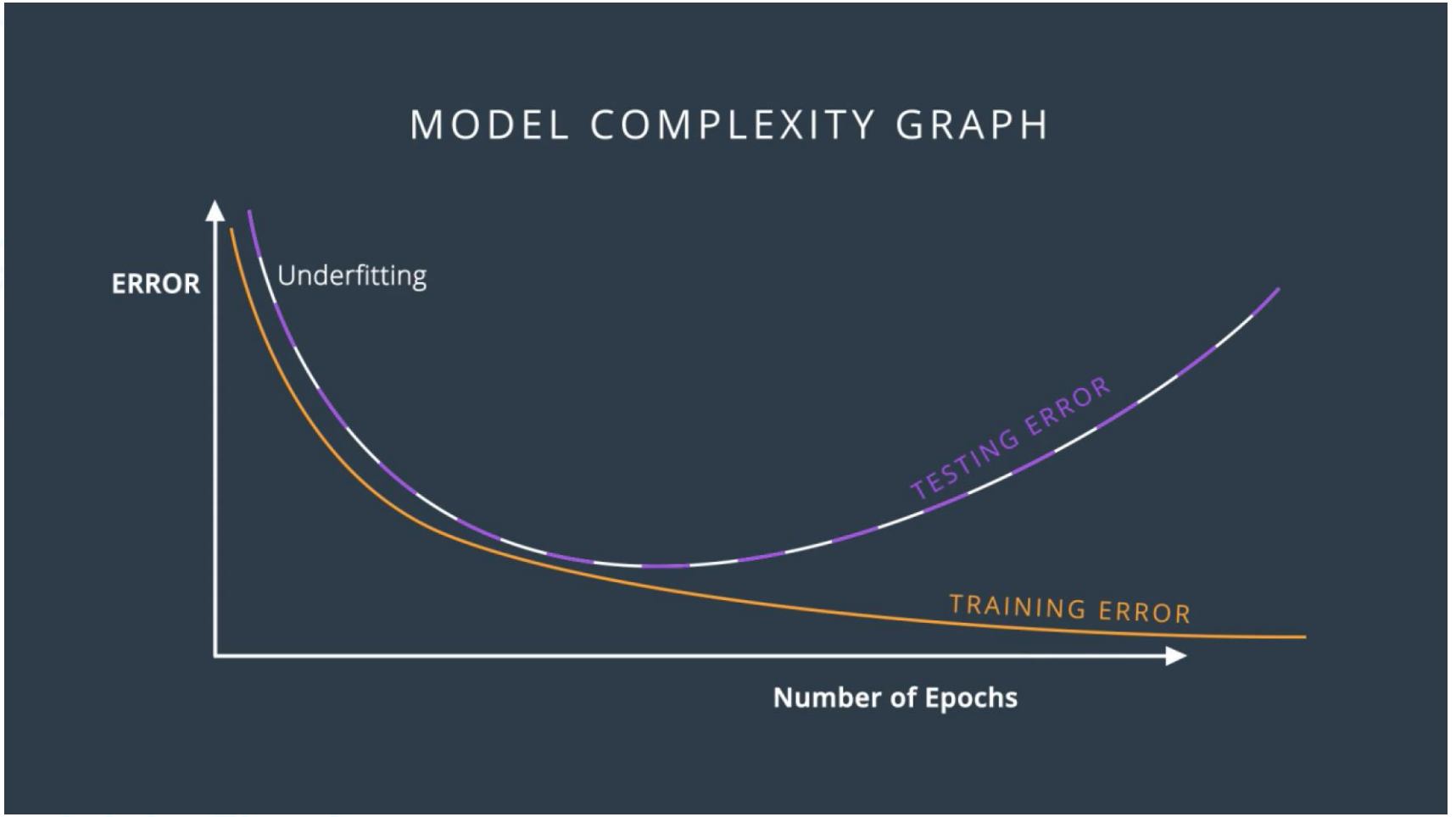
Redes Neurais – Model Complexity Graph



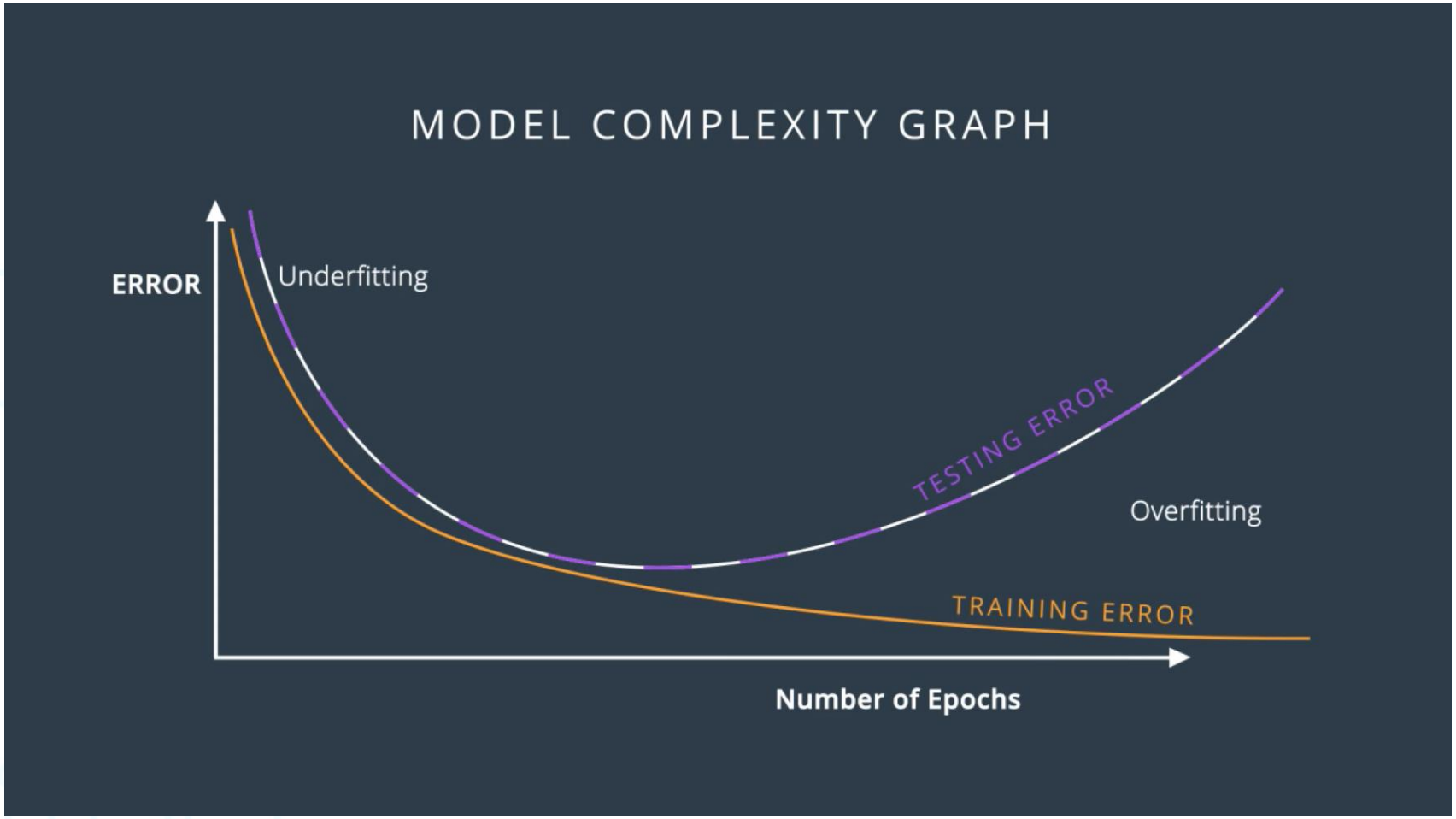
Redes Neurais – Model Complexity Graph



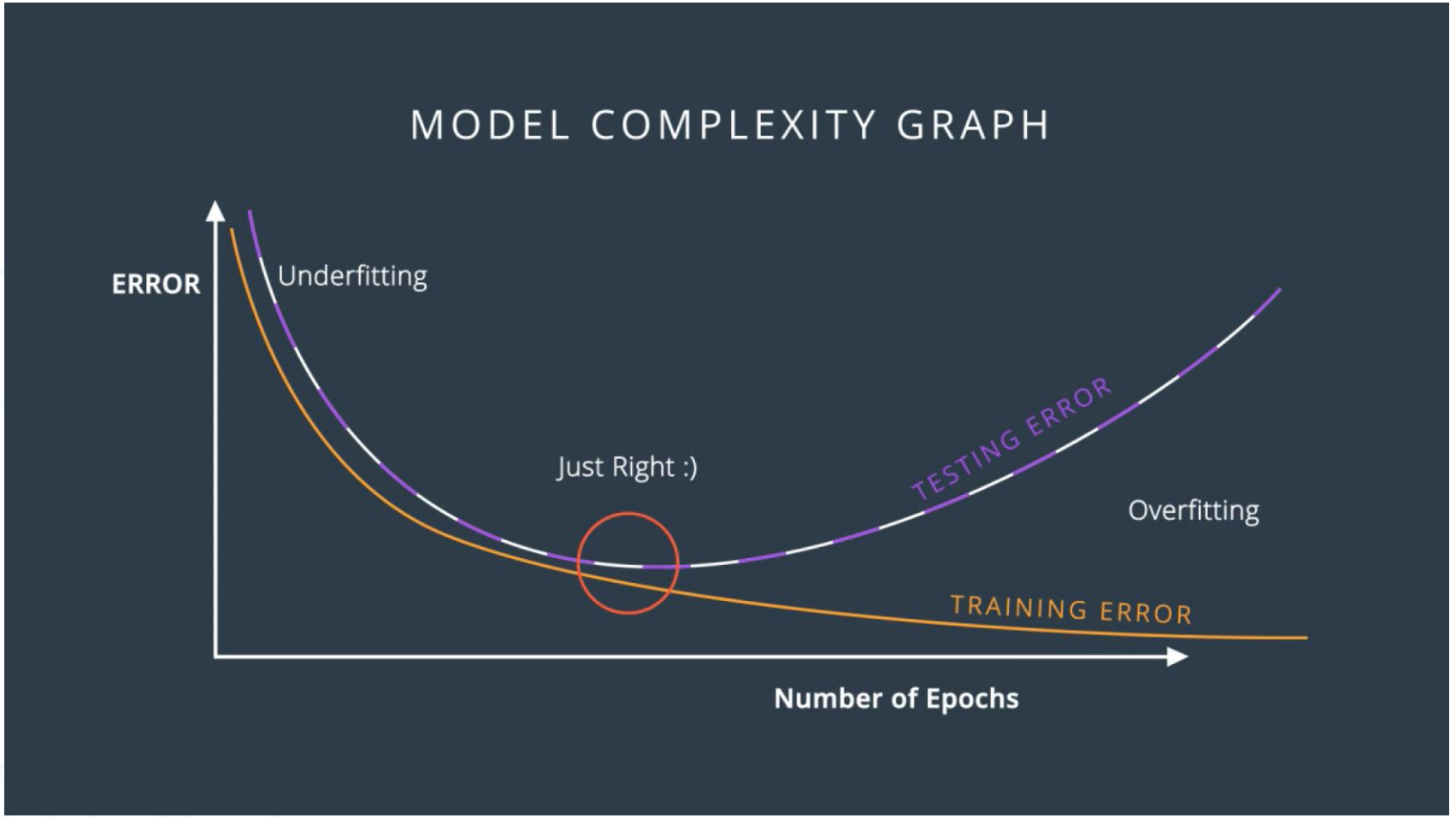
Redes Neurais – Model Complexity Graph



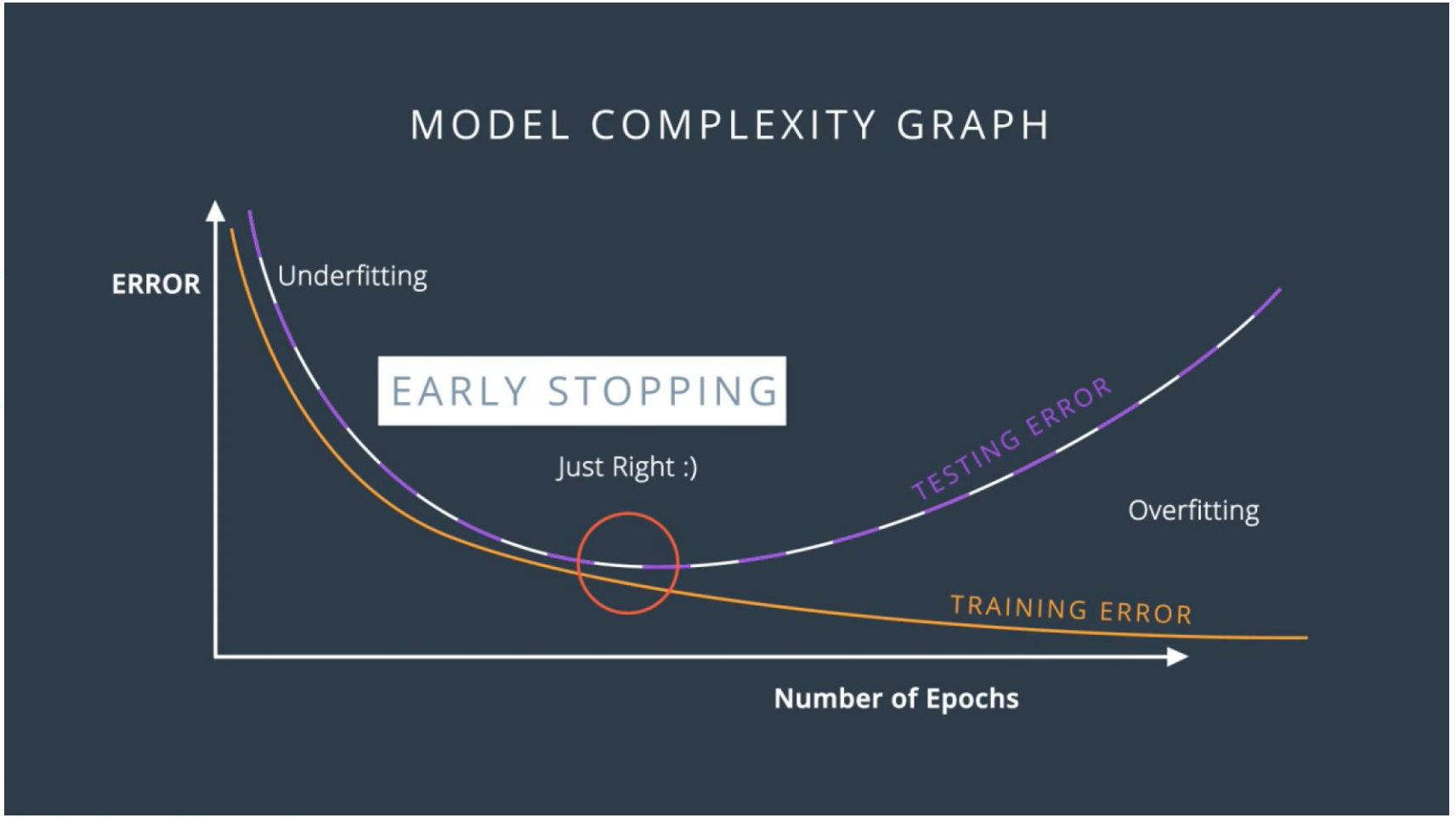
Redes Neurais – Model Complexity Graph



Redes Neurais – Model Complexity Graph



Redes Neurais – Model Complexity Graph



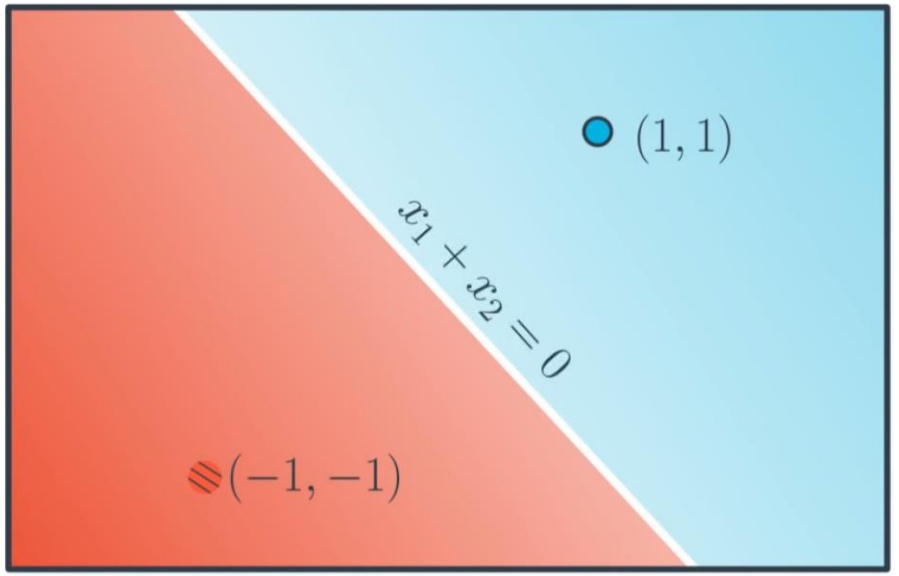
Goal: Split Two Points

Goal: Split Two Points

QUIZ: WHICH GIVES A SMALLER ERROR?

- SOLUTION 1: $x_1 + x_2$
- SOLUTION 2: $10x_1 + 10x_2$

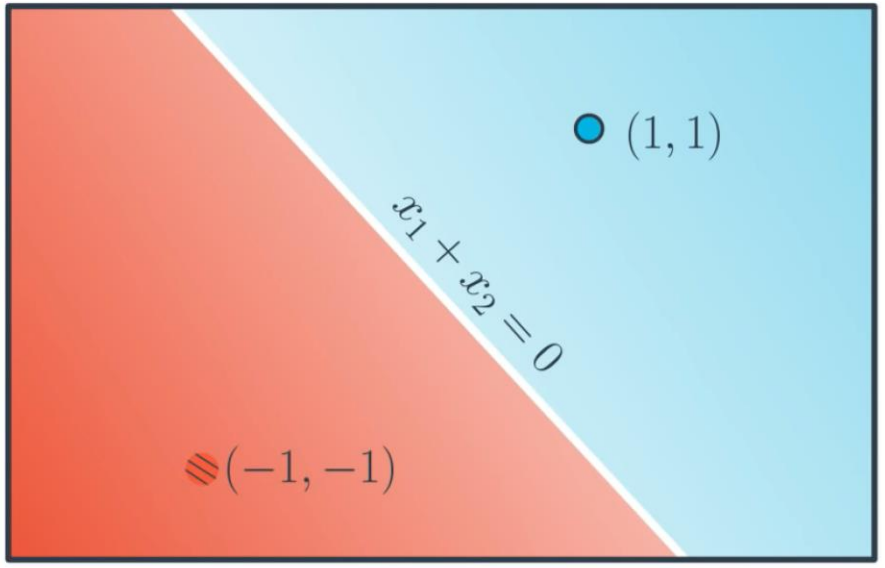
Goal: Split Two Points



QUIZ: WHICH GIVES A SMALLER ERROR?

- SOLUTION 1: $x_1 + x_2$
- SOLUTION 2: $10x_1 + 10x_2$

Goal: Split Two Points

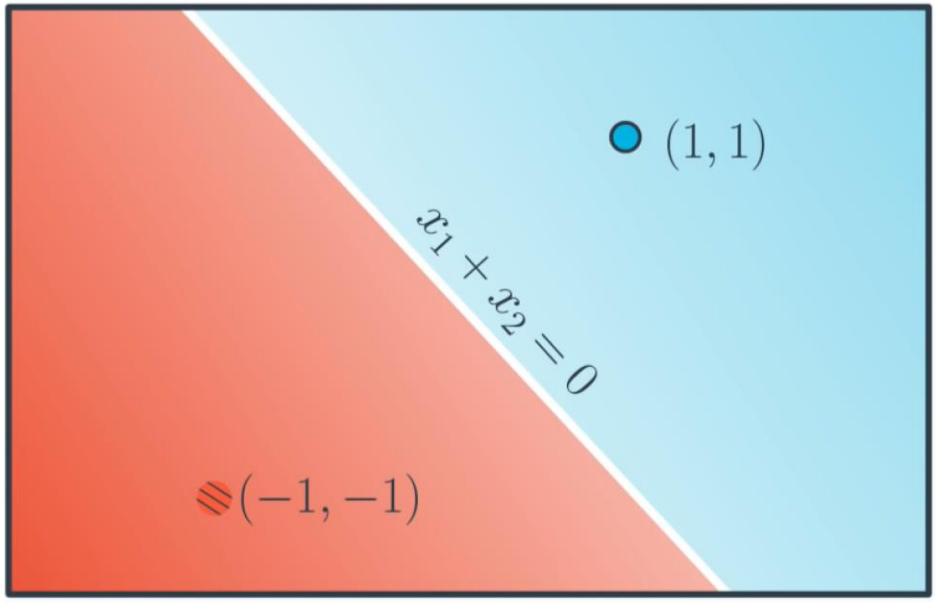


QUIZ: WHICH GIVES A SMALLER ERROR?

Prediction: $\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$

- SOLUTION 1: $x_1 + x_2$
- SOLUTION 2: $10x_1 + 10x_2$

Goal: Split Two Points



QUIZ: WHICH GIVES A SMALLER ERROR?

Prediction: $\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$

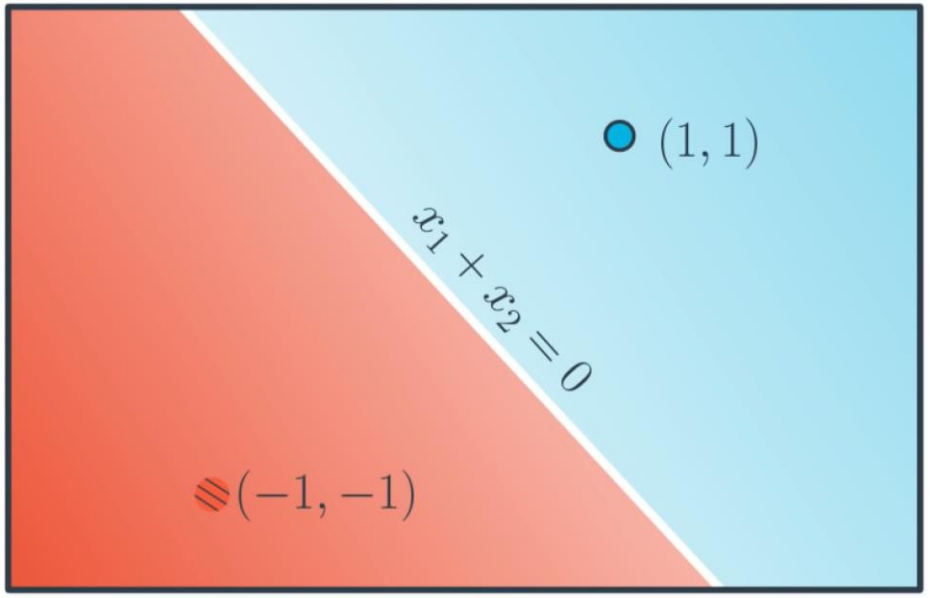
- SOLUTION 1: $x_1 + x_2$

  Predictions:

  $\sigma(1 + 1) = 0.88$

- SOLUTION 2: $10x_1 + 10x_2$

Goal: Split Two Points



QUIZ: WHICH GIVES A SMALLER ERROR?

