Master’s Degree in Intelligent Systems

Deep Learning

Manuel Piñar Molina

Universitat de les Illes Balears

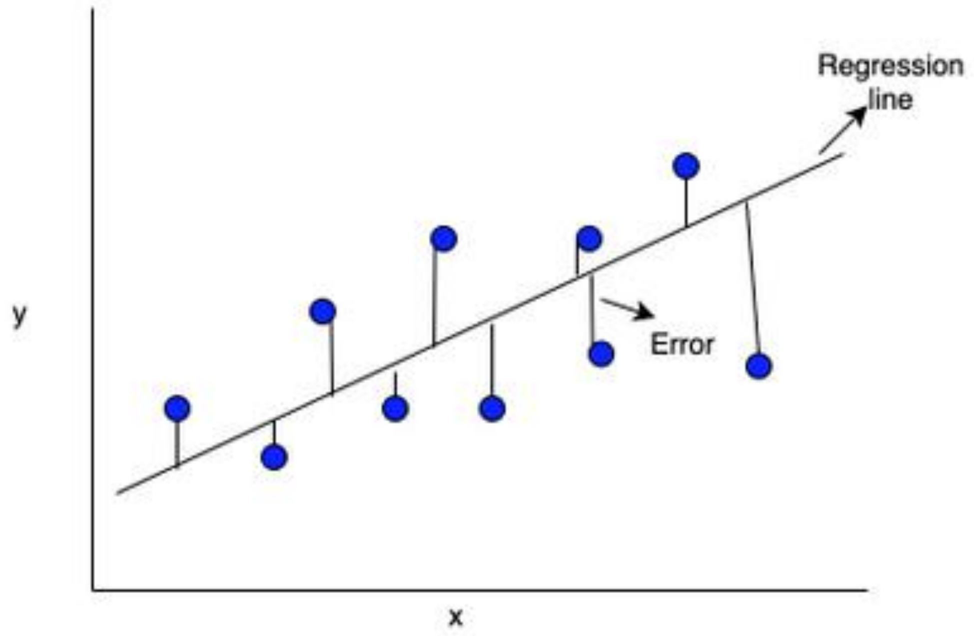
Grup de recerca de Sistemes, Robòtica i Visió

Intro to neural networks –part2

Images from udacity.com

Loss functions

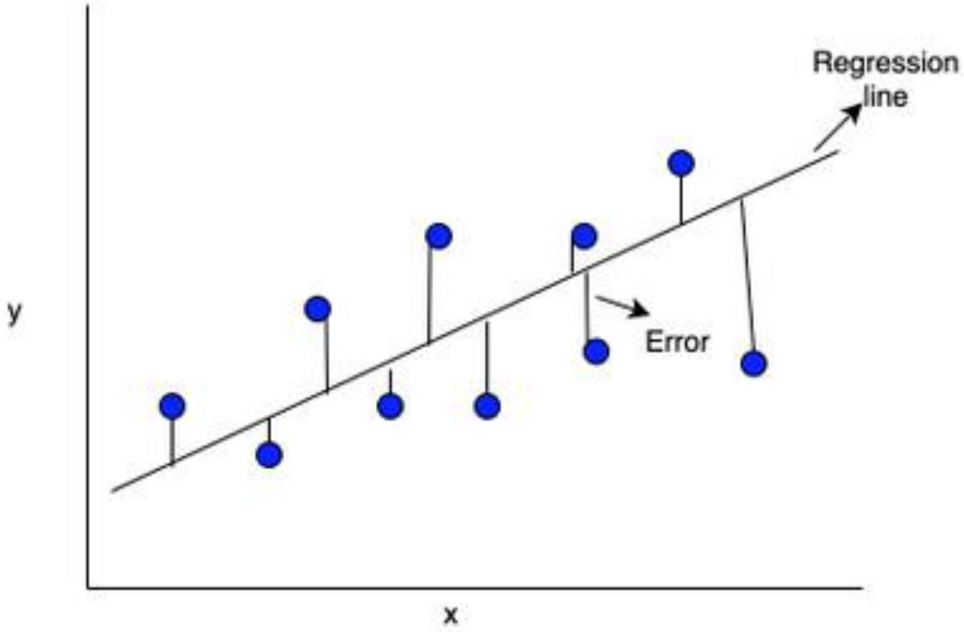
- What's the error?



The distance between our prediction or output to the correct answer.