Prediction: $\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$

- SOLUTION 1: $x_1 + x_2$

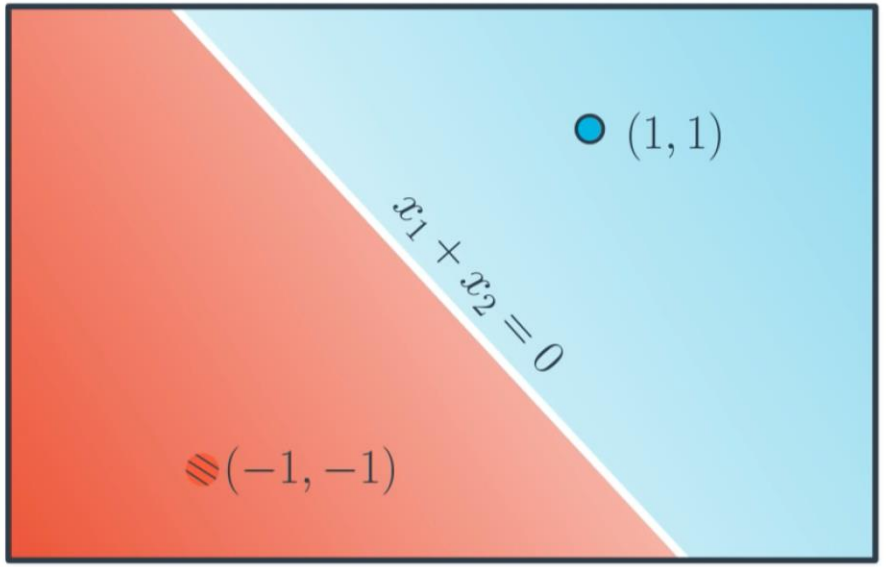
  Predictions:

  $\sigma(1 + 1) = 0.88$

  $\sigma(-1 - 1) = 0.12$

- SOLUTION 2: $10x_1 + 10x_2$

Goal: Split Two Points



QUIZ: WHICH GIVES A SMALLER ERROR?

Prediction: $\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$

- SOLUTION 1: $x_1 + x_2$

  Predictions:

  $\sigma(1 + 1) = 0.88$

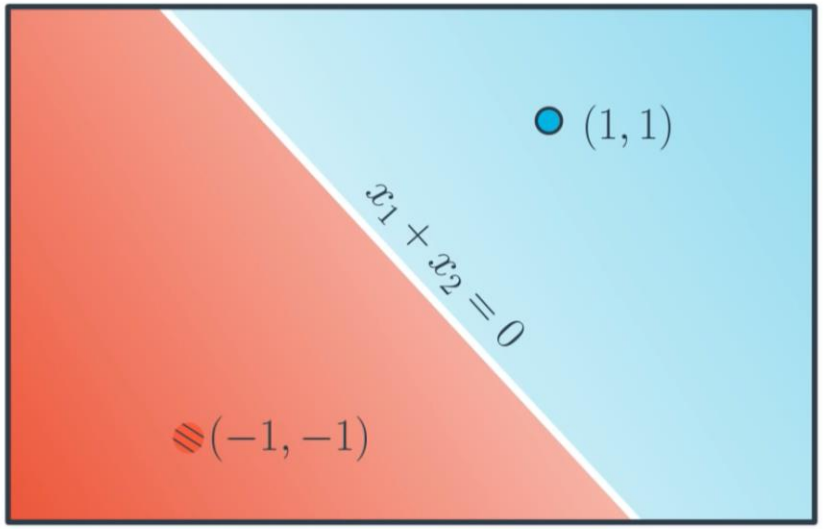
  $\sigma(-1 - 1) = 0.12$

- SOLUTION 2: $10x_1 + 10x_2$

  Predictions:

  $\sigma(10 + 10) = 0.9999999979$

Goal: Split Two Points



QUIZ: WHICH GIVES A SMALLER ERROR?

Prediction: $\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$

- SOLUTION 1: $x_1 + x_2$

  Predictions:

  $\sigma(1 + 1) = 0.88$

  $\sigma(-1 - 1) = 0.12$

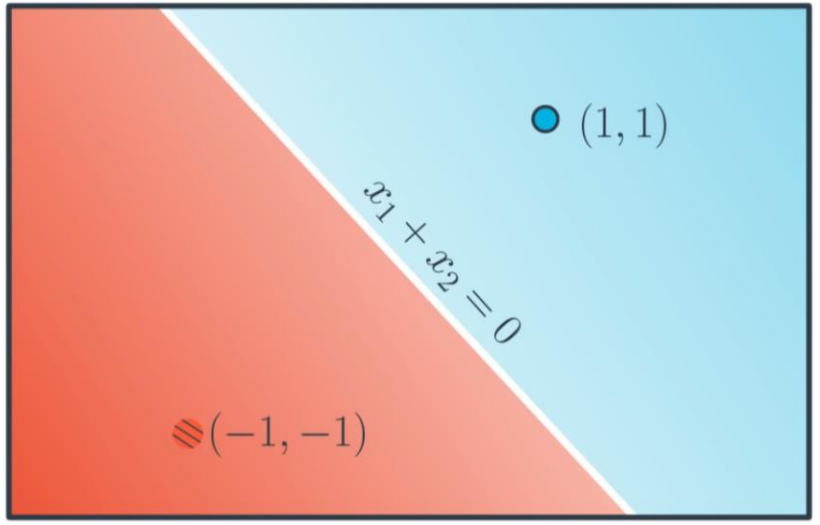
- SOLUTION 2: $10x_1 + 10x_2$

  Predictions:

  $\sigma(10 + 10) = 0.9999999979$

  $\sigma(-10 - 10) = 0.0000000021$

Goal: Split Two Points



QUIZ: WHICH GIVES A SMALLER ERROR?

Prediction: $\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$

○ SOLUTION 1: $x_1 + x_2$

Predictions:

$\sigma(1 + 1) = 0.88$

$\sigma(-1 - 1) = 0.12$

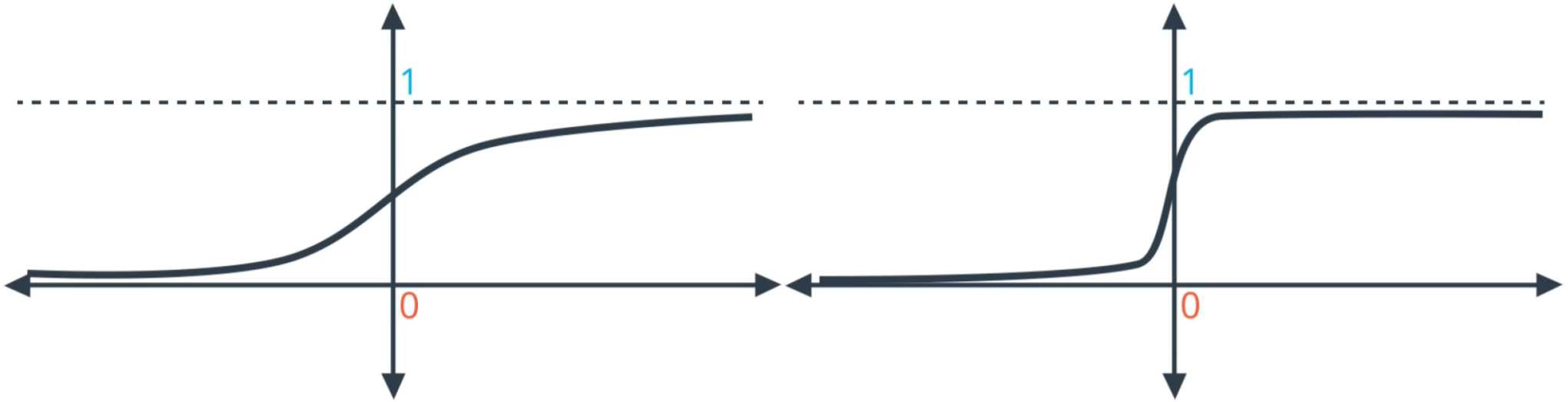
✓ SOLUTION 2: $10x_1 + 10x_2$

Predictions:

$\sigma(10 + 10) = 0.9999999979$

$\sigma(-10 - 10) = 0.0000000021$

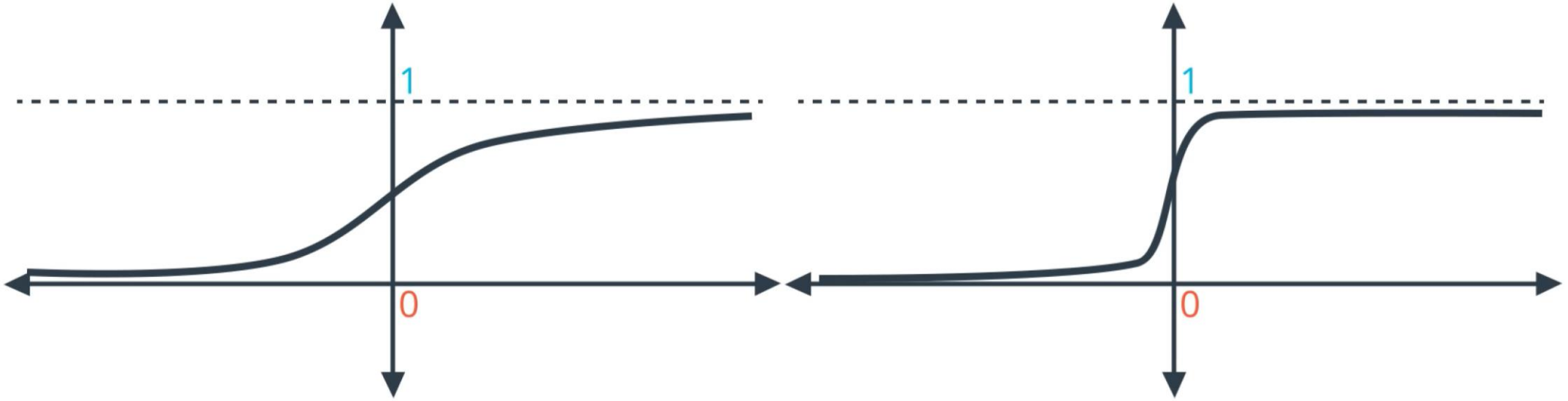
Redes Neurais – Regularização



Prediction:

$$\hat{y} = \sigma(x_1 + x_2 + b)$$

Redes Neurais – Regularização

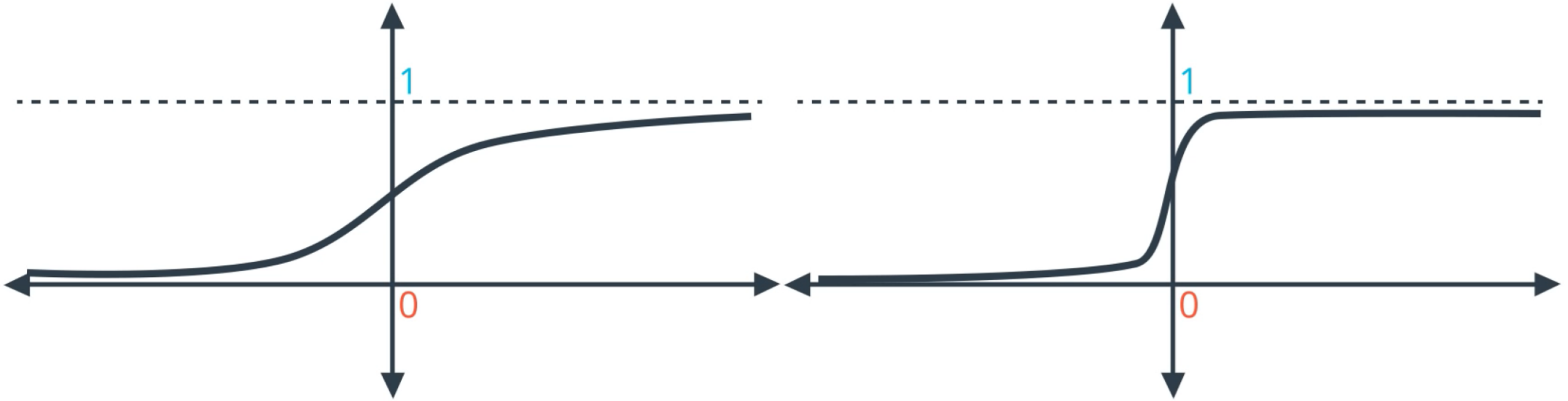


$\sigma(x_1 + x_2)$

Prediction:

$\hat{y} = \sigma(x_1 + x_2 + b)$

Redes Neurais – Regularização



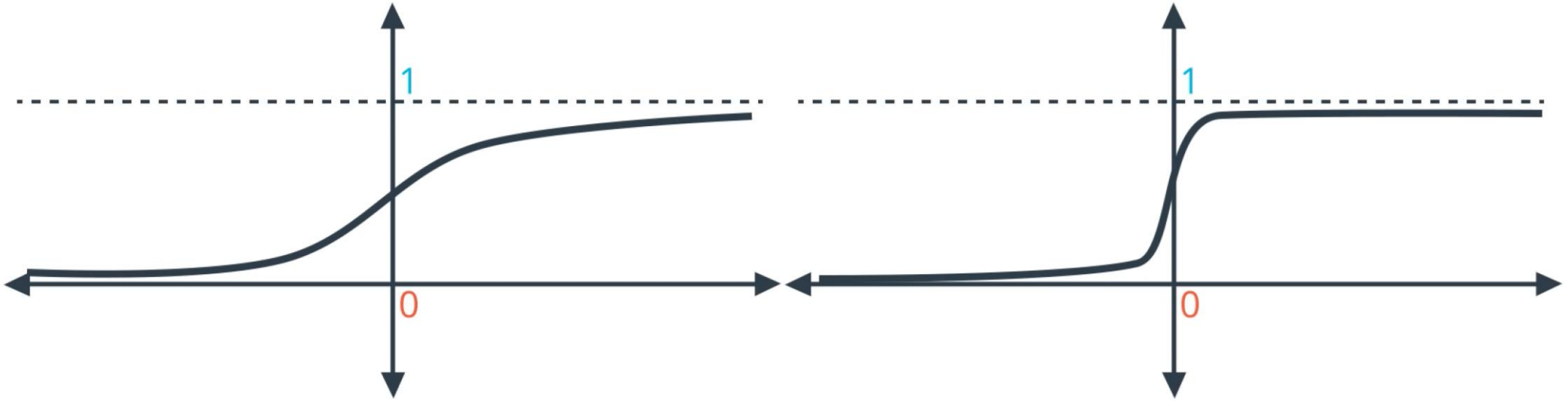
$\sigma(x_1 + x_2)$

Prediction:

$\hat{y} = \sigma(x_1 + x_2 + b)$

$\sigma(10x_1 + 10x_2)$

Redes Neurais – Regularização



$\sigma(x_1 + x_2)$

Prediction:

$\hat{y} = \sigma(x_1 + x_2 + b)$

$\sigma(10x_1 + 10x_2)$

TOO CERTAIN

Redes Neurais – Regularização

"The whole problem with Artificial Intelligence is that bad models are so certain of themselves, and good models so full of doubts."

BertrAIND Russell

Redes Neurais – Regularização

Solution: Regularization

LARGE COEFFICIENTS ⟶ OVERFITTING

PENALIZE LARGE WEIGHTS

$$(w_1, \ldots, w_n)$$

Redes Neurais – Regularização

Solution: Regularization

LARGE COEFFICIENTS ⟶ OVERFITTING

PENALIZE LARGE WEIGHTS

$$(w_1, \ldots, w_n)$$

$$\text{ERROR FUNCTION} = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)ln(1-\hat{y}_i) + y_i ln(\hat{y}_i) +$$

Redes Neurais – Regularização

Solution: Regularization

LARGE COEFFICIENTS ⟶ OVERFITTING

PENALIZE LARGE WEIGHTS

$$(w_1, \ldots, w_n)$$

$$\text{ERROR FUNCTION} = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)ln(1-\hat{y}_i) + y_i ln(\hat{y}_i) + \boxed{\lambda(|w_1| + \ldots + |w_n|)}$$

$$\text{ERROR FUNCTION} = -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)ln(1-\hat{y}_i) + y_i ln(\hat{y}_i) + \boxed{\lambda({w_1}^2 + \ldots + {w_n}^2)}$$

Redes Neurais – Regularização

Solution: Regularization

LARGE COEFFICIENTS ⟶ OVERFITTING

PENALIZE LARGE WEIGHTS

$$(w_1, \ldots, w_n)$$

L1 ERROR FUNCTION $= -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)ln(1-\hat{y}_i) + y_i ln(\hat{y}_i) + \boxed{\lambda(|w_1| + \ldots + |w_n|)}$

L2 ERROR FUNCTION $= -\frac{1}{m}\sum_{i=1}^{m}(1-y_i)ln(1-\hat{y}_i) + y_i ln(\hat{y}_i) + \boxed{\lambda({w_1}^2 + \ldots + {w_n}^2)}$

L1 vs L2 Regularization

L1

SPARSITY: $(1, 0, 0, 1, 0)$

GOOD FOR FEATURE SELECTION

L2

SPARSITY: $(0.5, 0.3, -0.2, 0.4, 0.1)$

NORMALLY BETTER FOR TRAINING MODELS

Redes Neurais – Regularização

L1 vs L2 Regularization

L1

SPARSITY: $(1, 0, 0, 1, 0)$

GOOD FOR FEATURE SELECTION

L2

SPARSITY: $(0.5, 0.3, -0.2, 0.4, 0.1)$

NORMALLY BETTER FOR TRAINING MODELS

$$(1, 0) \rightarrow (0.5, 0.5)$$

Redes Neurais – Regularização

L1 vs L2 Regularization

L1

SPARSITY: $(1, 0, 0, 1, 0)$

GOOD FOR FEATURE SELECTION

L2

SPARSITY: $(0.5, 0.3, -0.2, 0.4, 0.1)$

NORMALLY BETTER FOR TRAINING MODELS

$$(1, 0) \rightarrow (0.5, 0.5)$$

$$1^2 + 0^2 = 1$$

VINI CARIDÁ
Human-Centered Data & AI

L1 vs L2 Regularization

L1

SPARSITY: $(1, 0, 0, 1, 0)$

GOOD FOR FEATURE SELECTION

L2

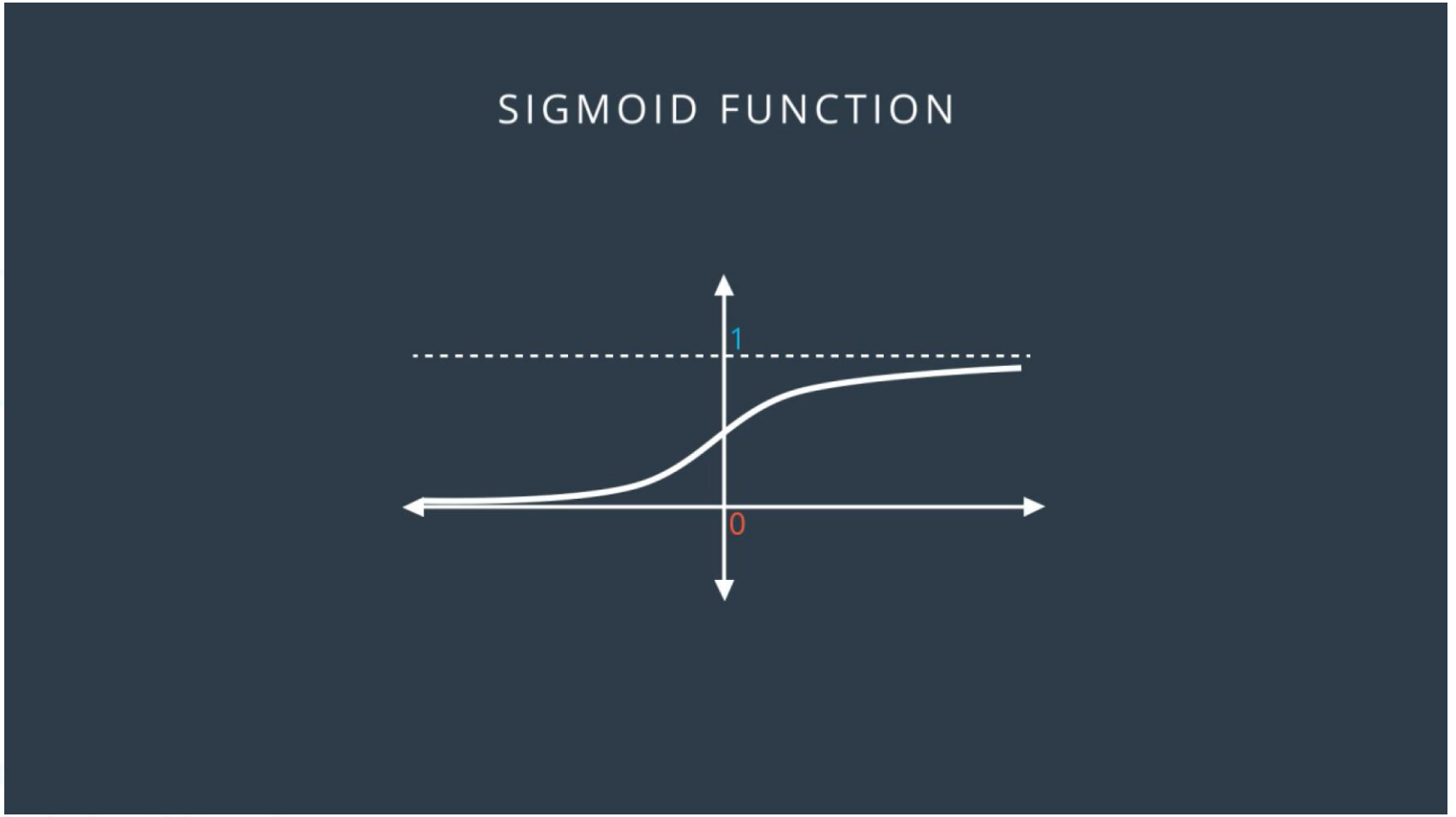
SPARSITY: $(0.5, 0.3, -0.2, 0.4, 0.1)$

NORMALLY BETTER FOR TRAINING MODELS

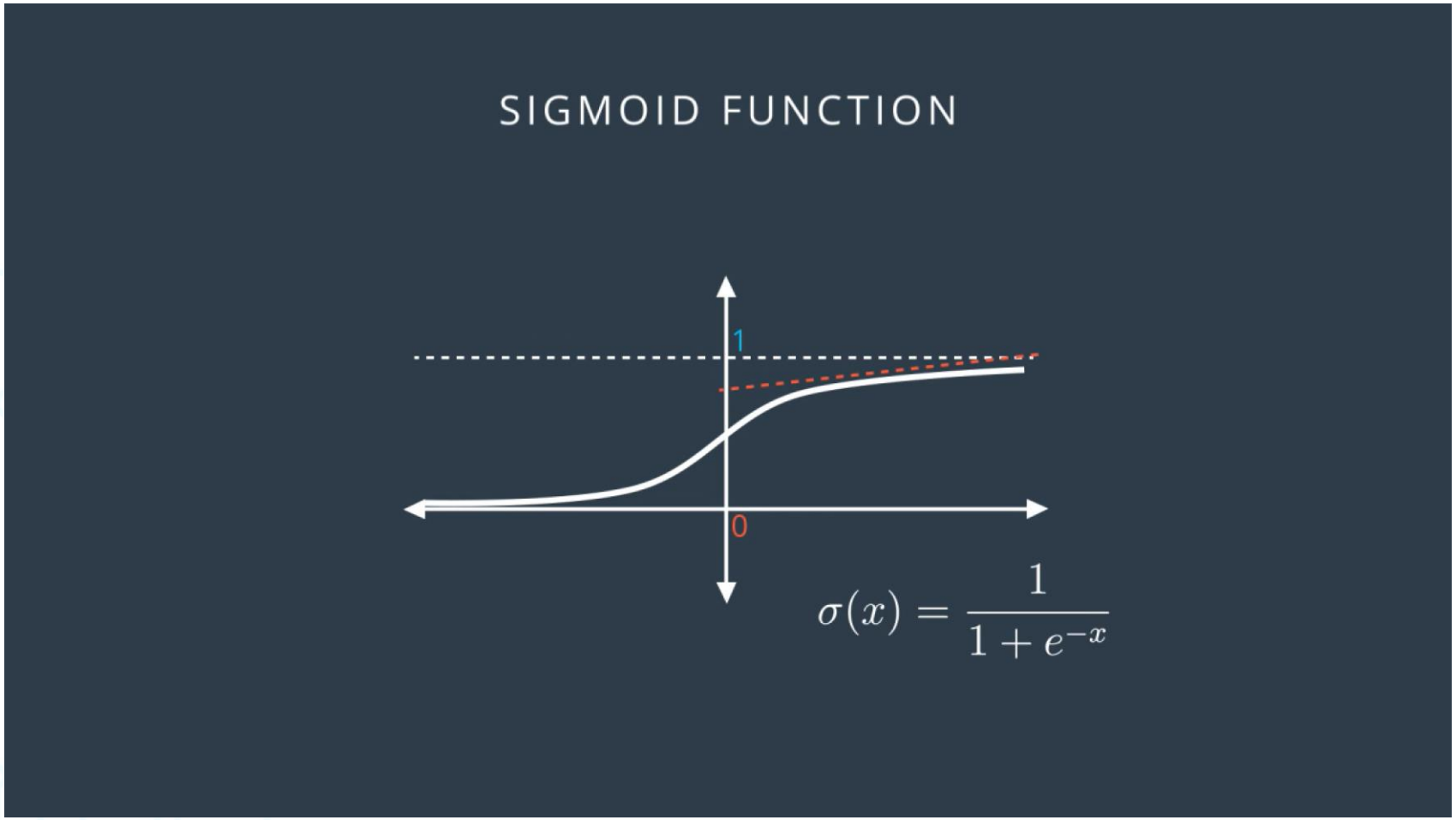
$$(1, 0) \rightarrow (0.5, 0.5)$$

$$1^2 + 0^2 = 1 \quad 0.5^2 + 0.5^2 = 0.5$$

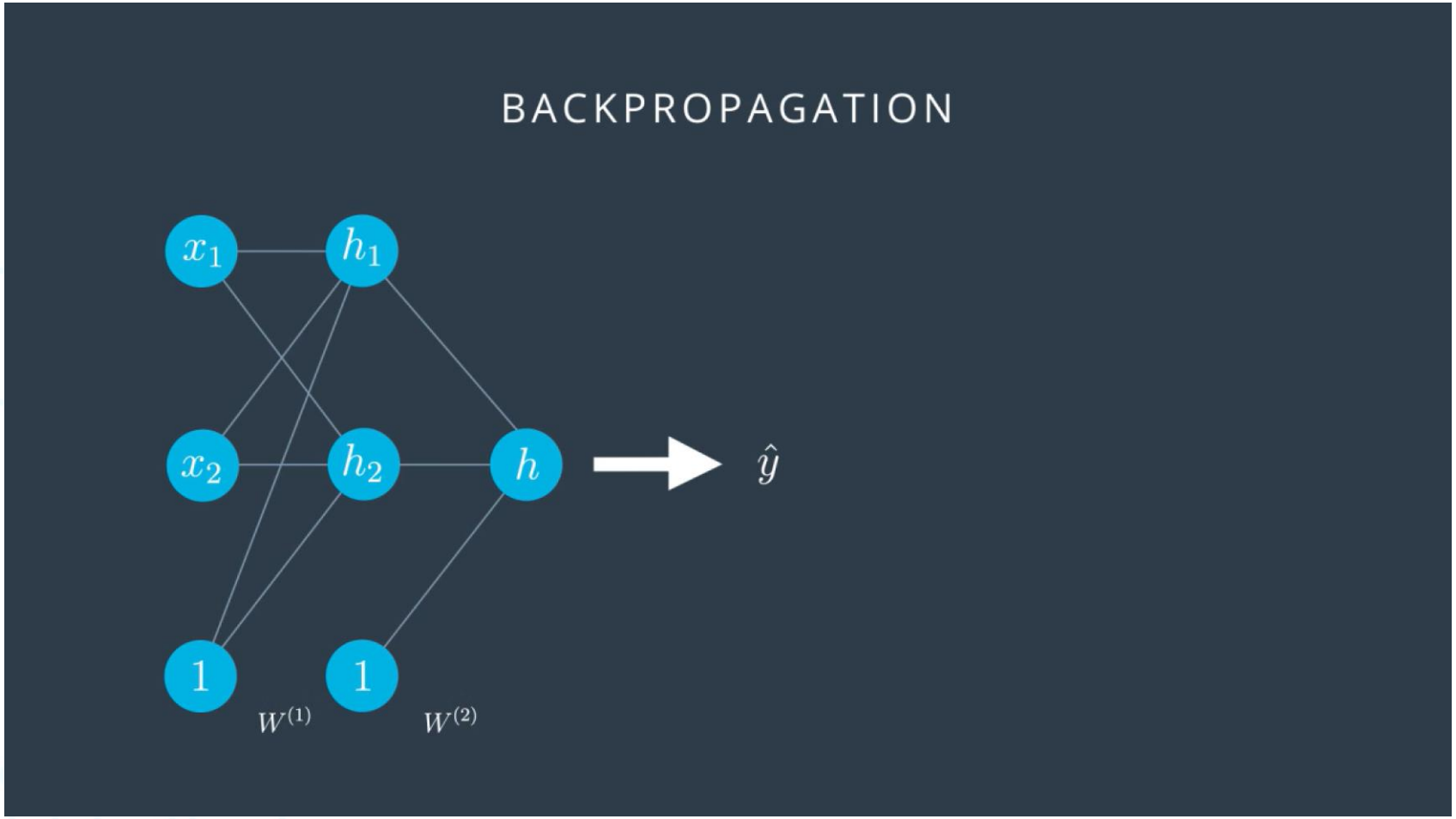
Vanishing Gradient



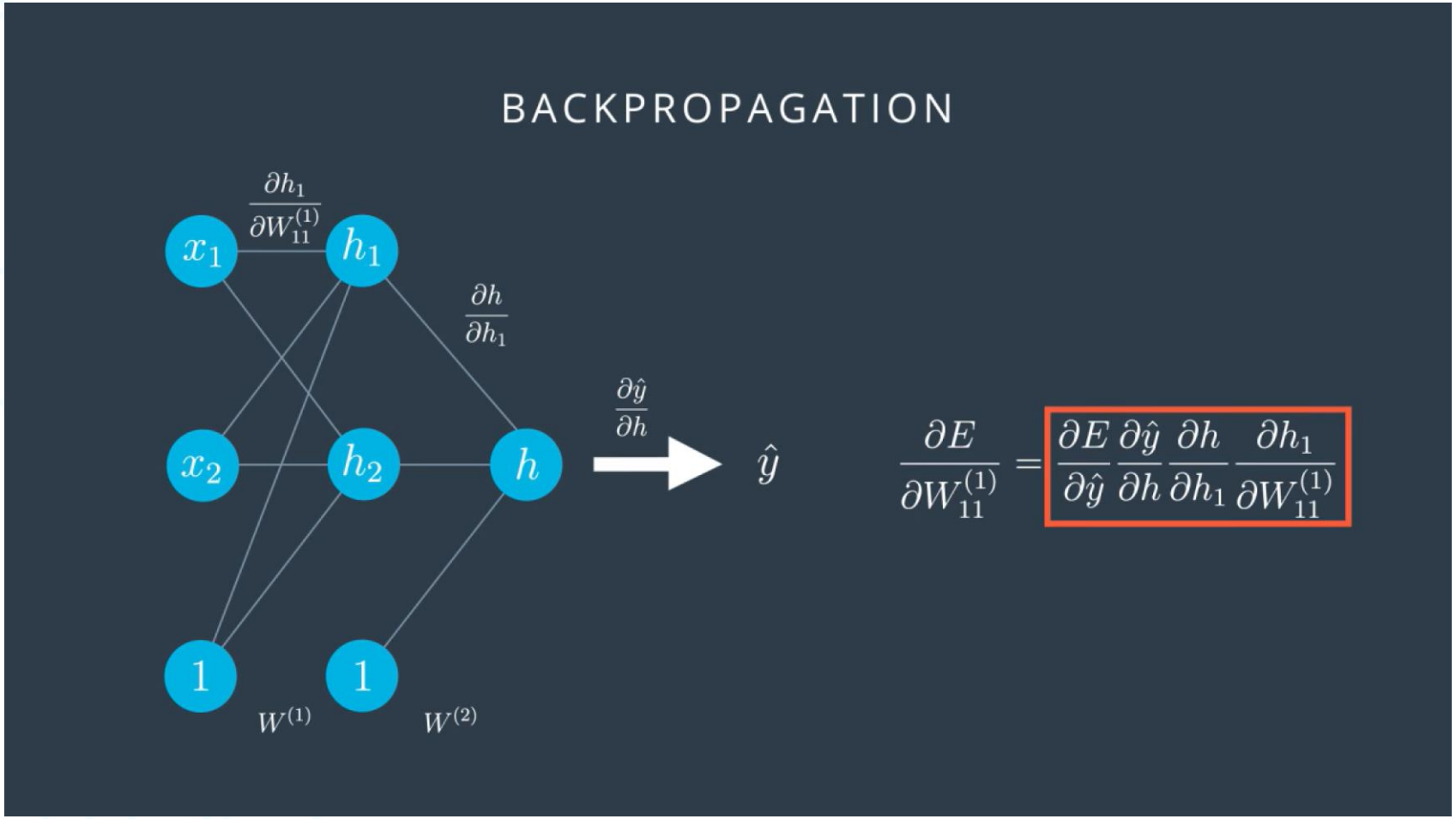
Vanishing Gradient



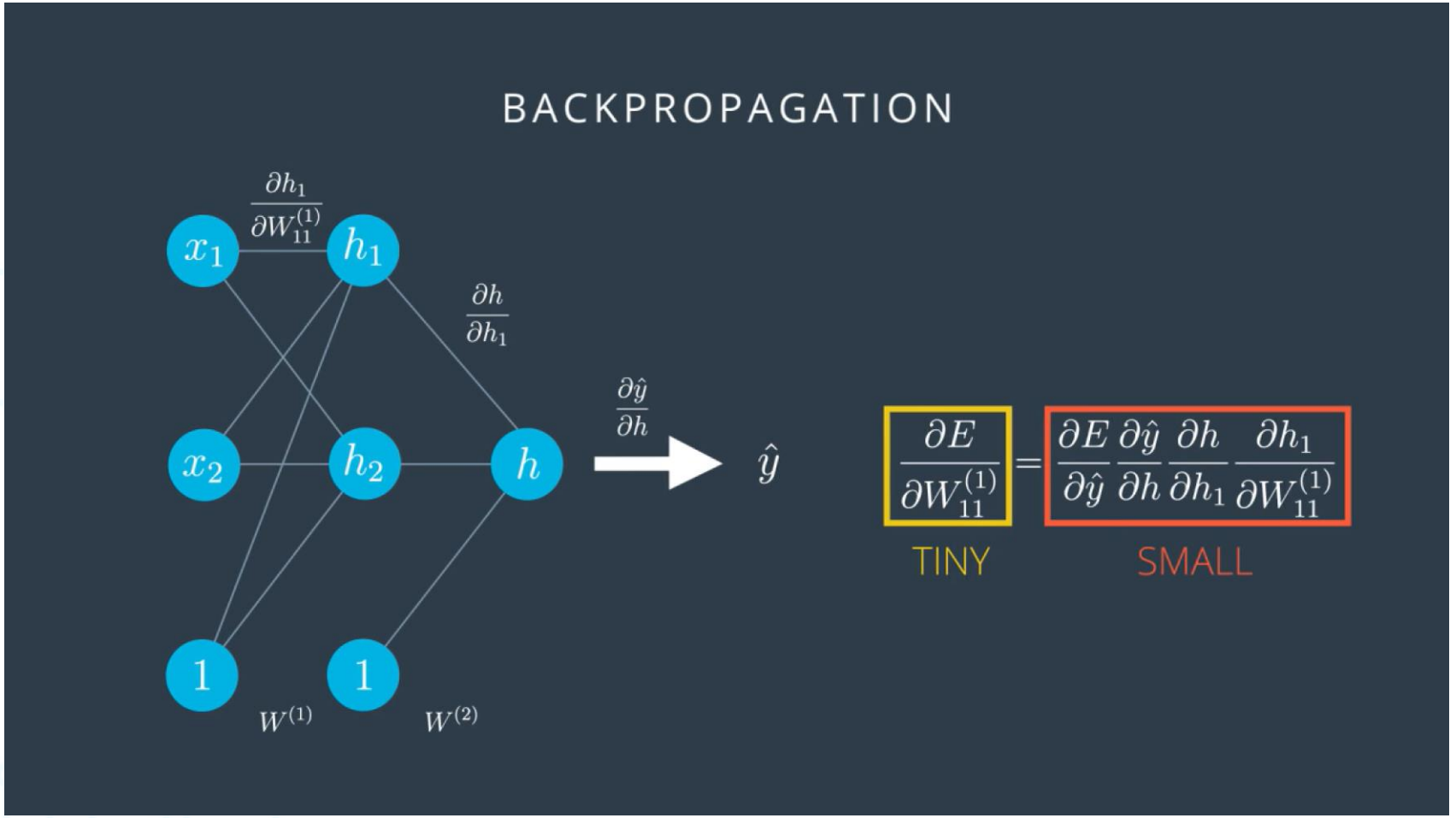
Vanishing Gradient



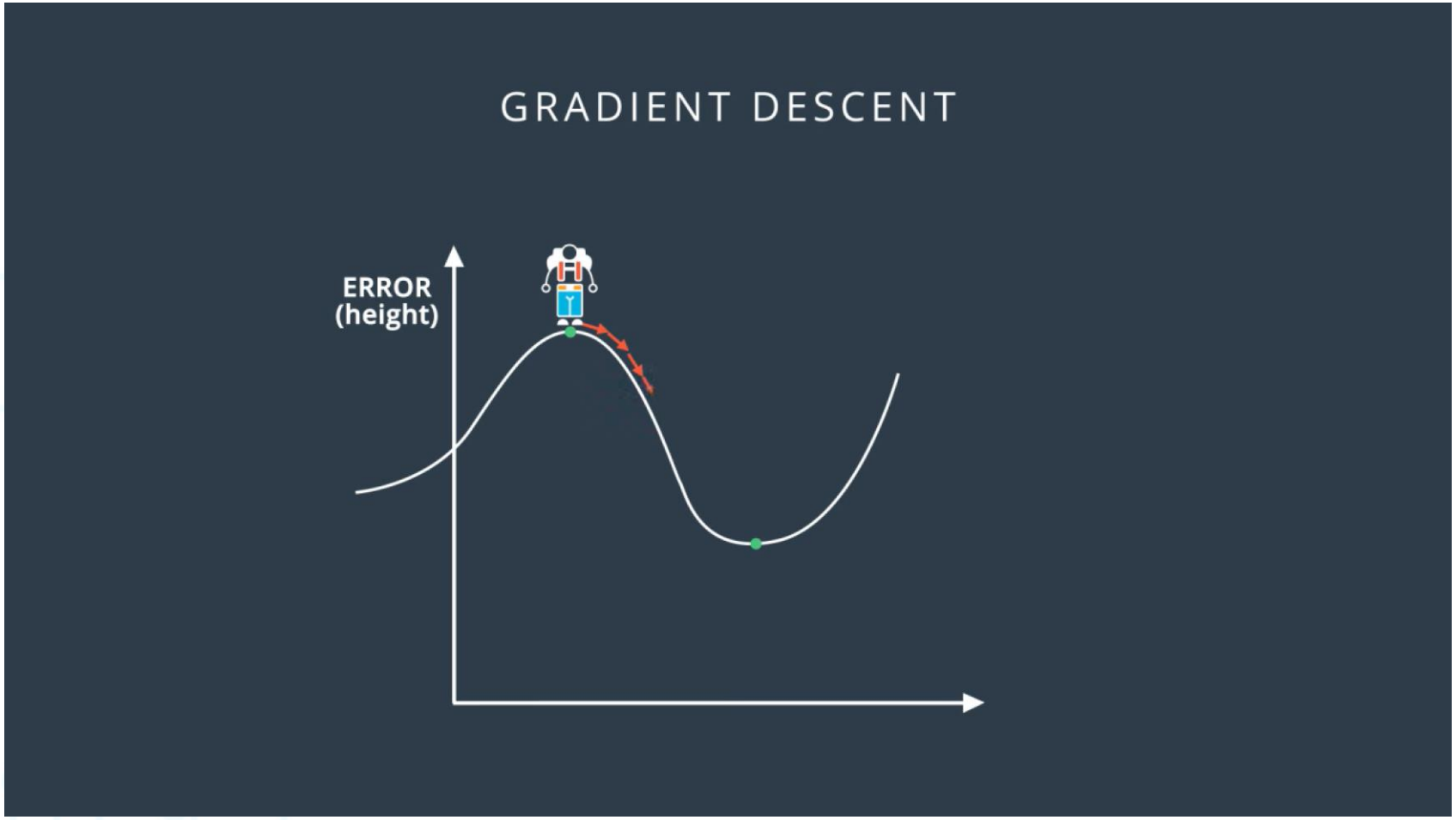
Vanishing Gradient



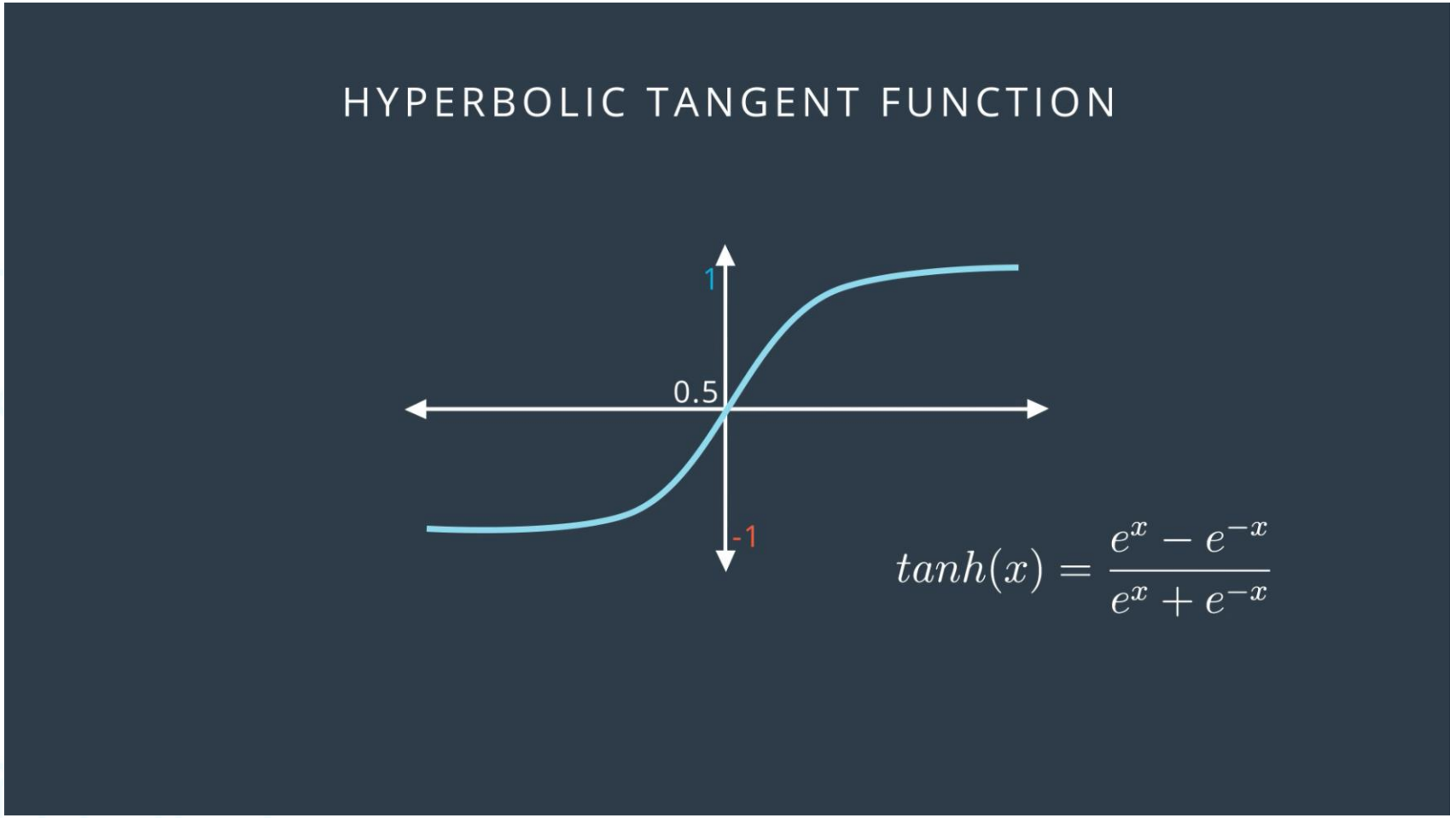
Vanishing Gradient



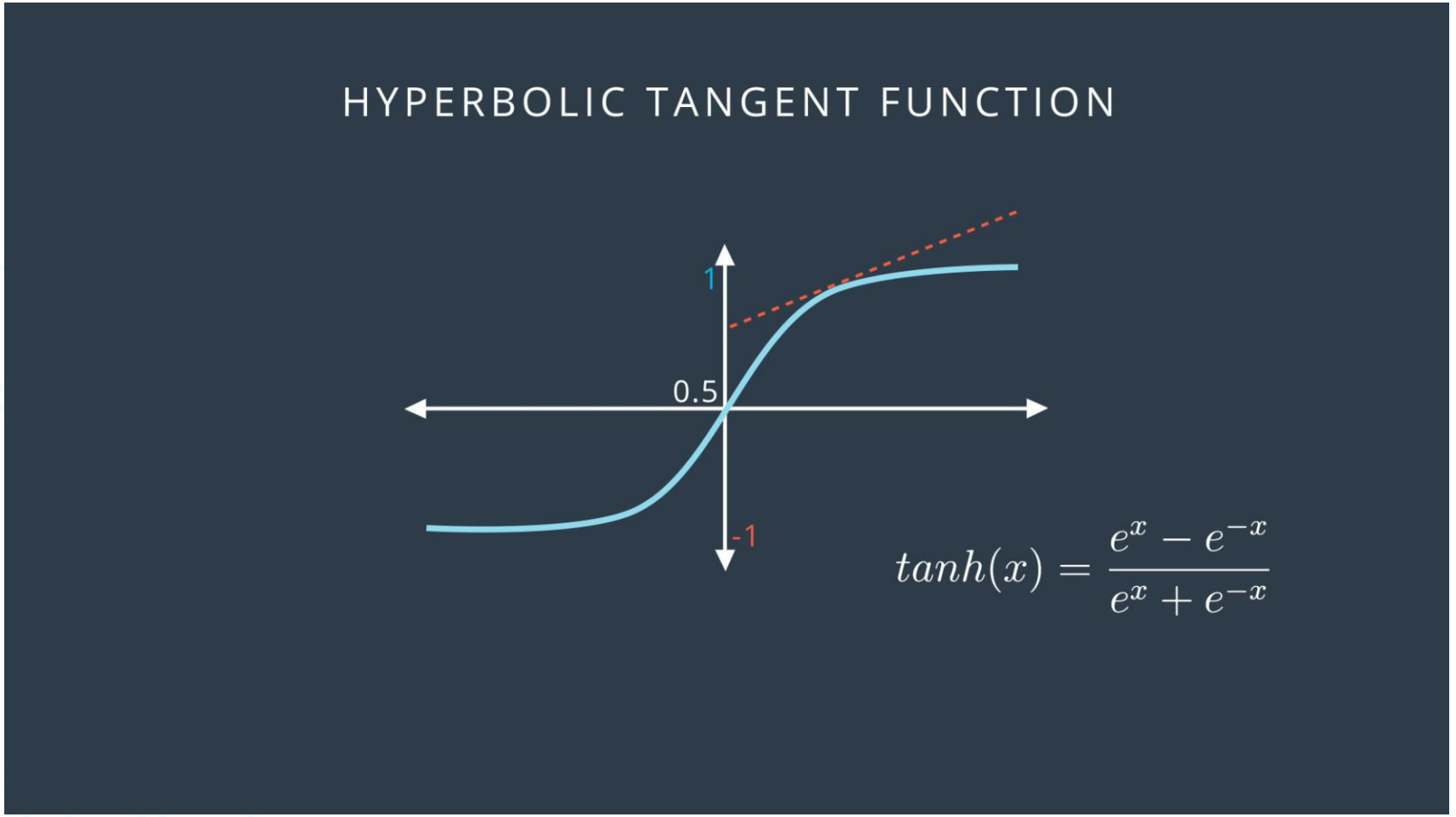
Vanishing Gradient



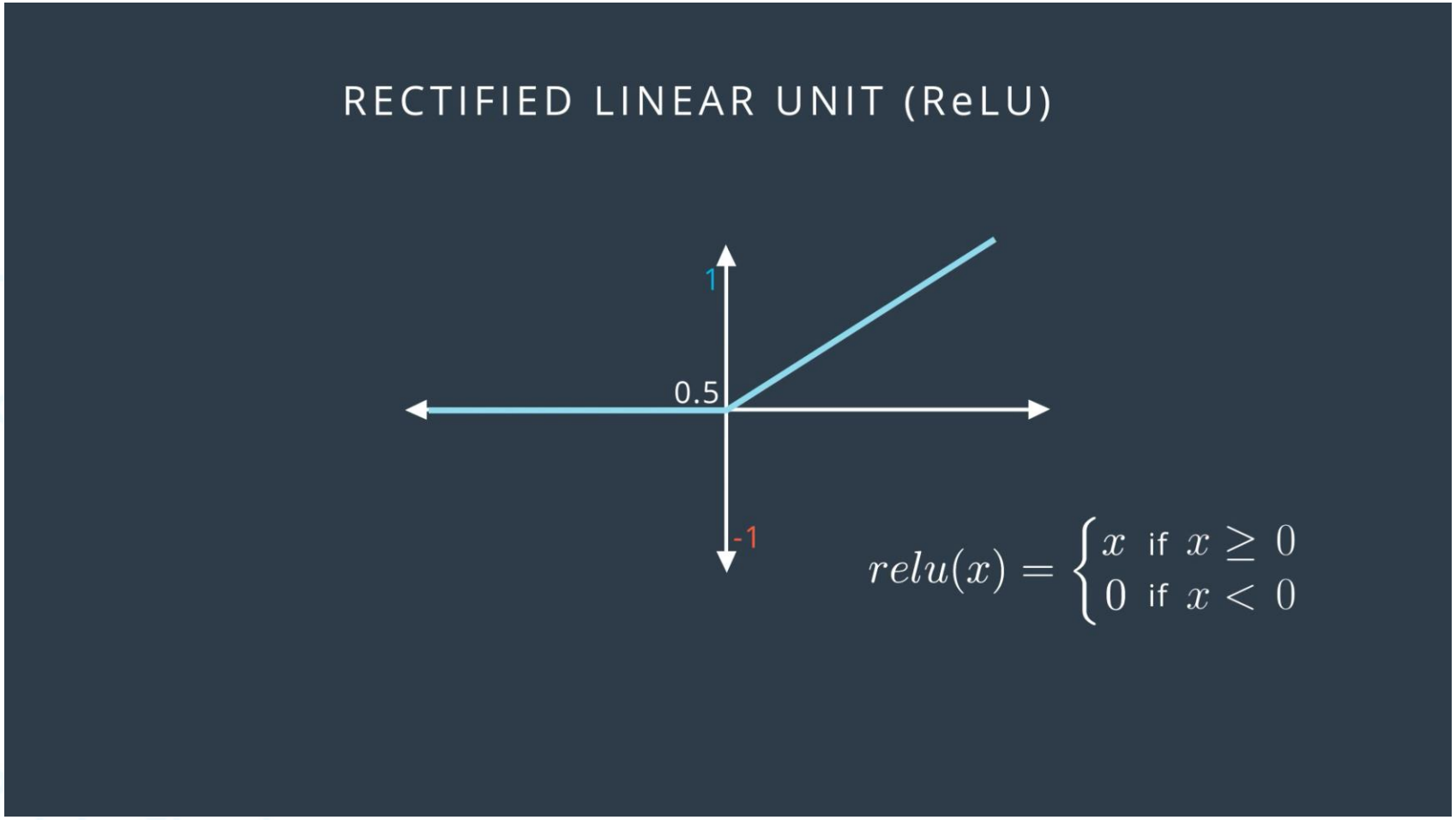
Funções de Ativação



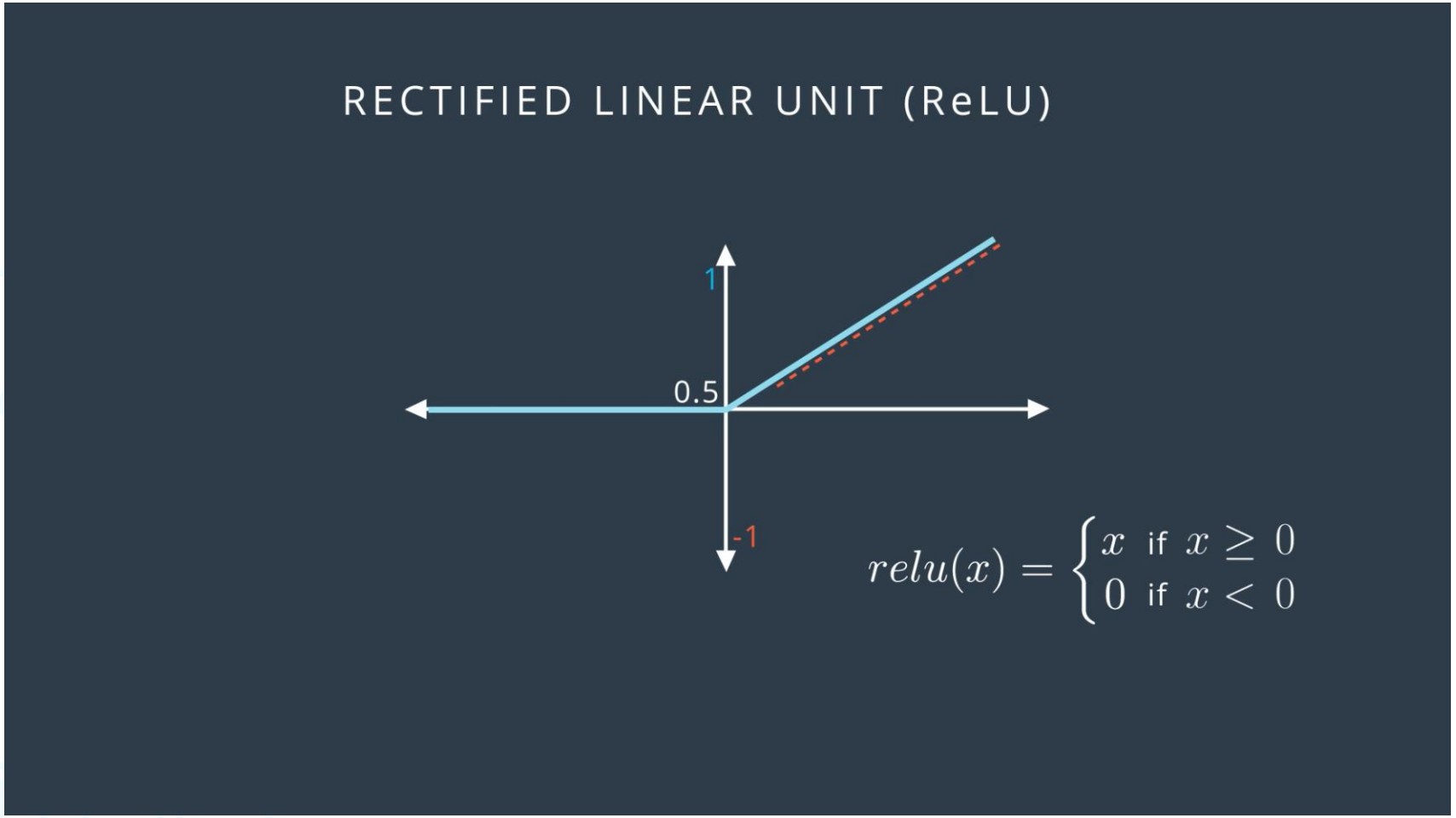
Funções de Ativação



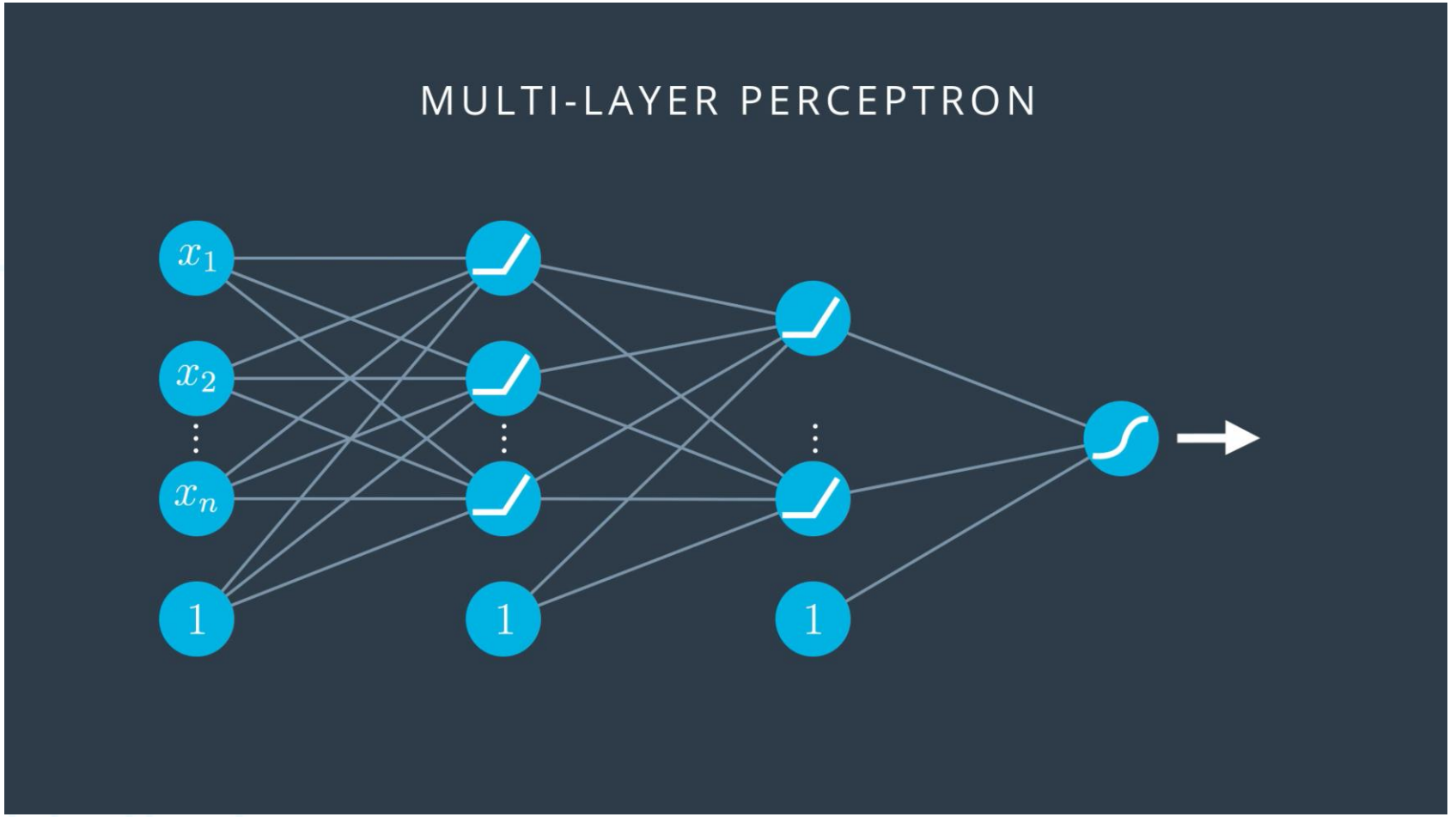
Funções de Ativação



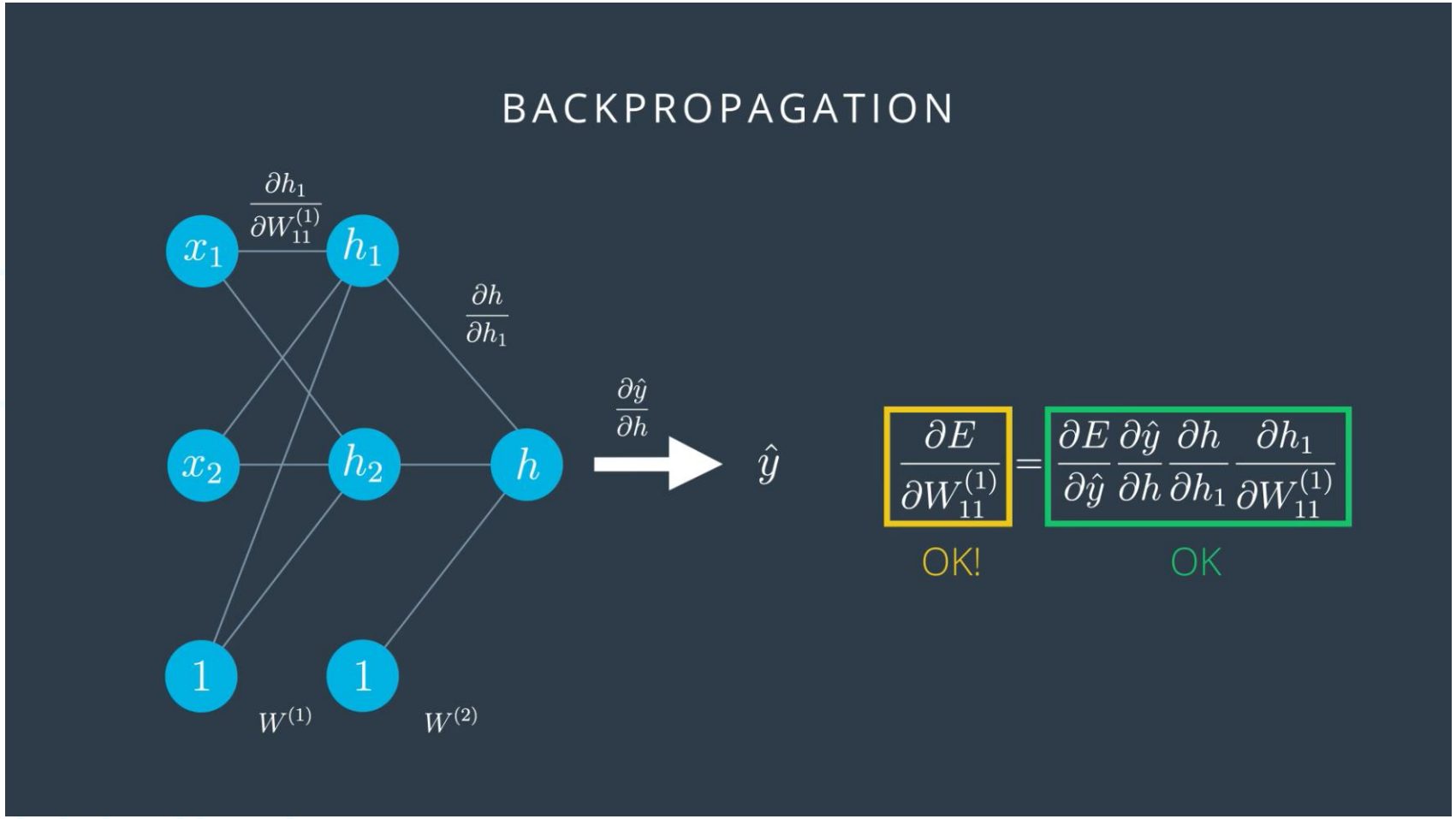
Funções de Ativação



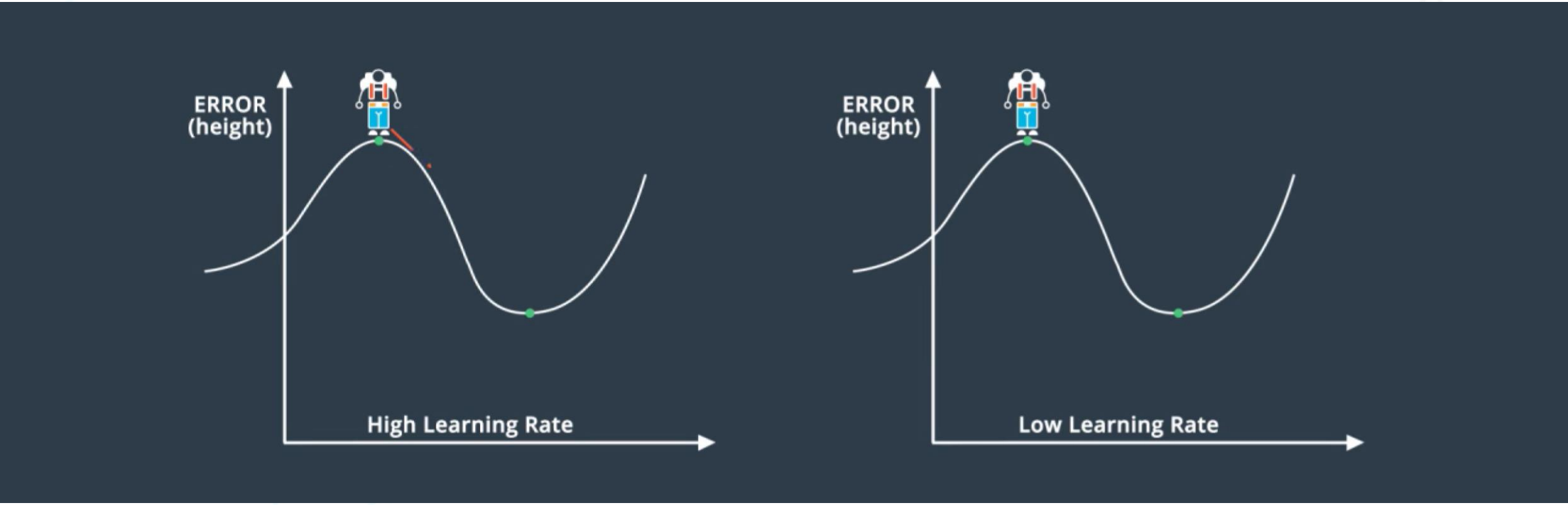
Funções de Ativação



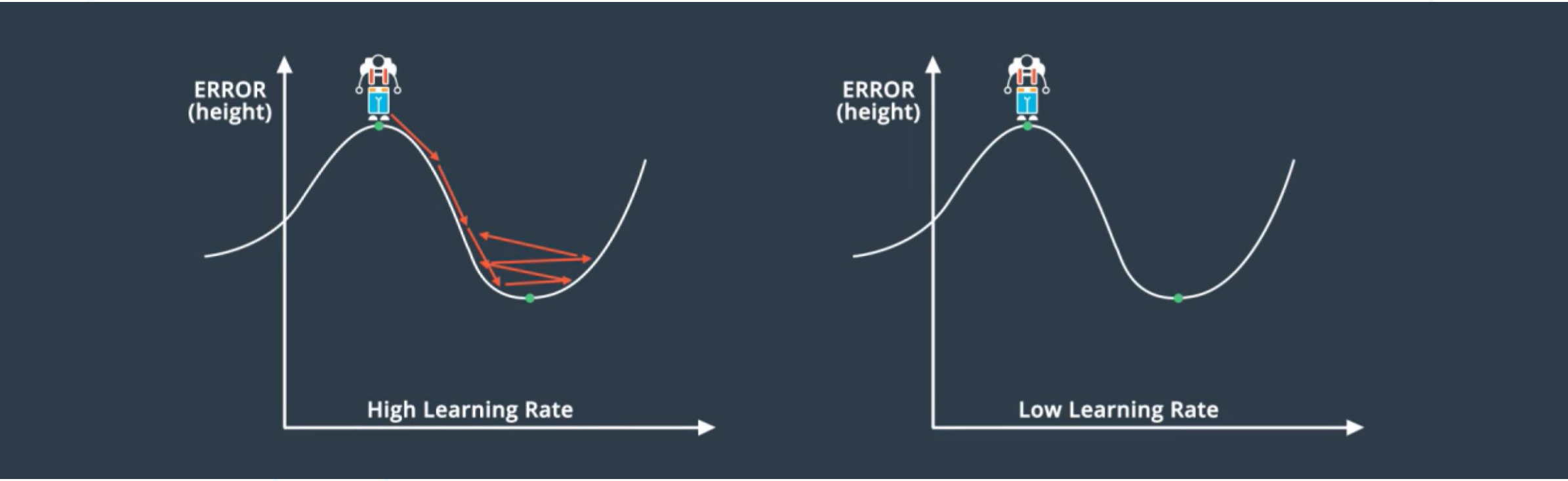
Funções de Ativação



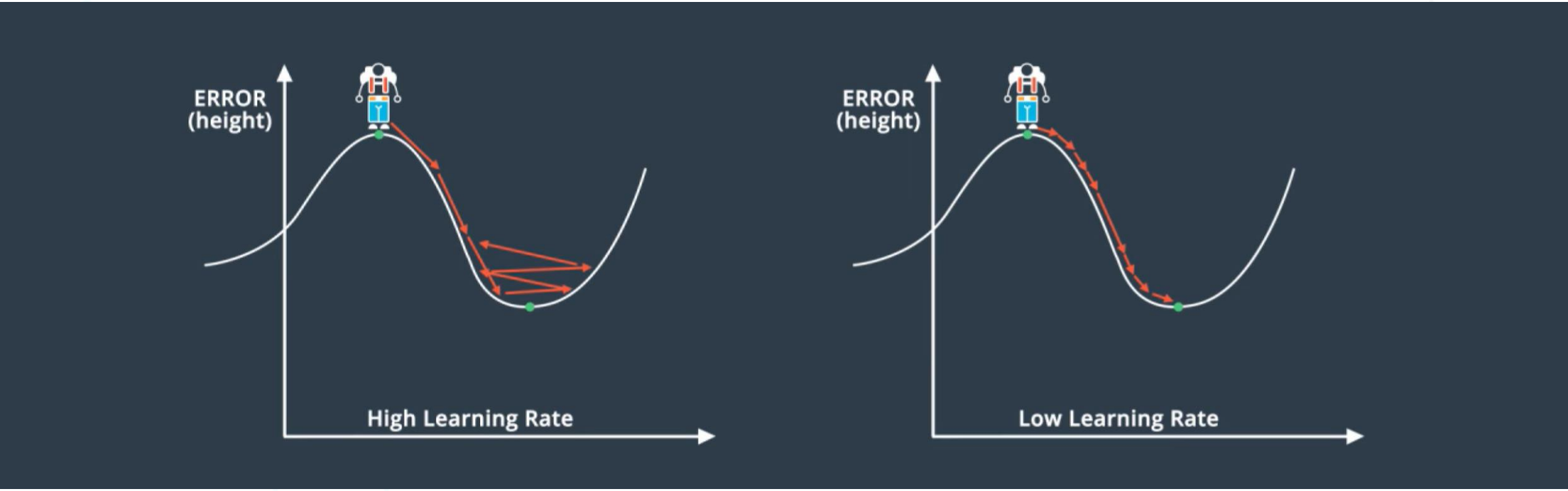
Learning Rate



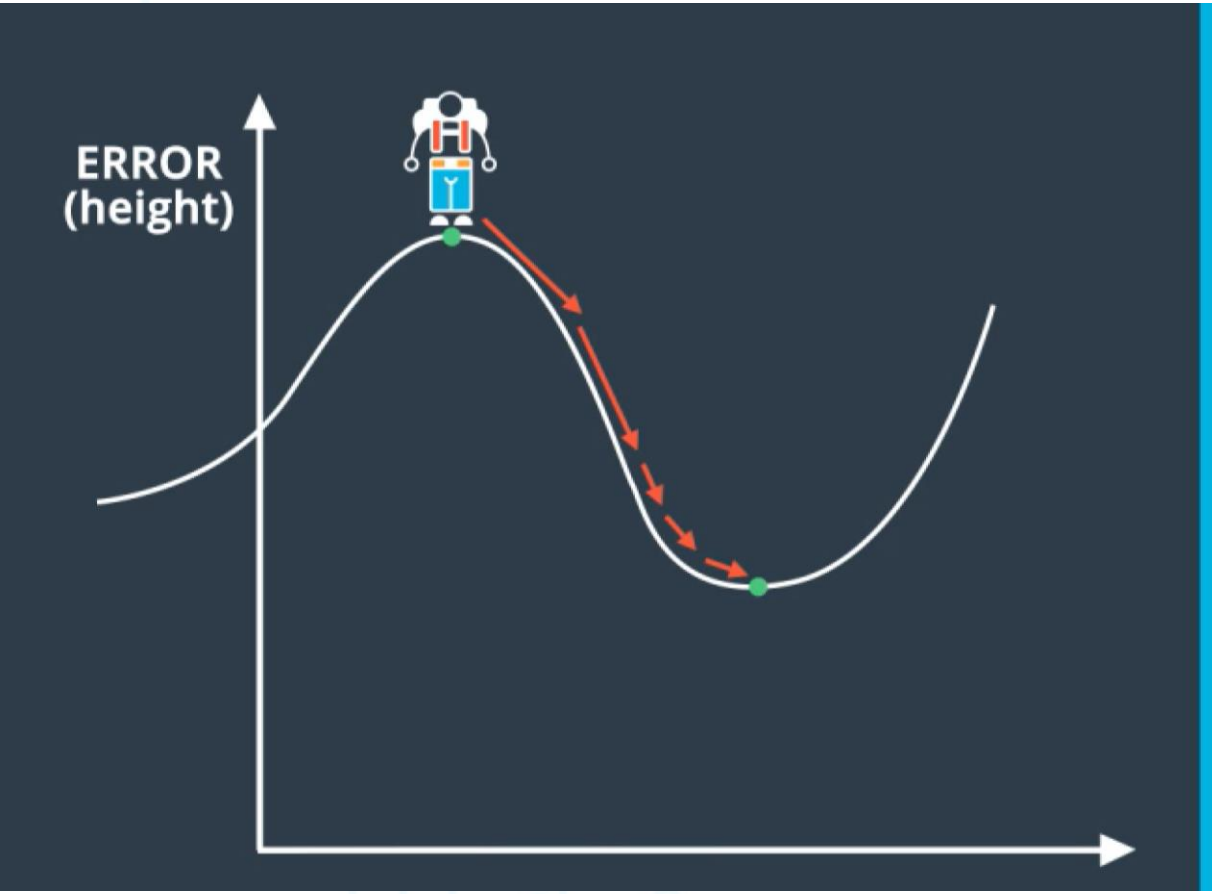
Learning Rate



Learning Rate



Learning Rate

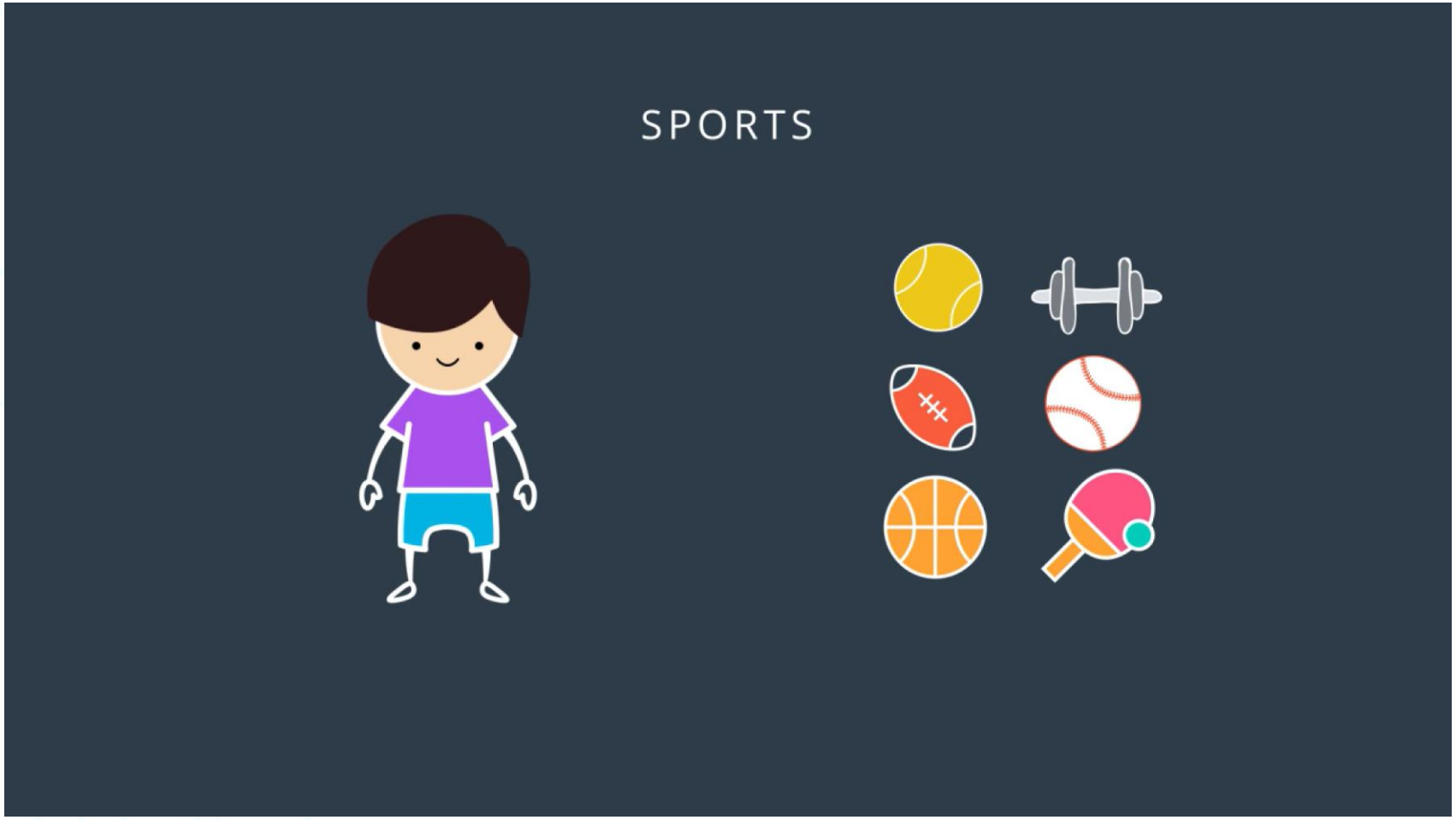


Decreasing Learning Rate

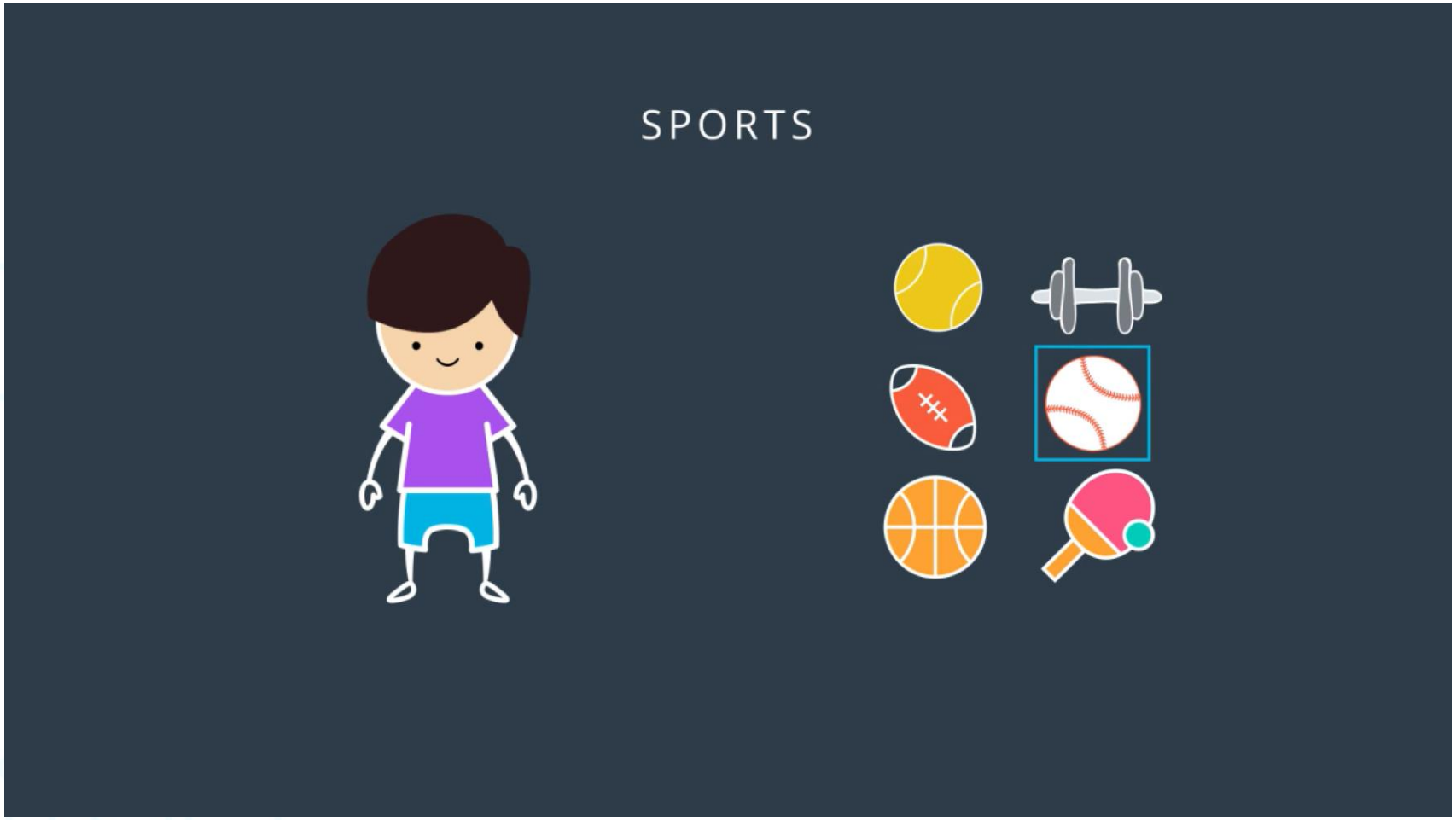
Rule:

- If steep: long steps
- If plain: small steps

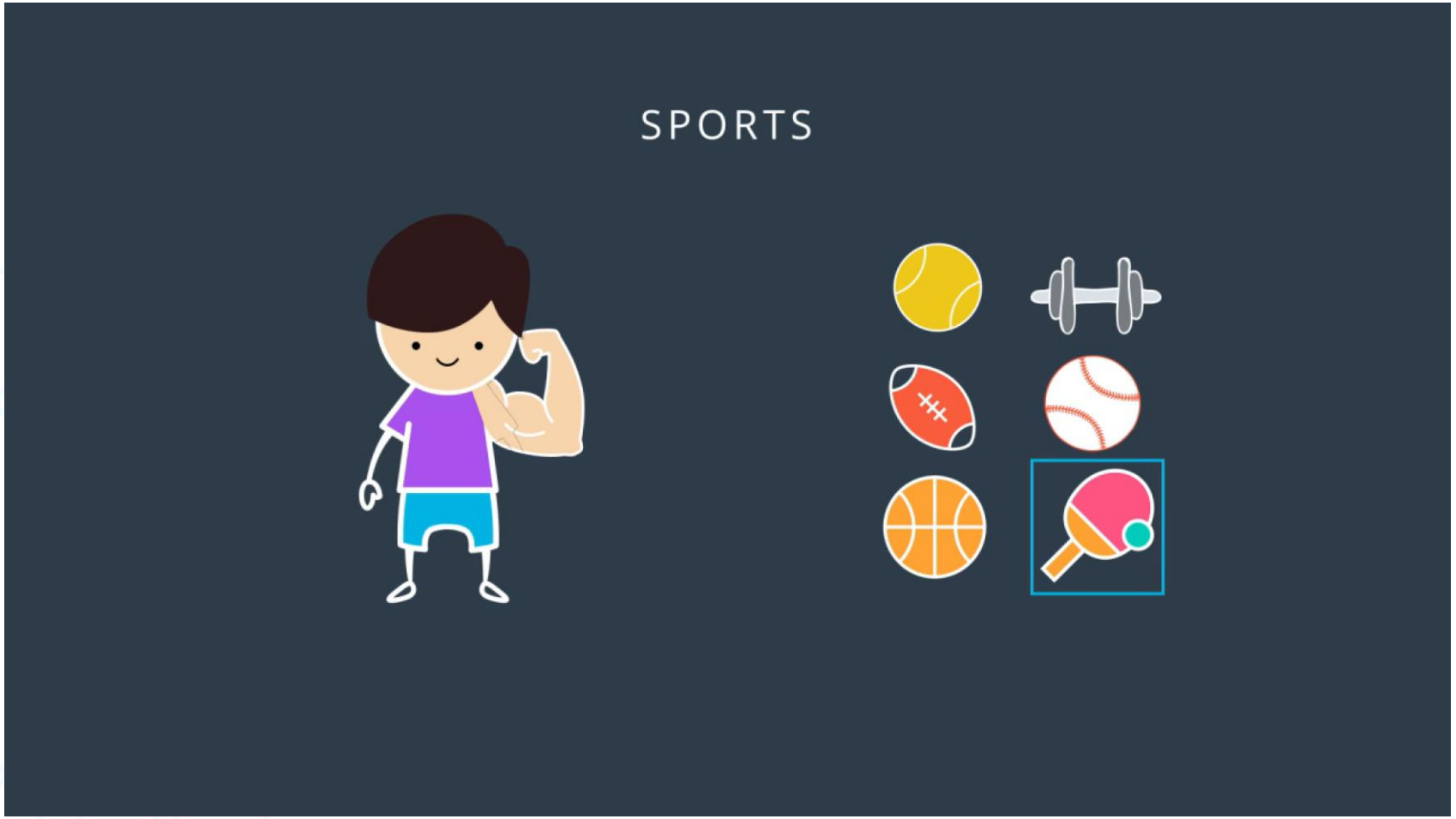
Redes Neurais – Dropout



Redes Neurais – Dropout

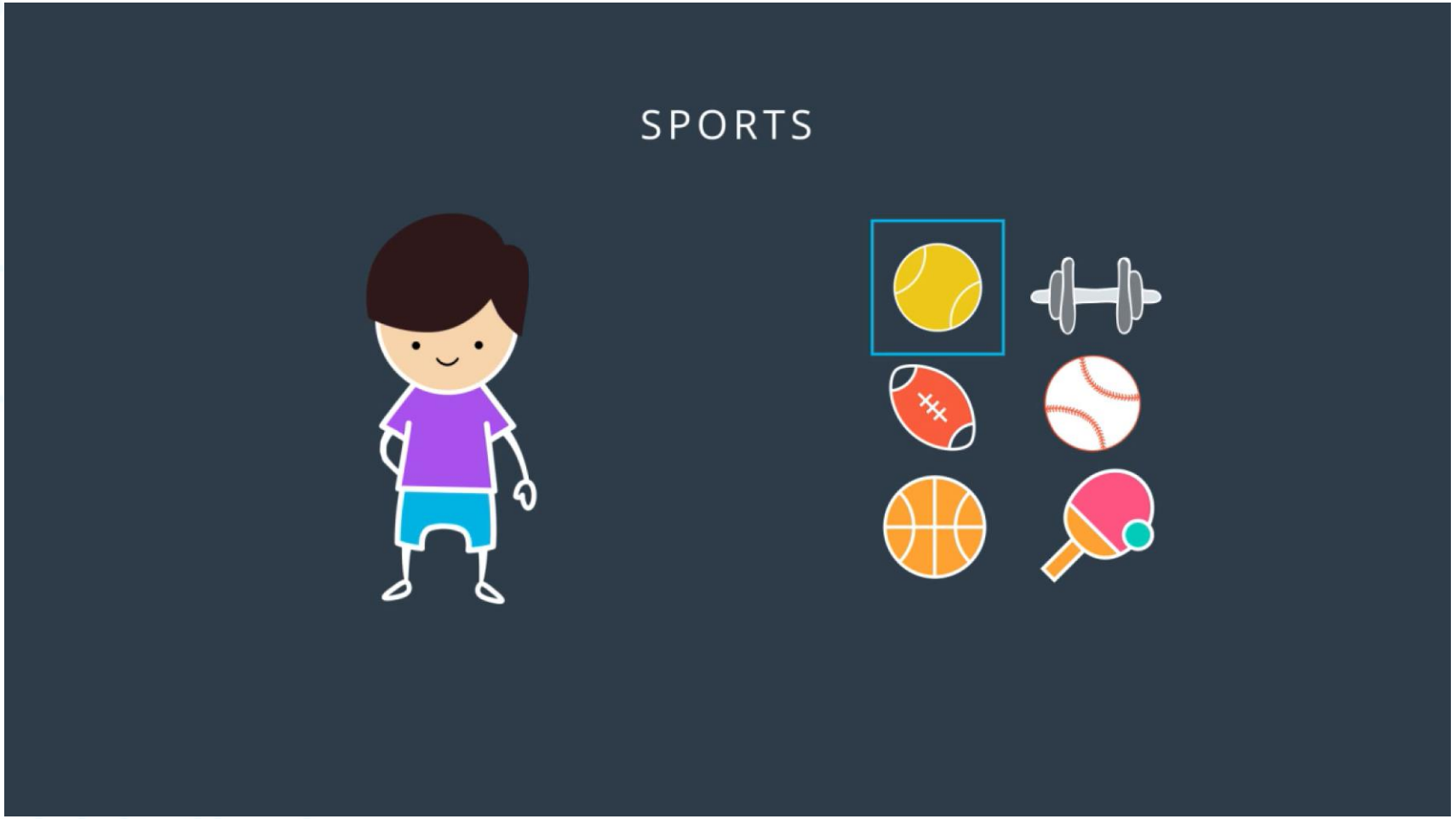


Redes Neurais – Dropout

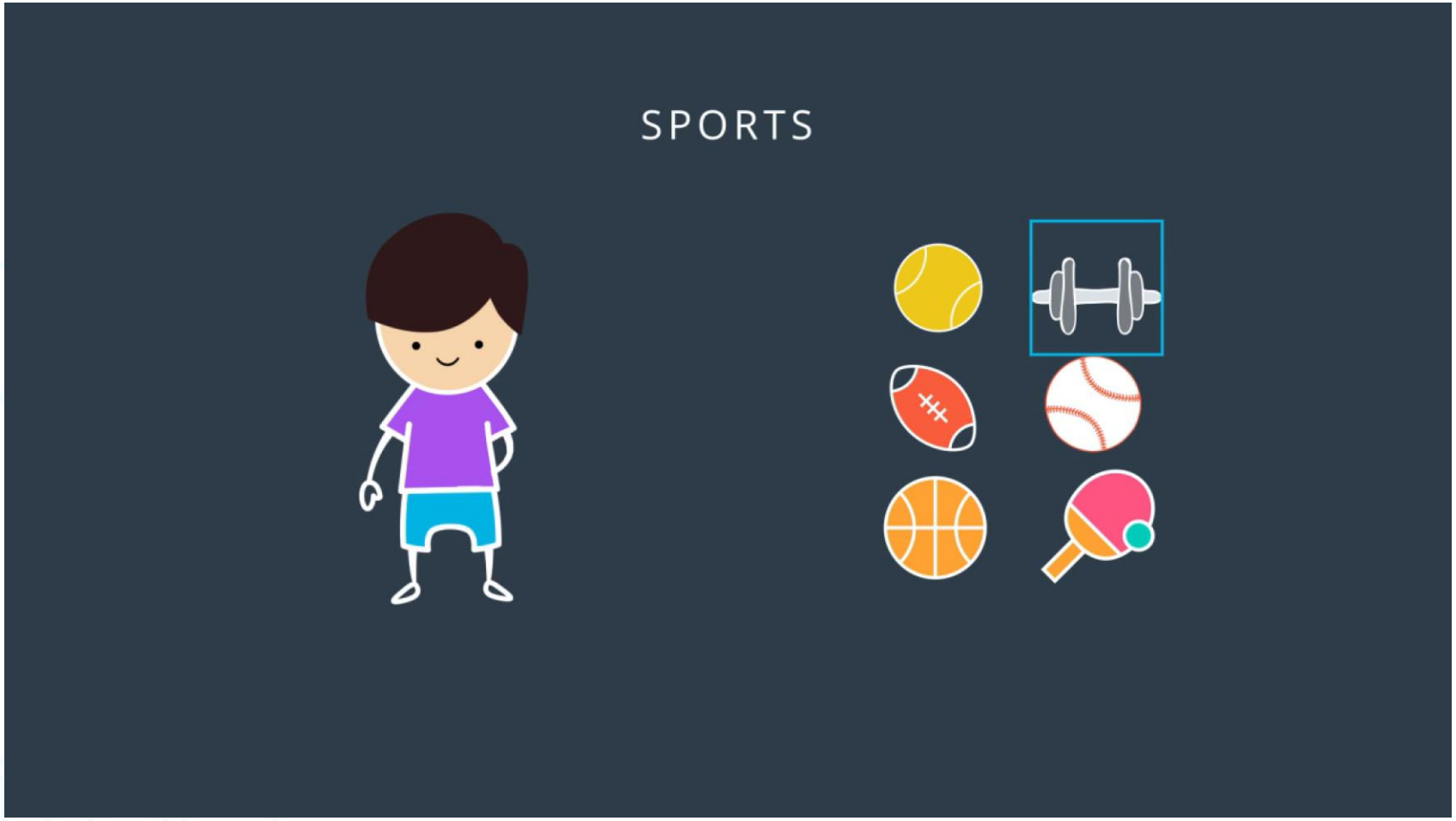


Redes Neurais – Dropout

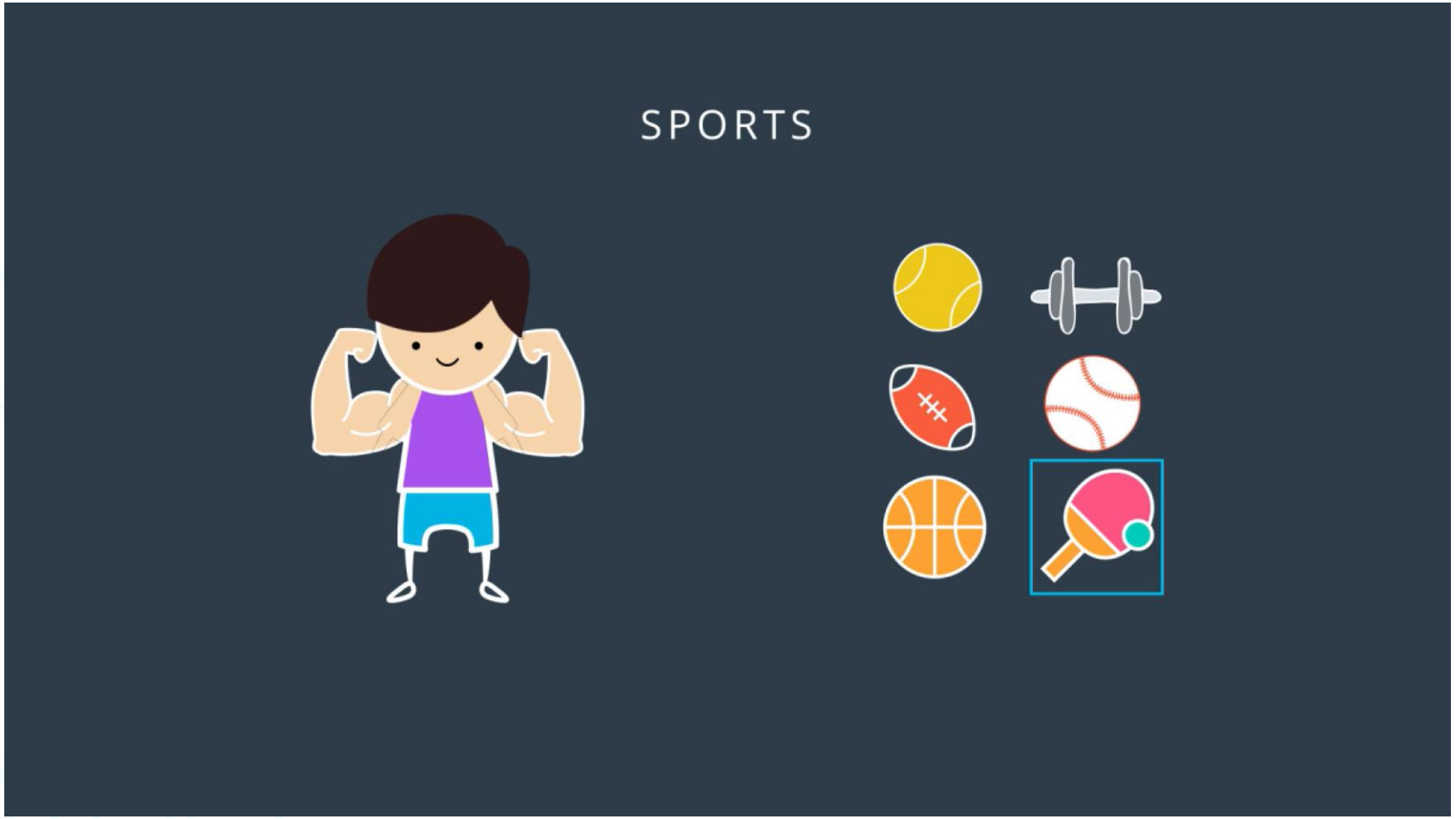
Redes Neurais – Dropout



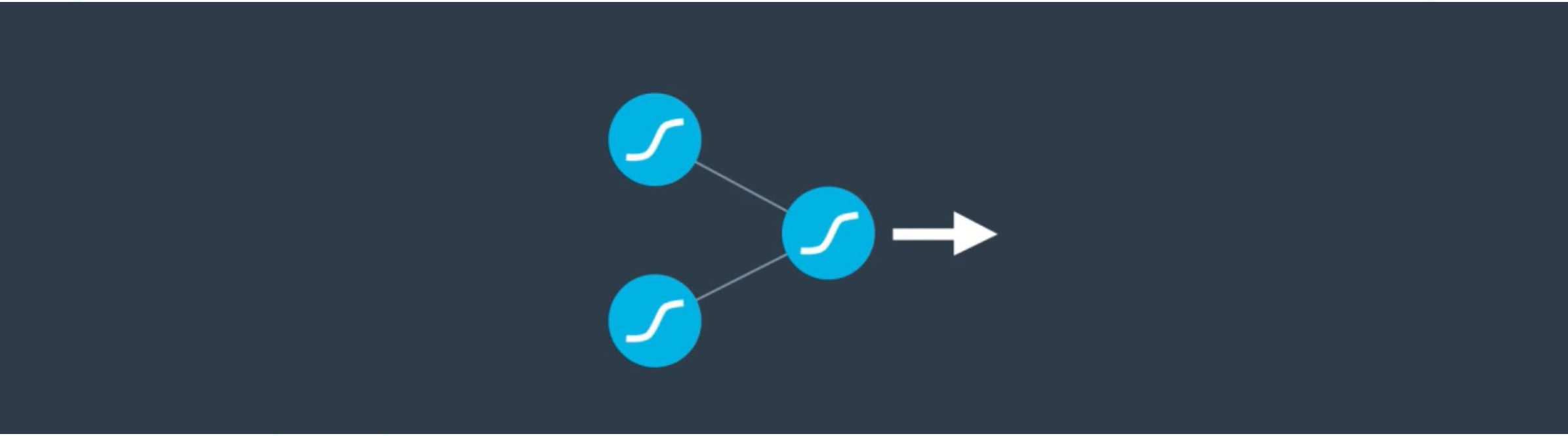
Redes Neurais – Dropout



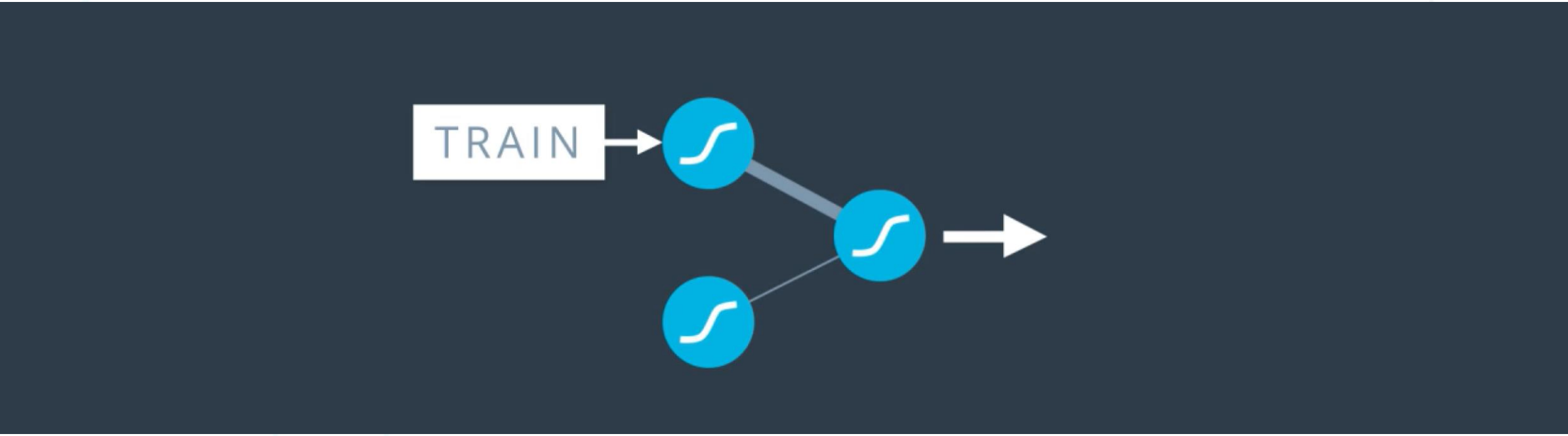
Redes Neurais – Dropout



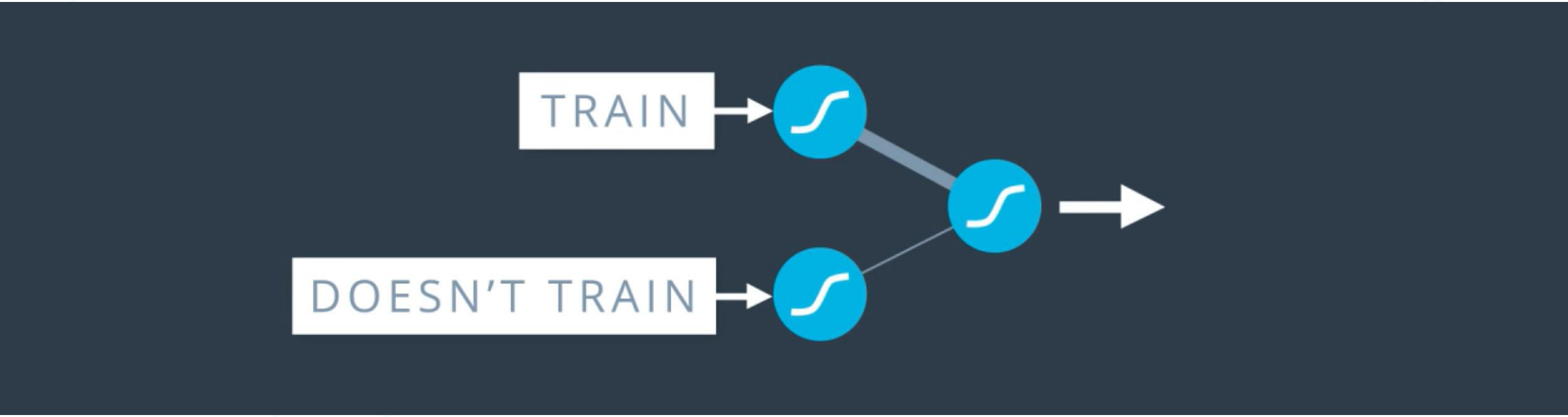
Redes Neurais – Dropout



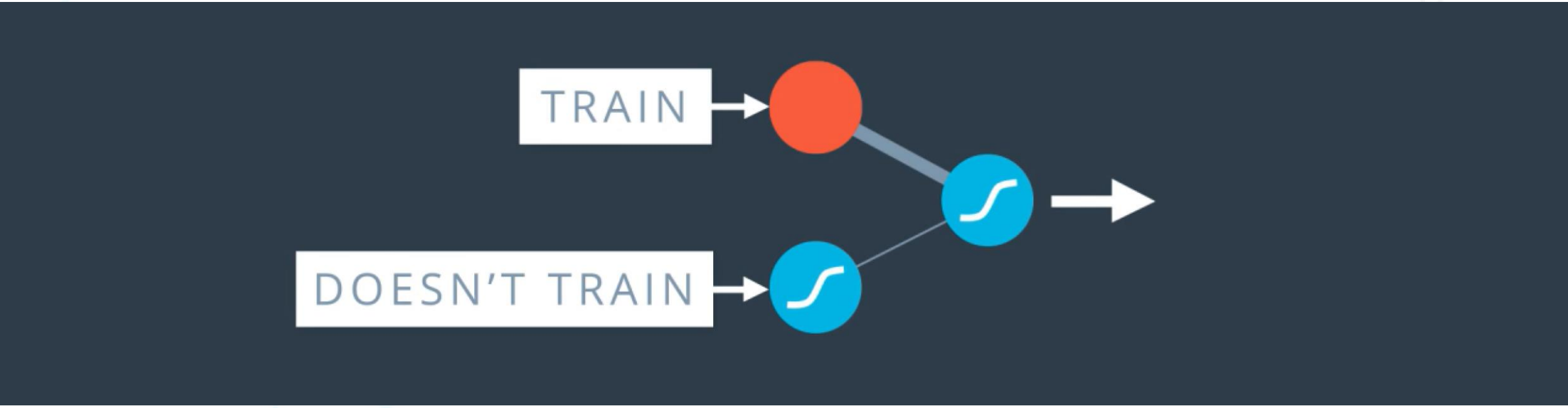
Redes Neurais – Dropout



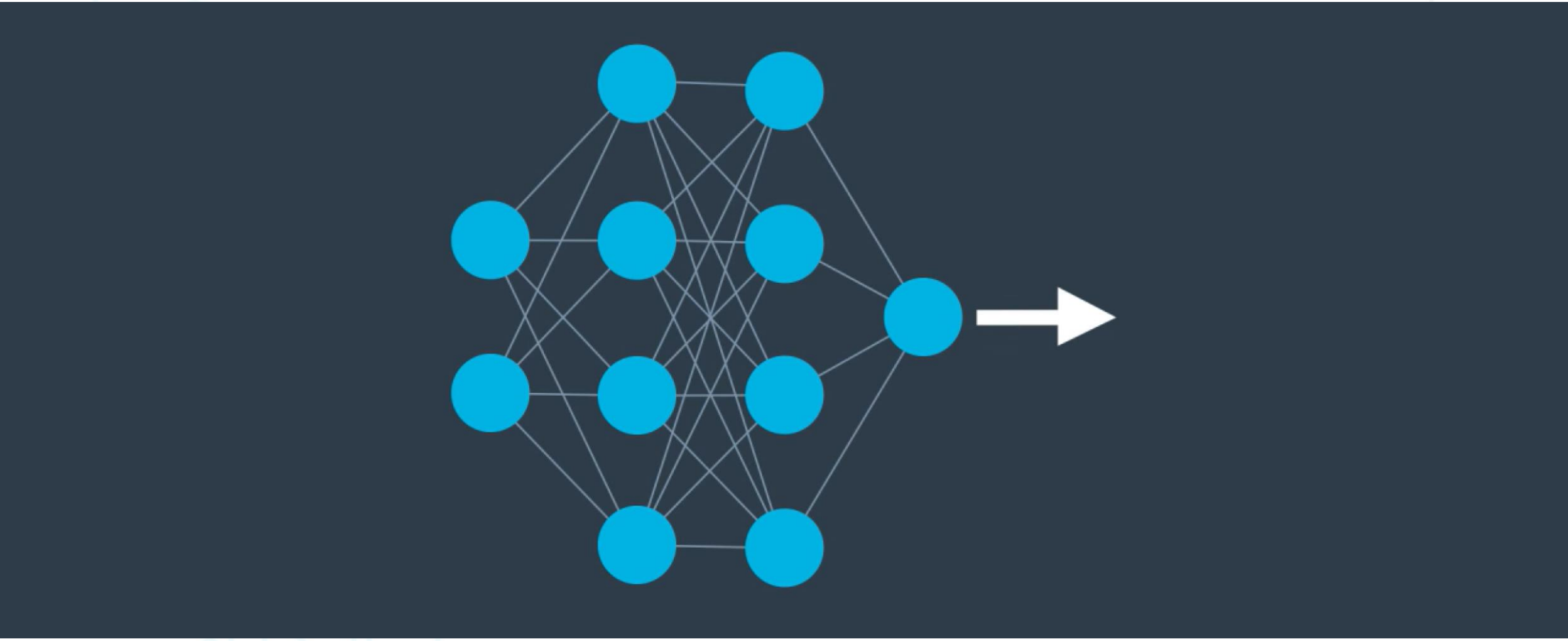
Redes Neurais – Dropout



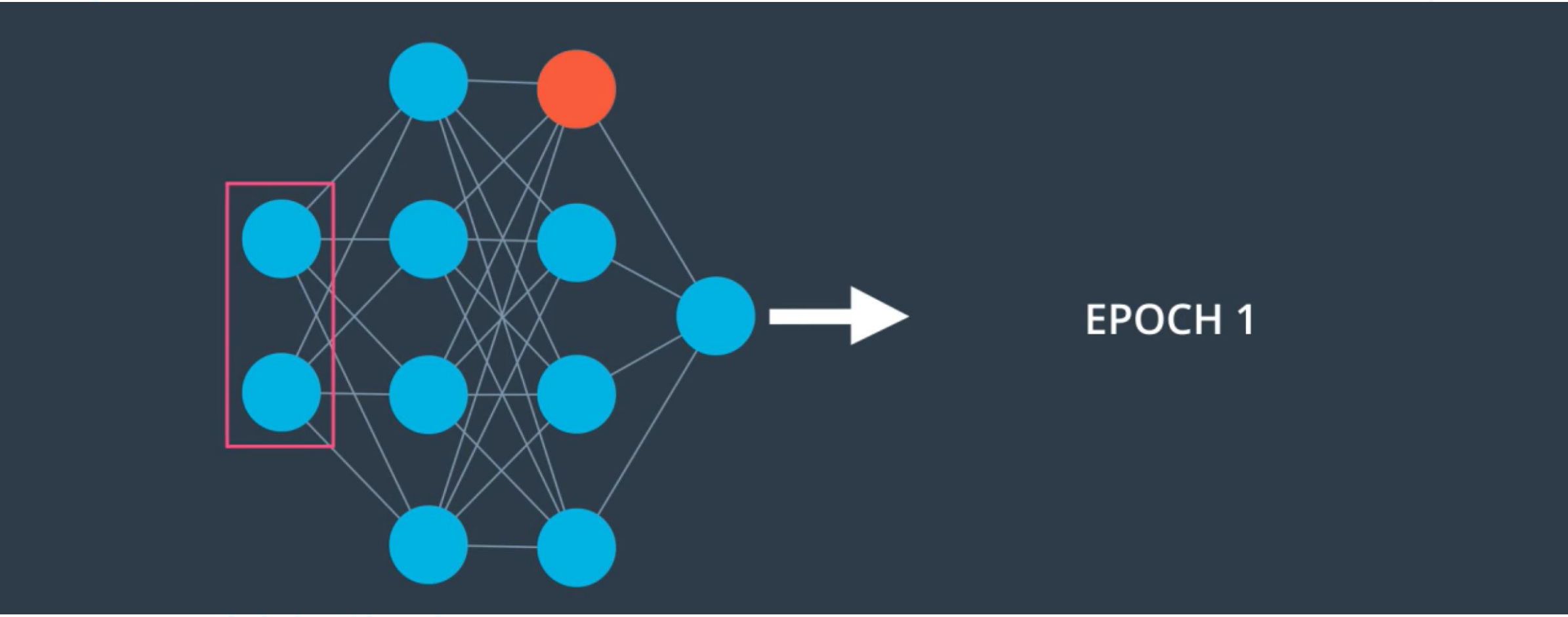
Redes Neurais – Dropout



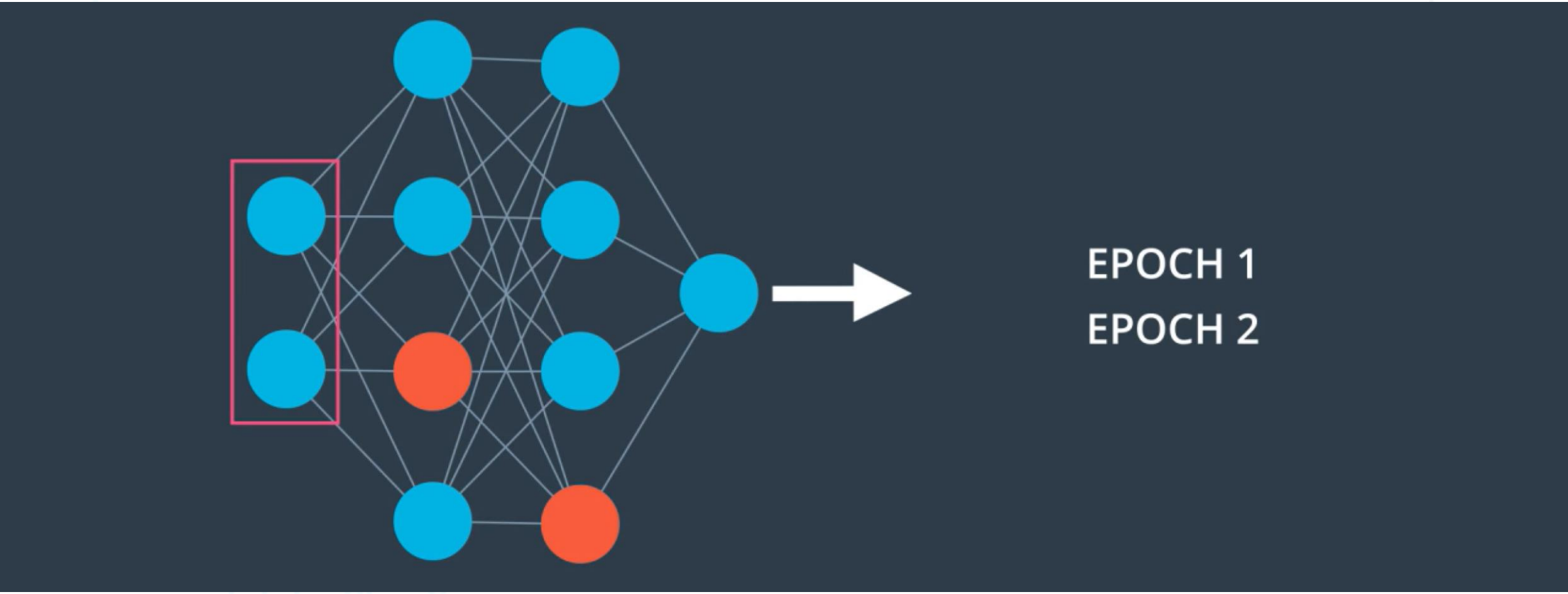
Redes Neurais – Dropout



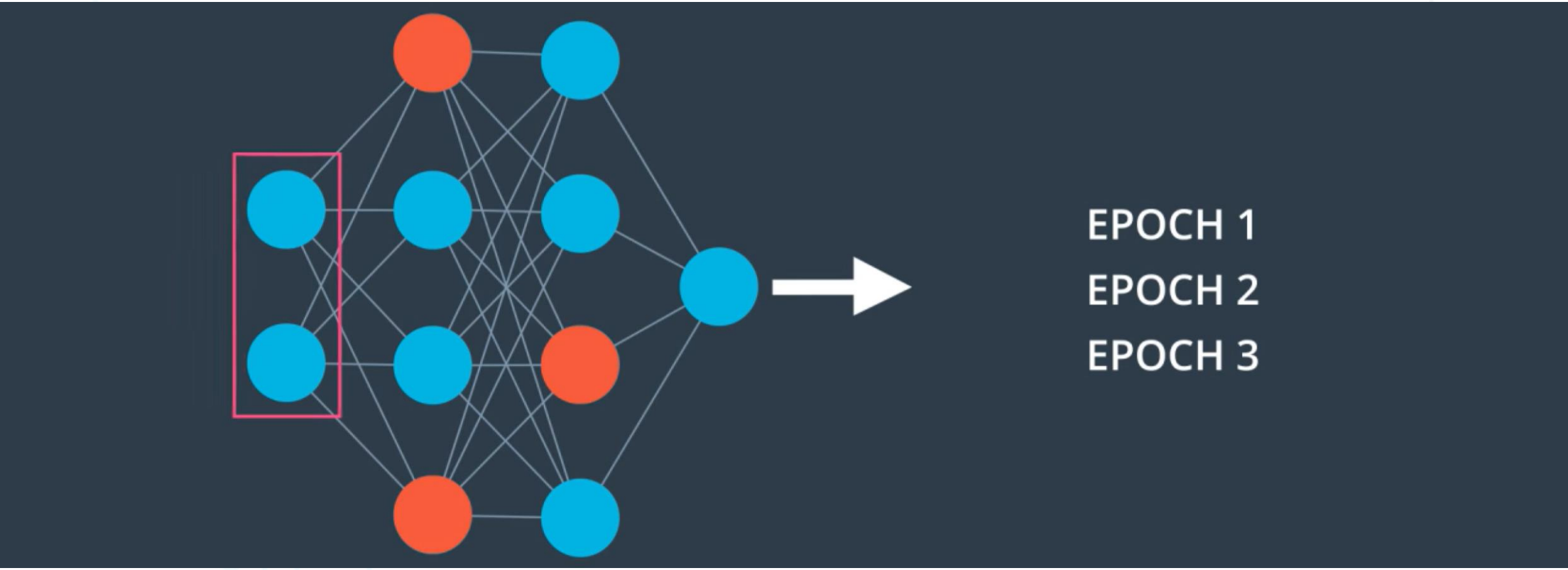
Redes Neurais – Dropout



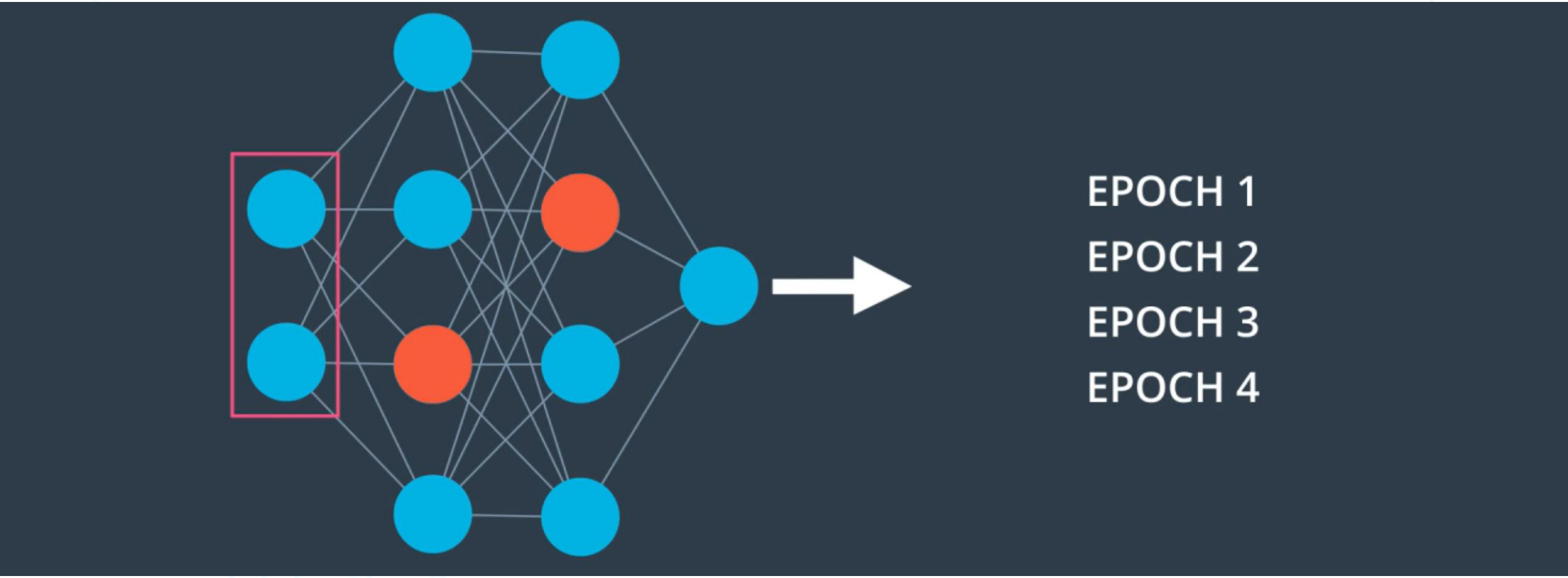
Redes Neurais – Dropout



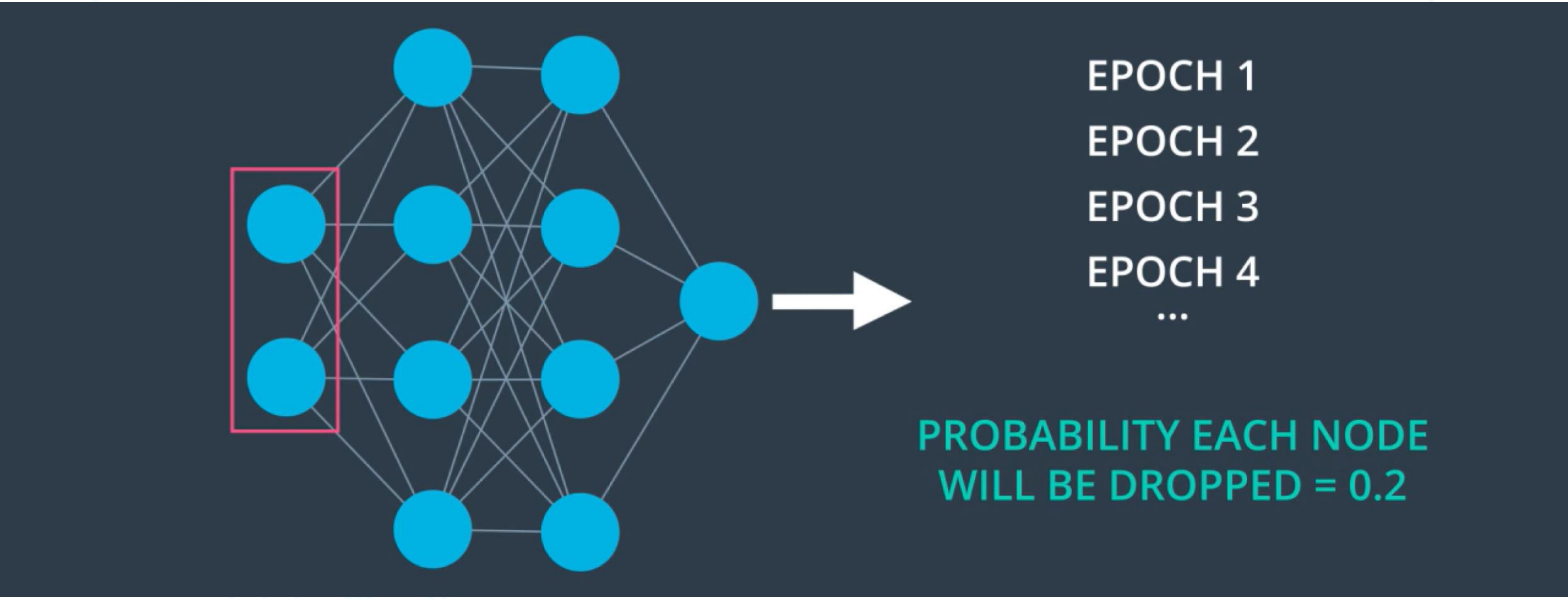
Redes Neurais – Dropout



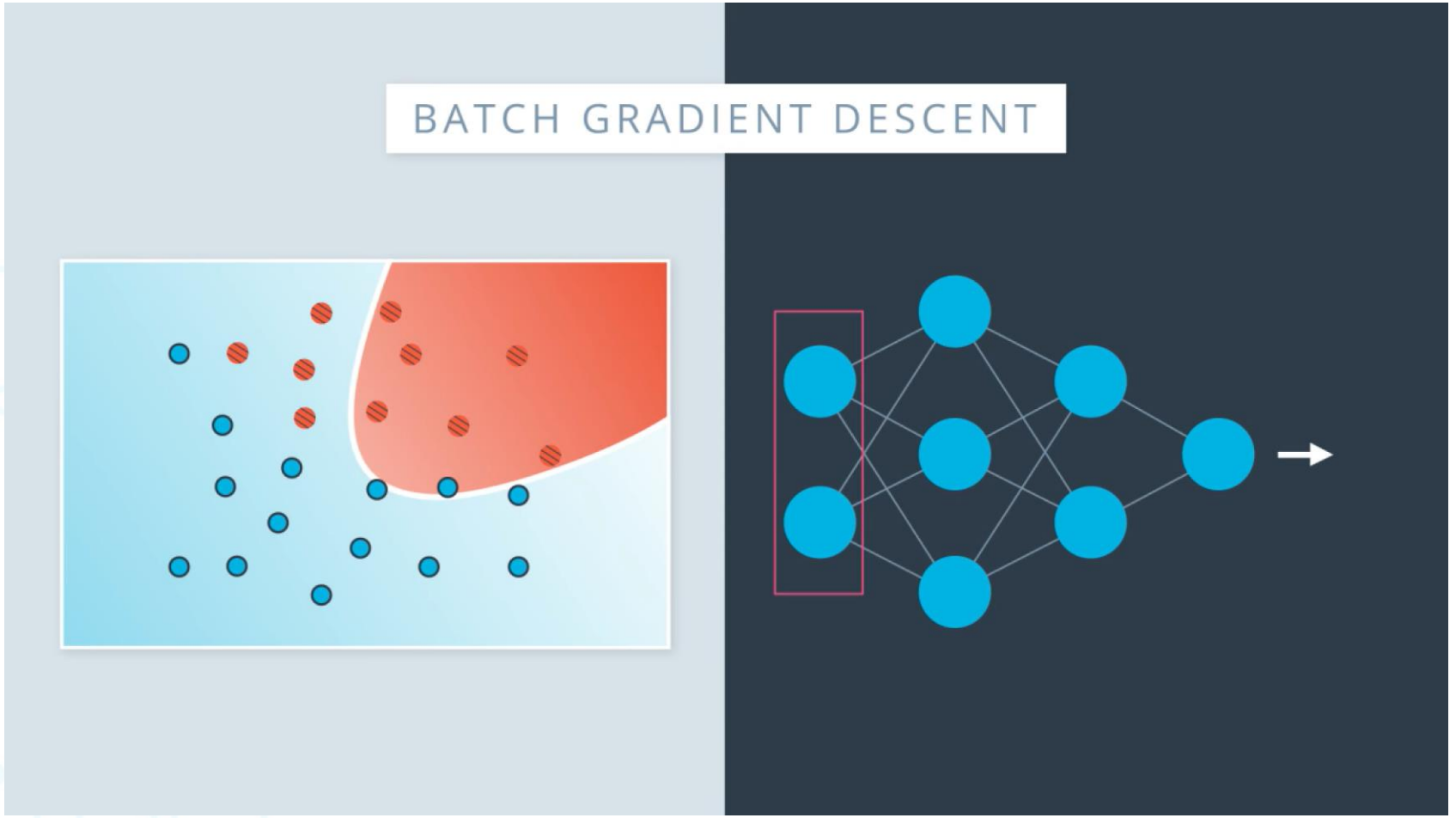
Redes Neurais – Dropout



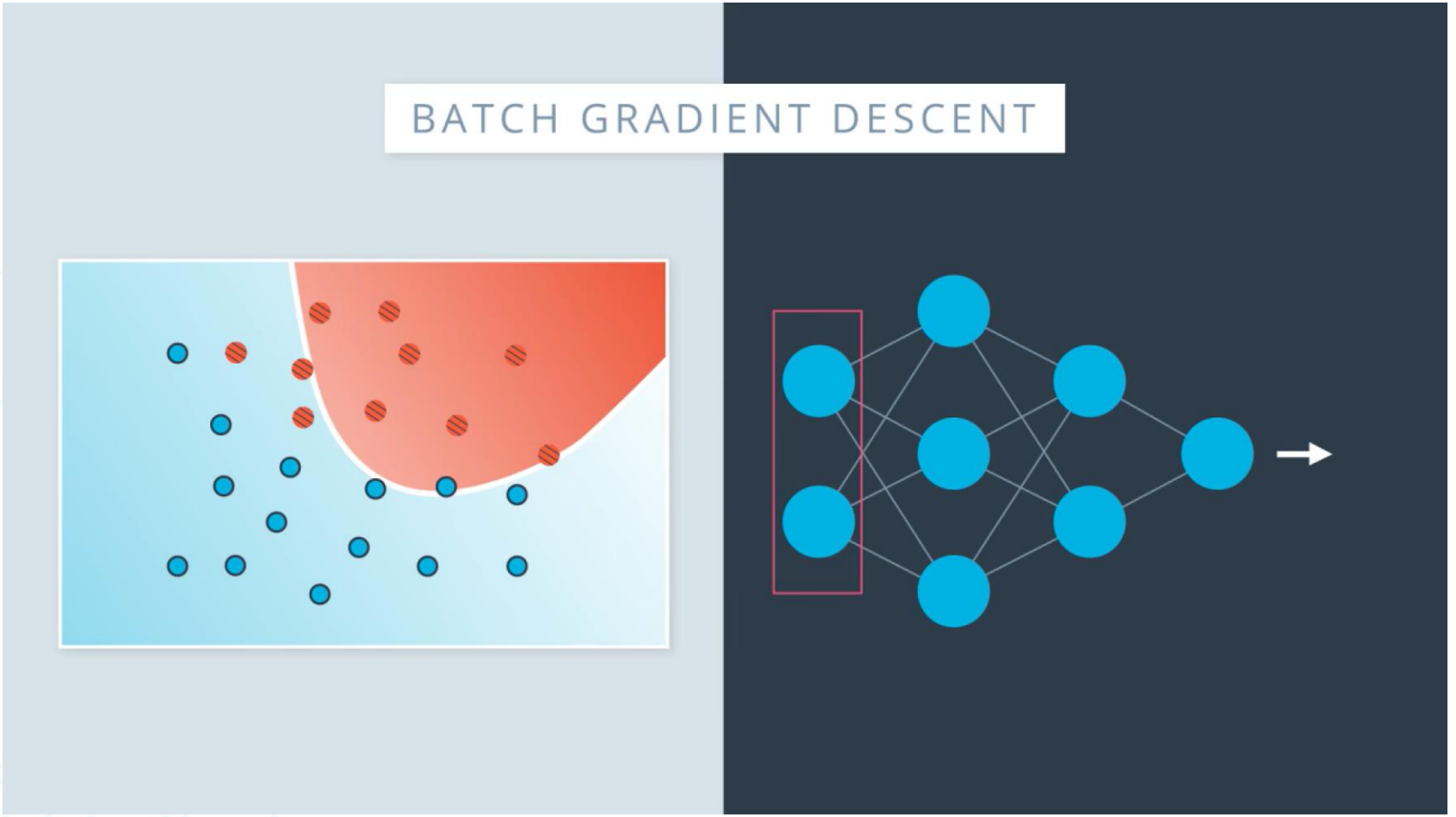
Redes Neurais – Dropout



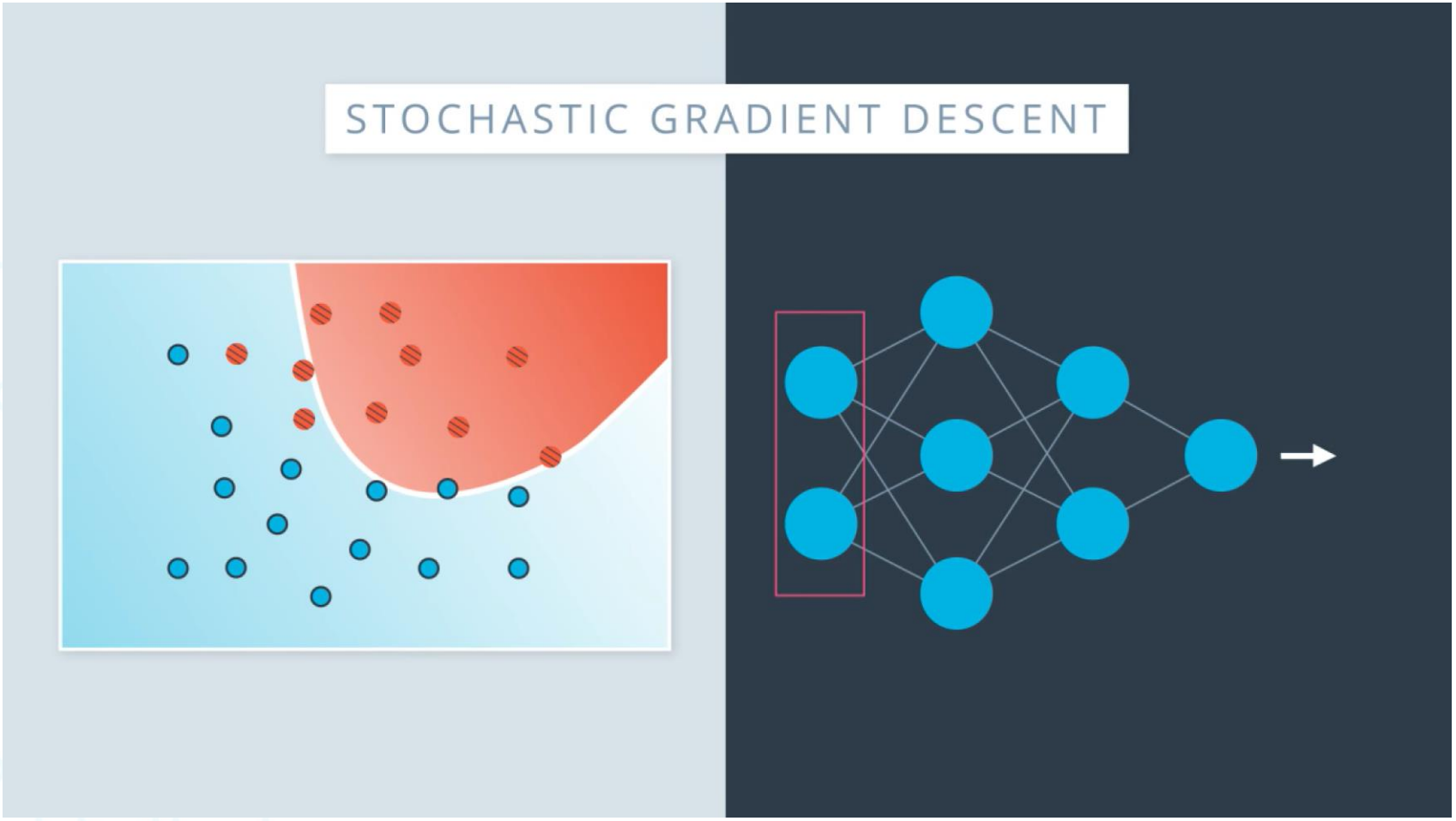
Stochastic Gradient Descent



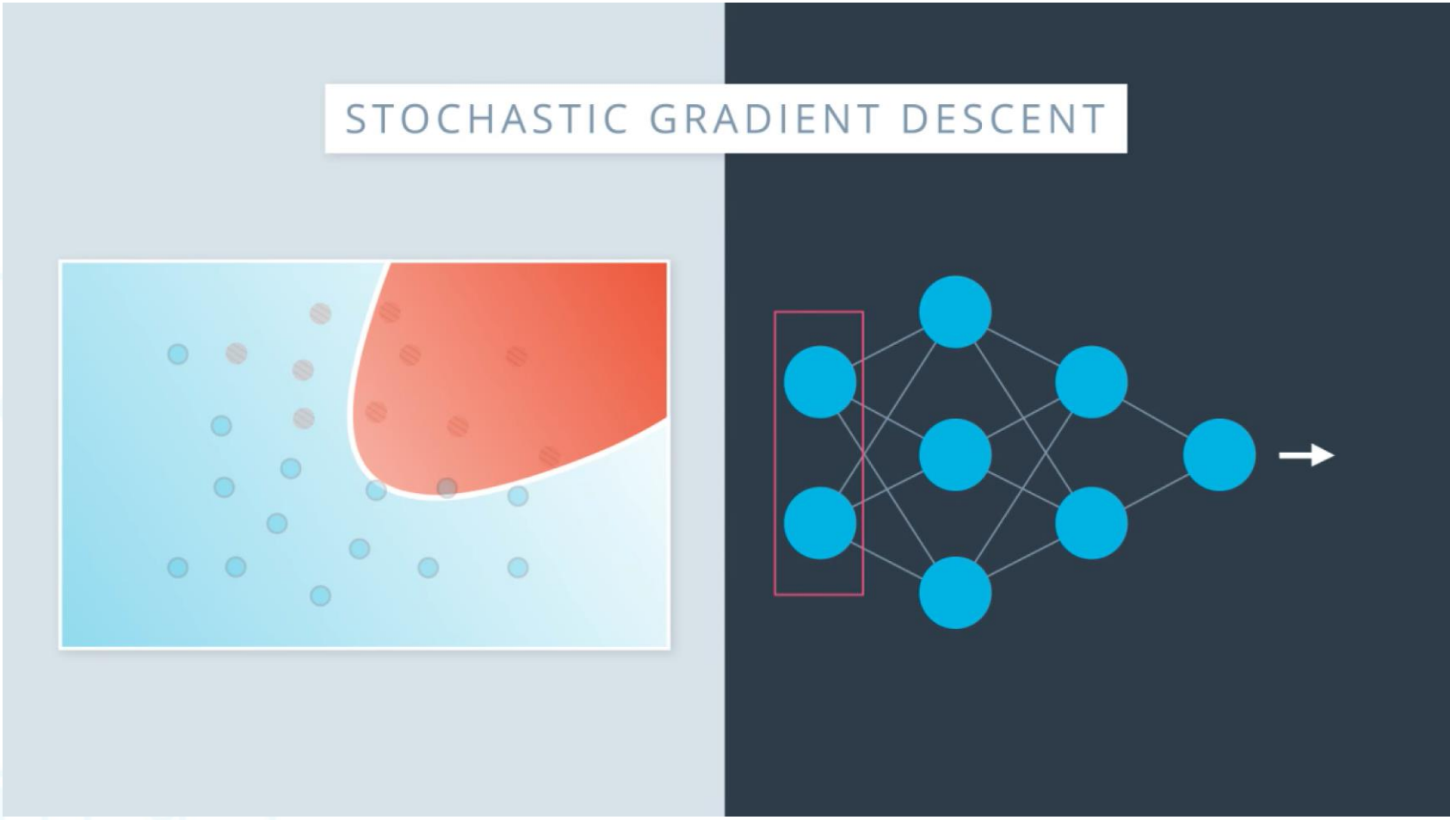
Stochastic Gradient Descent



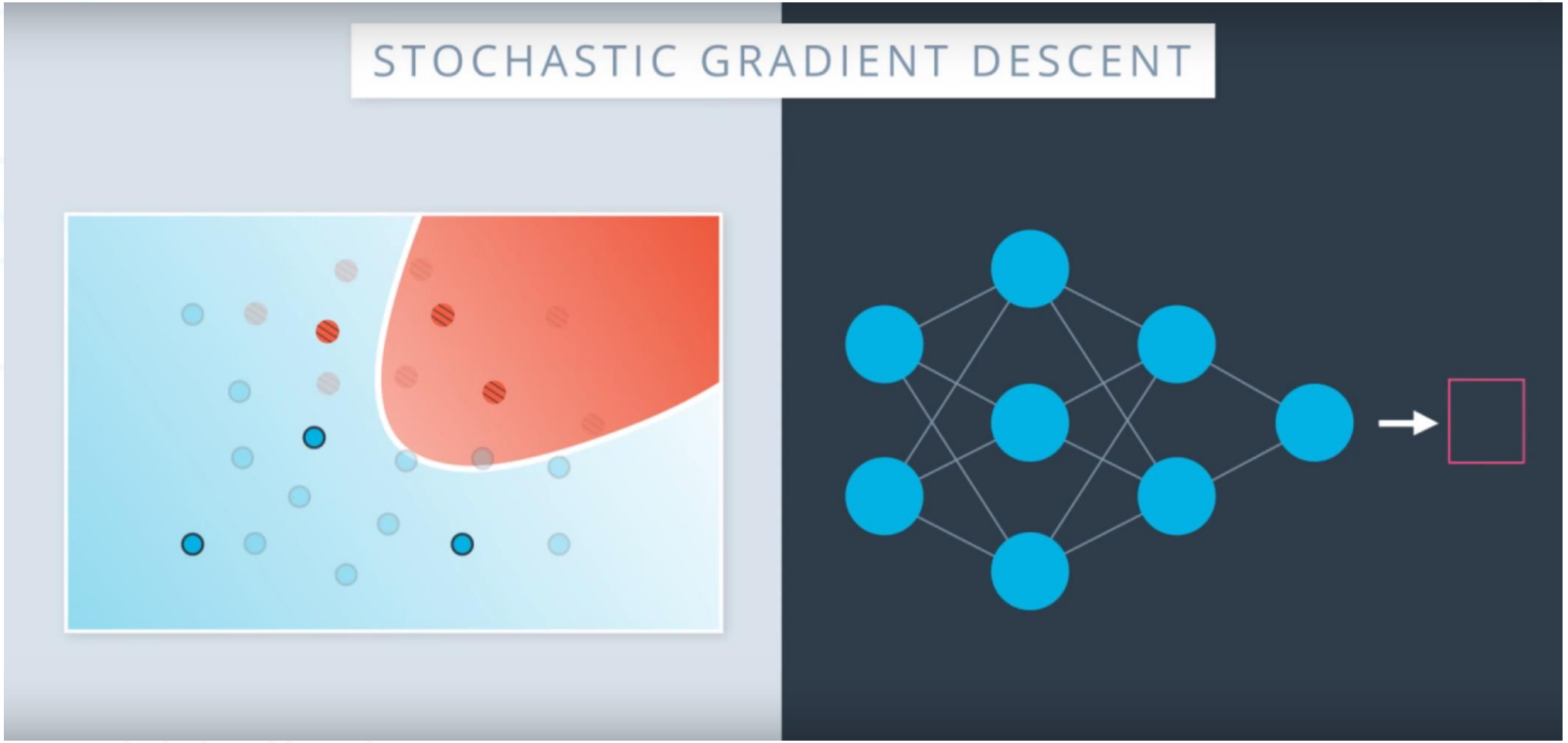
Stochastic Gradient Descent



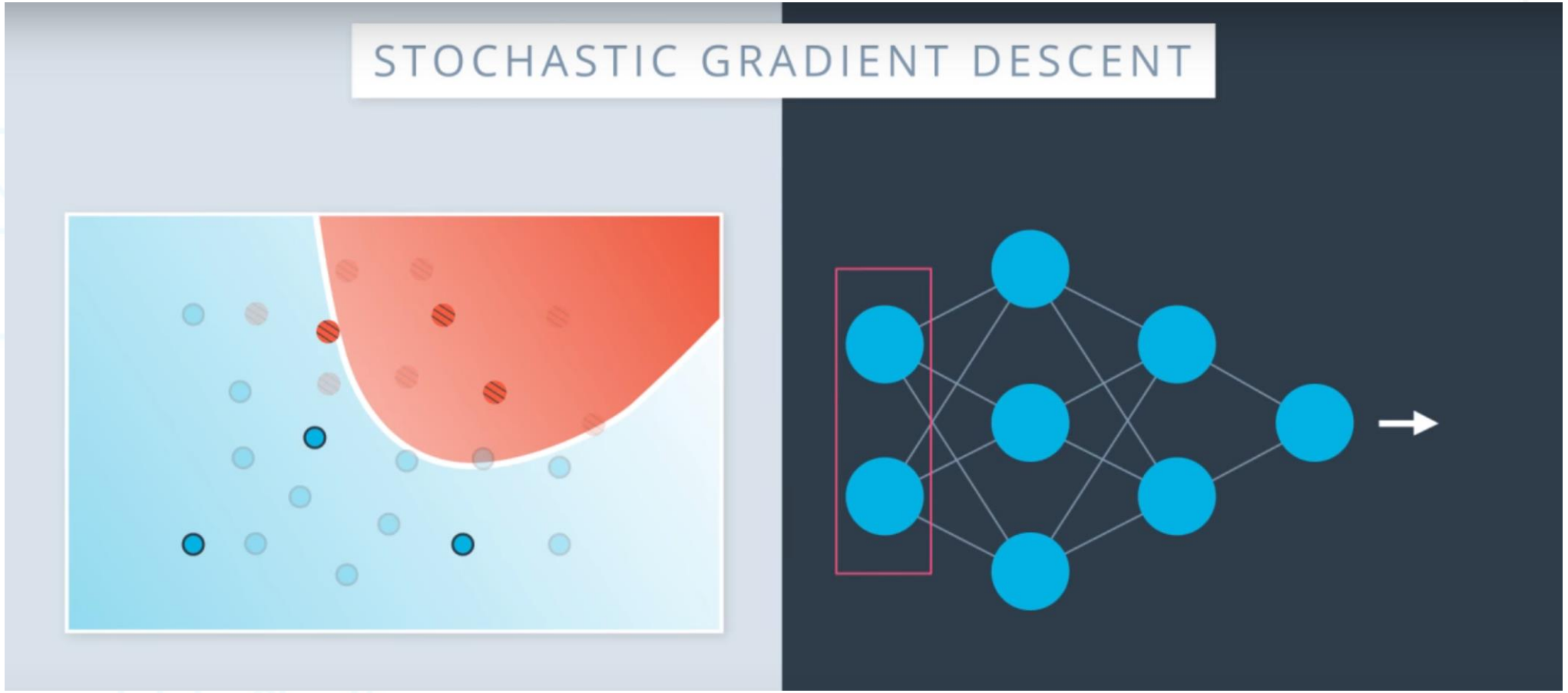
Stochastic Gradient Descent



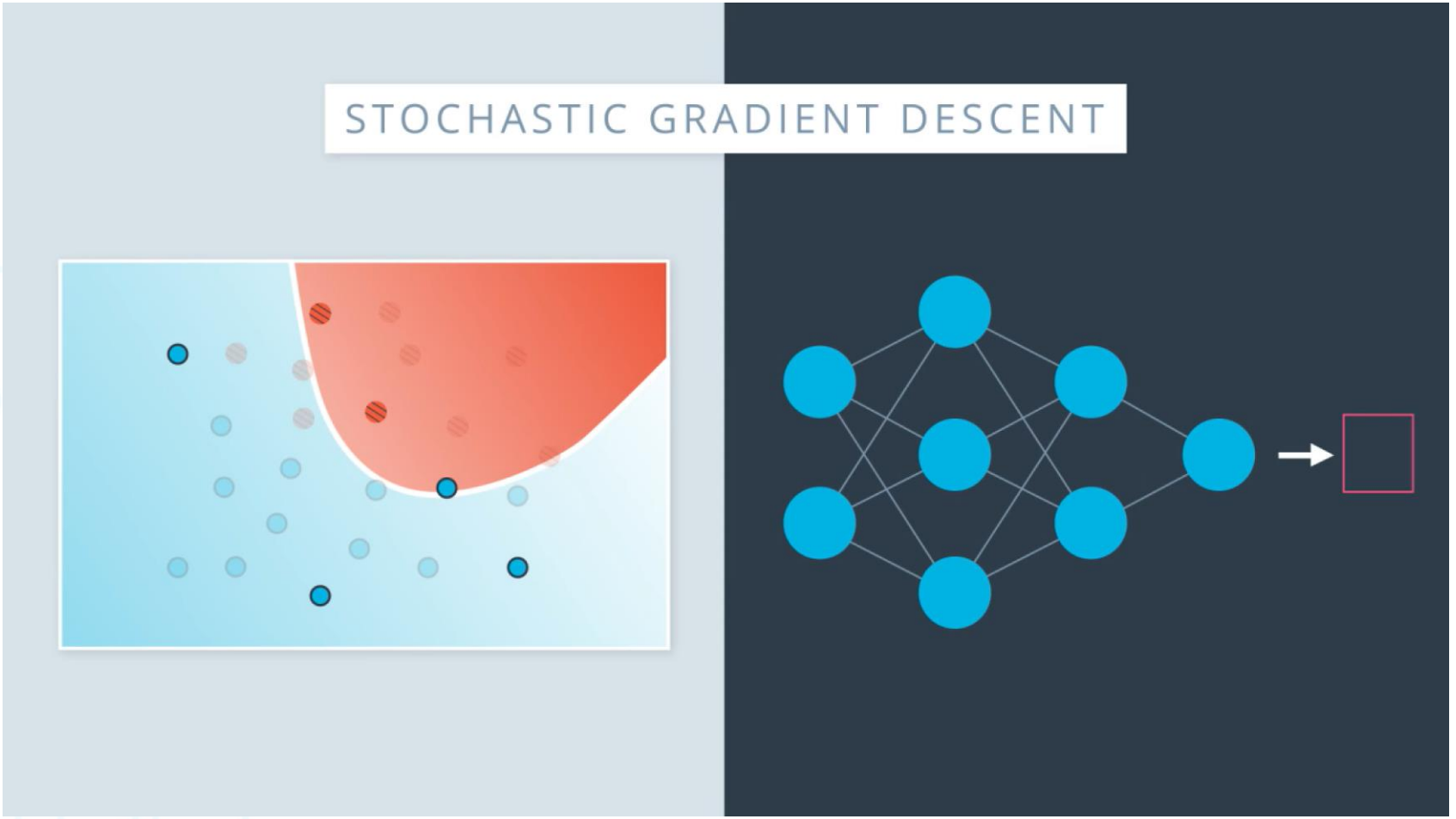
Stochastic Gradient Descent



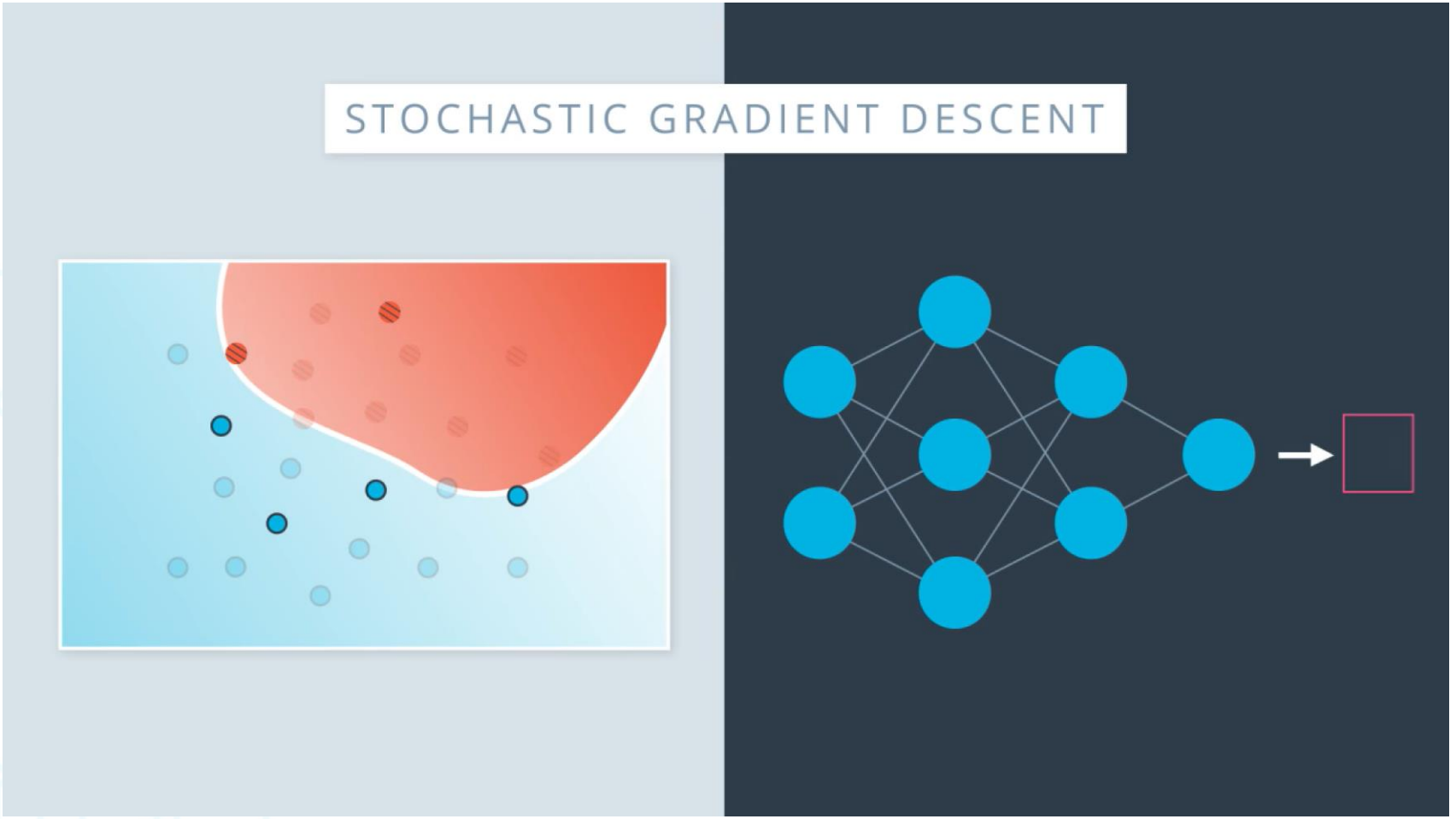
Stochastic Gradient Descent



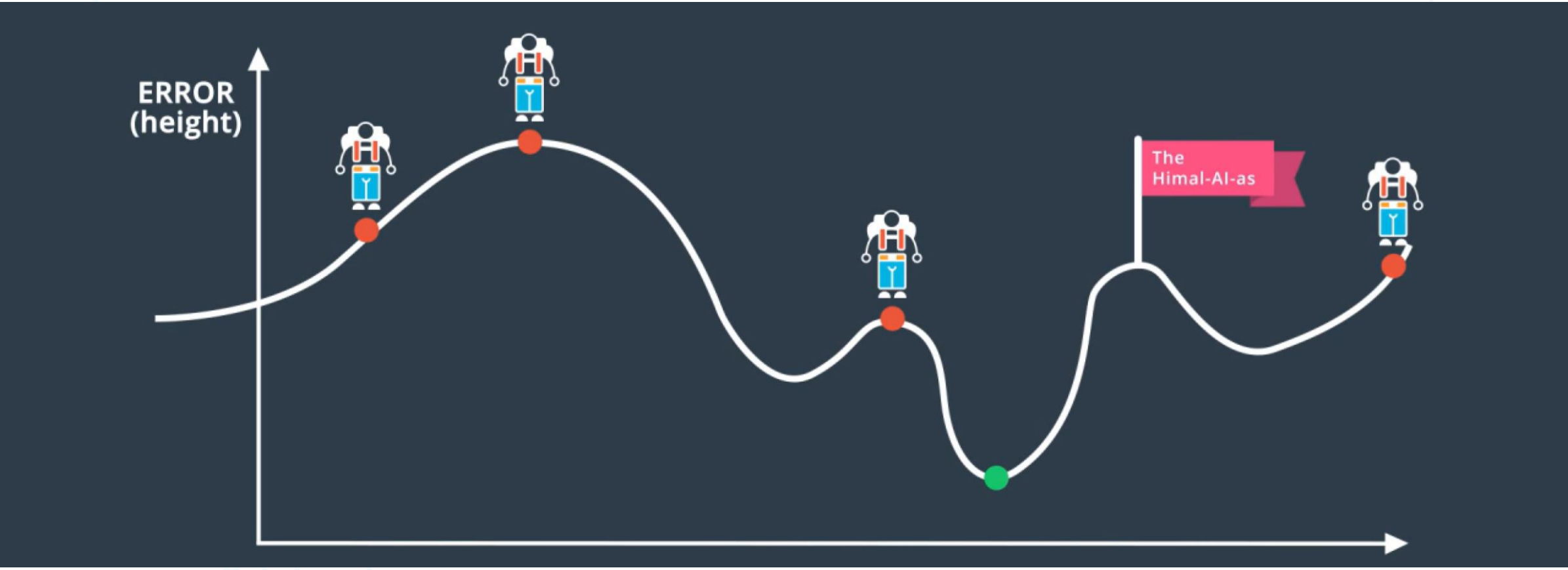
Stochastic Gradient Descent



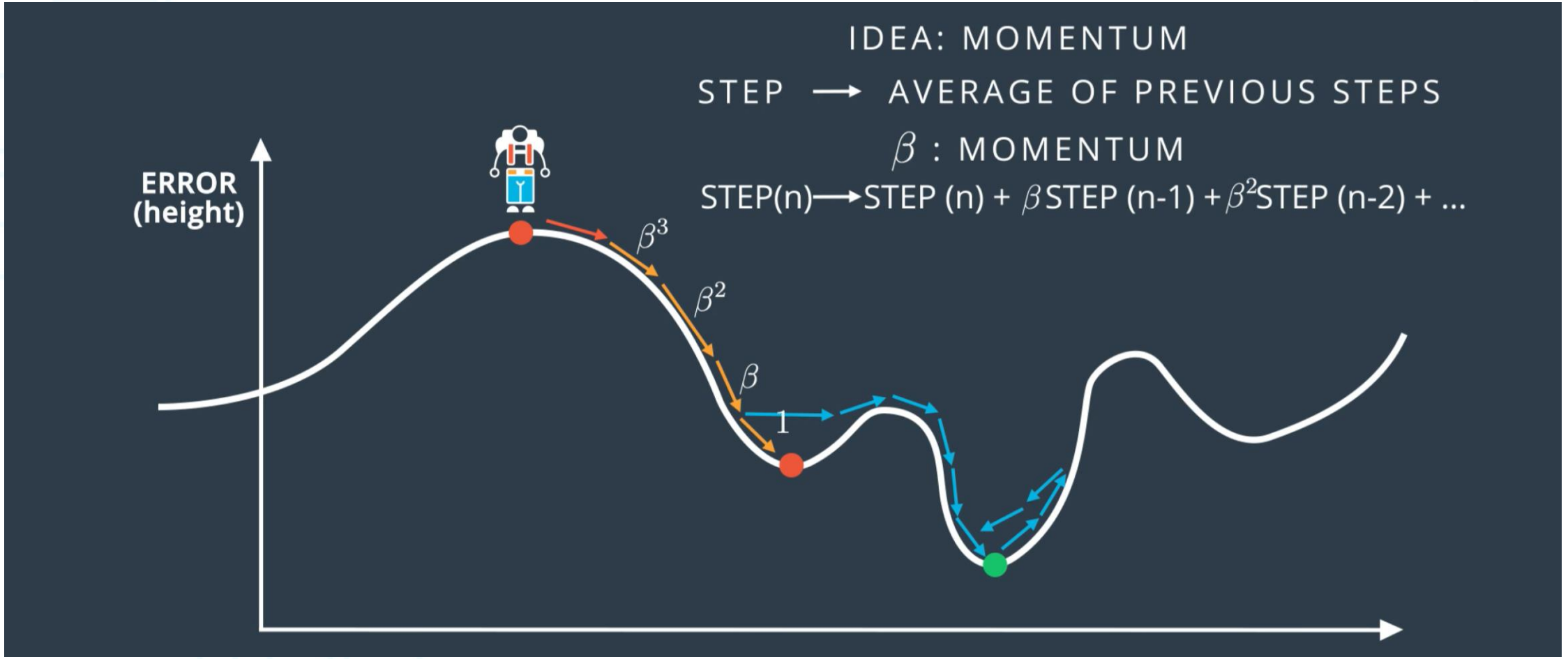
Stochastic Gradient Descent



Random Restart



Momentum


IDEA: MOMENTUM
STEP ⟶ AVERAGE OF PREVIOUS STEPS
β : MOMENTUM
STEP(n)⟶STEP (n) + βSTEP (n-1) + β²STEP (n-2) + ...
ERROR
(height)
β³
β²
β
1

Thanks !

VINICARIDÁ

Human-Centered Data & AI

Vinicius Fernandes Caridá

vfcarida@gmail.com


@vinicius caridá


@vfcarida


@vinicius caridá


@vfcarida


@vinicius caridá


@vfcarida