Loss functions

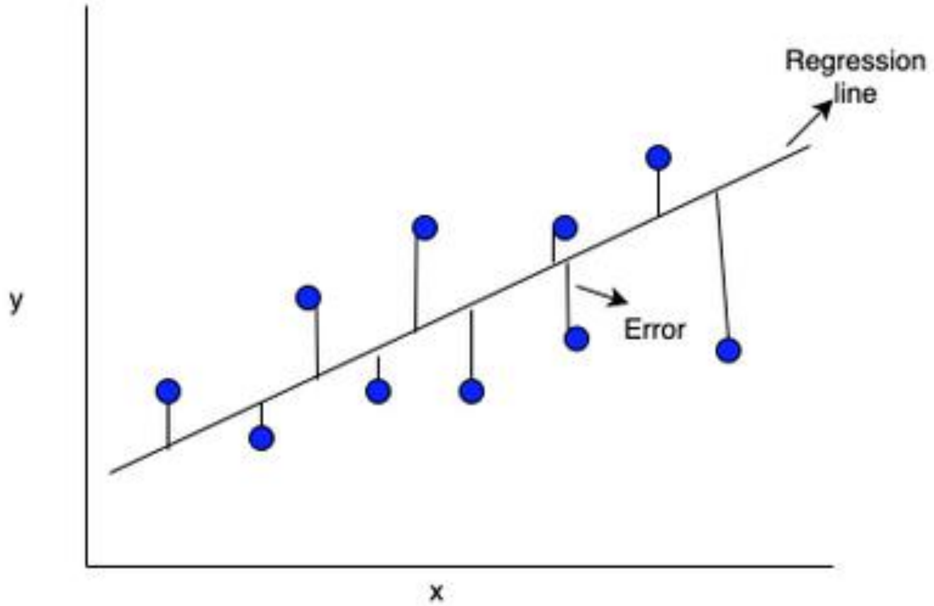
- ERROR → Could be negative?



NO!

Loss functions

- *Local error*



E = objective value – prediction

Loss functions

- ***MSE (Mean squared error)***

$$E\,(W,b) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

Being:

E = loss function

W = weight matrix.

b = bias

N = total number of predictions (outputs)

$\tilde{y}_i$ = output predicted

y_i = real output

Loss functions

- ***MAE (Mean absolute error)***

$$\mathrm{MAE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{test set}} |\underbrace{y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i}_{\text{actual value}}|$$

Being:

E = loss function

W = weight matrix.

b = bias

N = total number of predictions (outputs)

$\tilde{y}_i$ = output predicted

y_i = real output

~~**OUTLIERS**~~

Loss functions

- ***CROSS-ENTROPY***

$$E\,(W,b) = -\sum_{i=1}^{m} y_i \, log(\, p_i)$$

Being:

E = loss function

W = weight matrix.

b = bias

m = total number of predictions (outputs)

p_{i} = probability predicted

y_{i} = target probability

Loss functions

- ***BINARY CROSS-ENTROPY***

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Being:

H(q) = loss function

y_i = real output

N= total number of data

$p(y_i)$ = probability that the data y_i is 1

$1 - p(y_i)$ = probability that the data y_i is 0

Loss functions

- ***BINARY CROSS-ENTROPY***

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

When the expected output is 1 :

$$y_i \cdot log(p(y_i))$$

When the expected output is 0 :

$$(1 - y_i) \cdot log(1 - p(y_i))$$

Loss functions

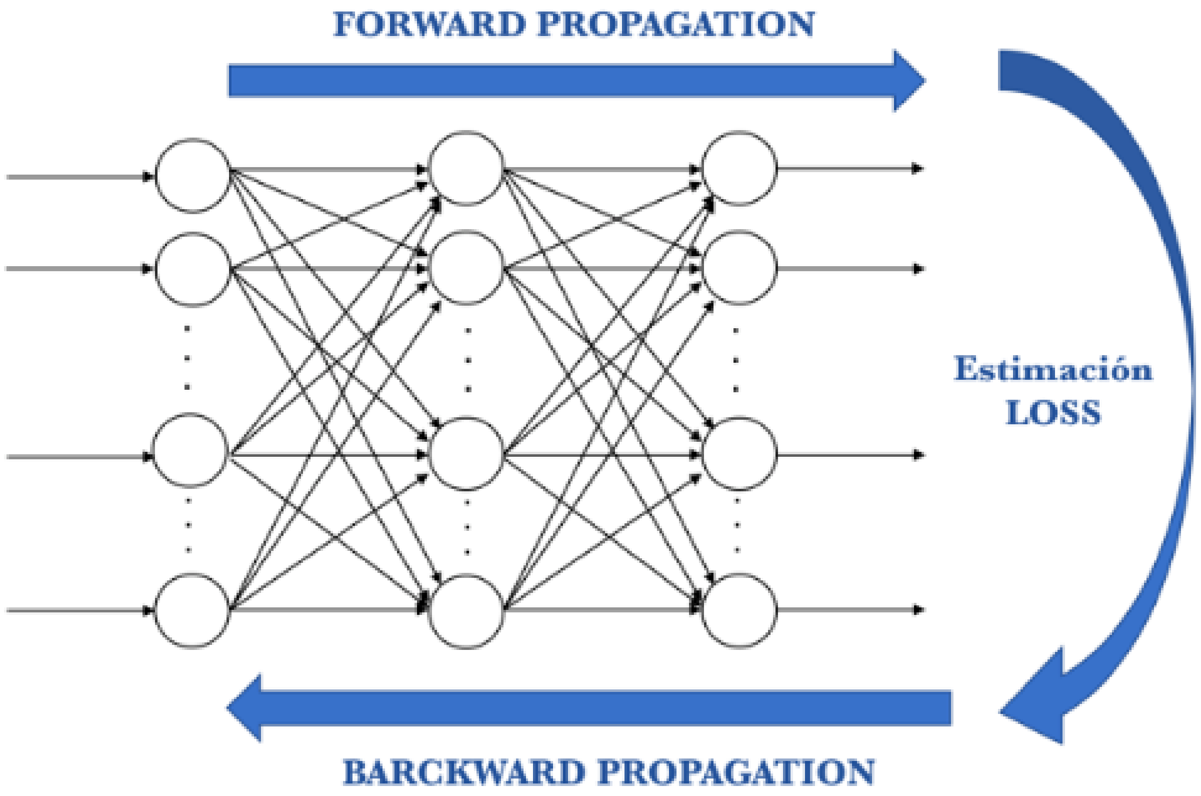
We are working on a neural network that will be able to classify images of snakes, fish and bears.
Input image:

Loss function??

$$E\,(W,b) = -\sum_{i=1}^{m} y_i \, log(\, p_i)$$

Train neural networks

We call training the procedure used to carry out the learning process in a neural network:

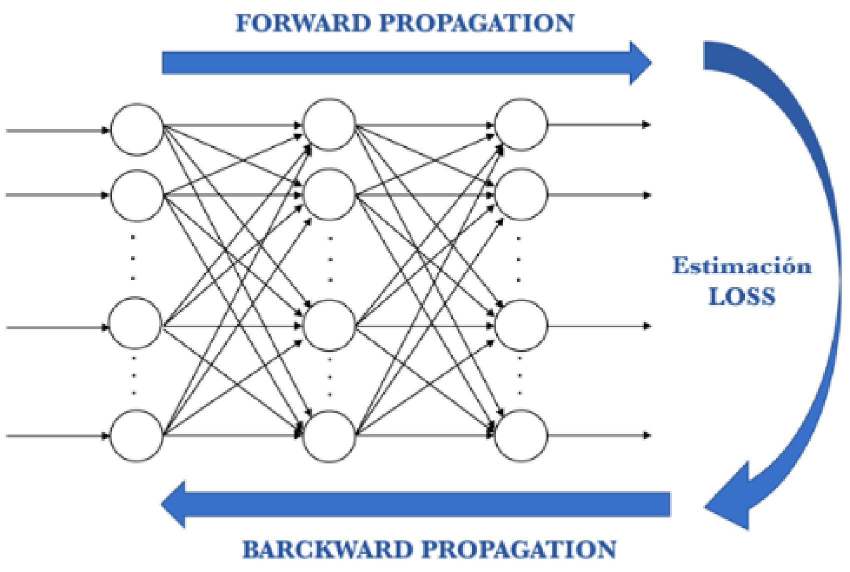


WEB UPC

Training process

Training steps:

1) Set random weights values, hyperparameters…
2) Take input values and make passforward
3) Calculate loss function
4) Make the backpropagation with error information
5) Modify weights and bias with backpropagation info
6) Go to step 2) until error 0 or below threshold



WEB UPC

Optimizations algorithms

Types of optimizers:

1) ADADELTA: An Adaptive Learning Rate Method
https://arxiv.org/abs/1212.5701

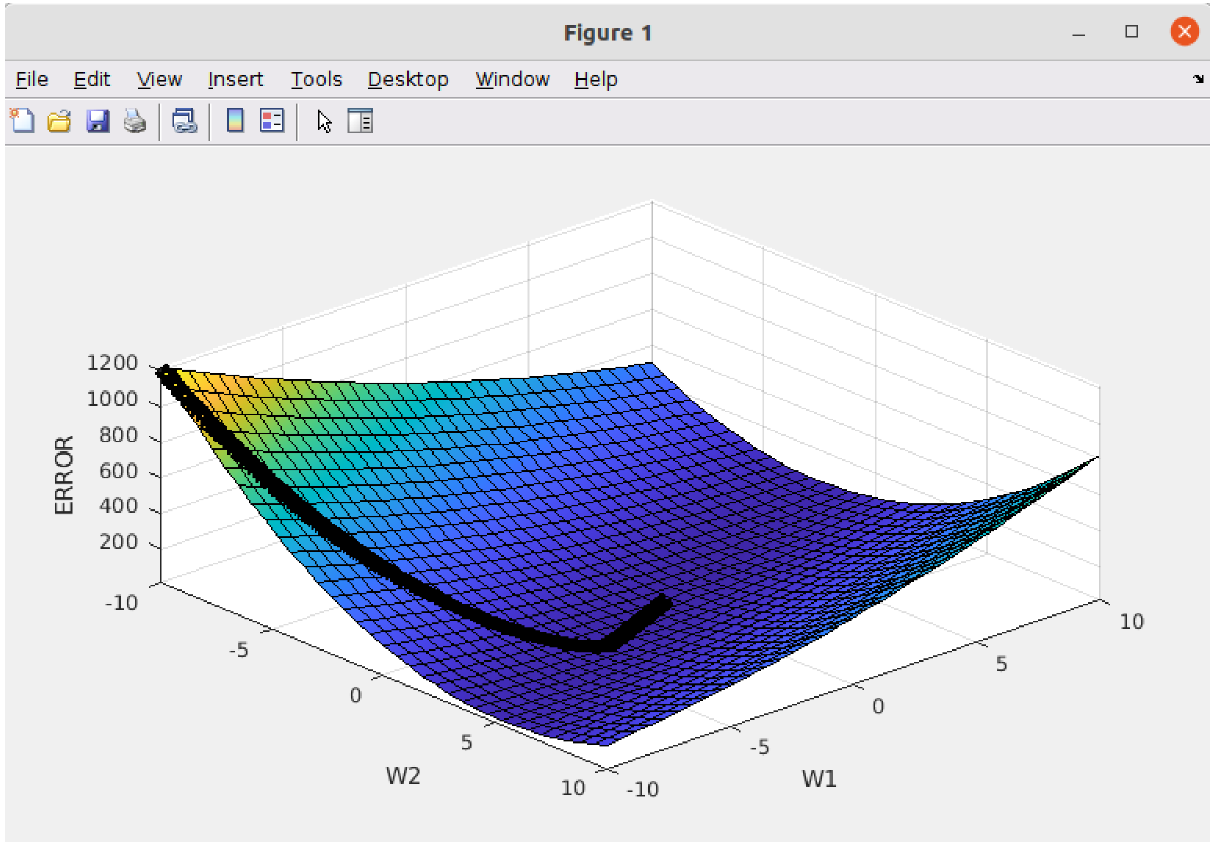
2) ADAGRADIENT: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization

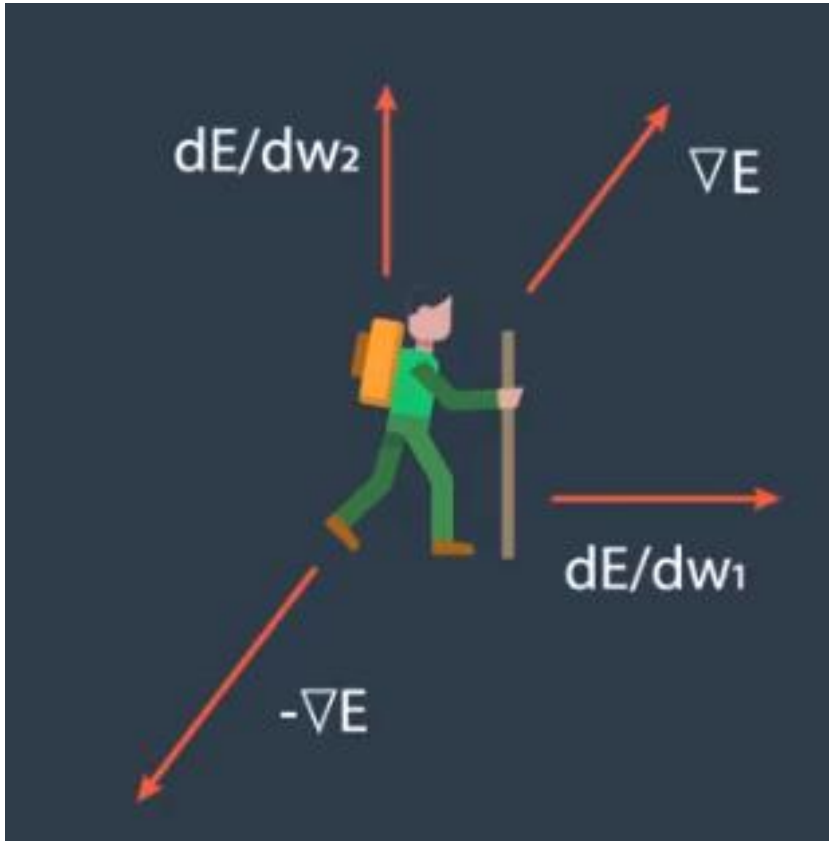
https://jmlr.org/papers/v12/duchi11a.html

3) Adam: A Method for Stochastic Optimization
https://arxiv.org/abs/1412.6980

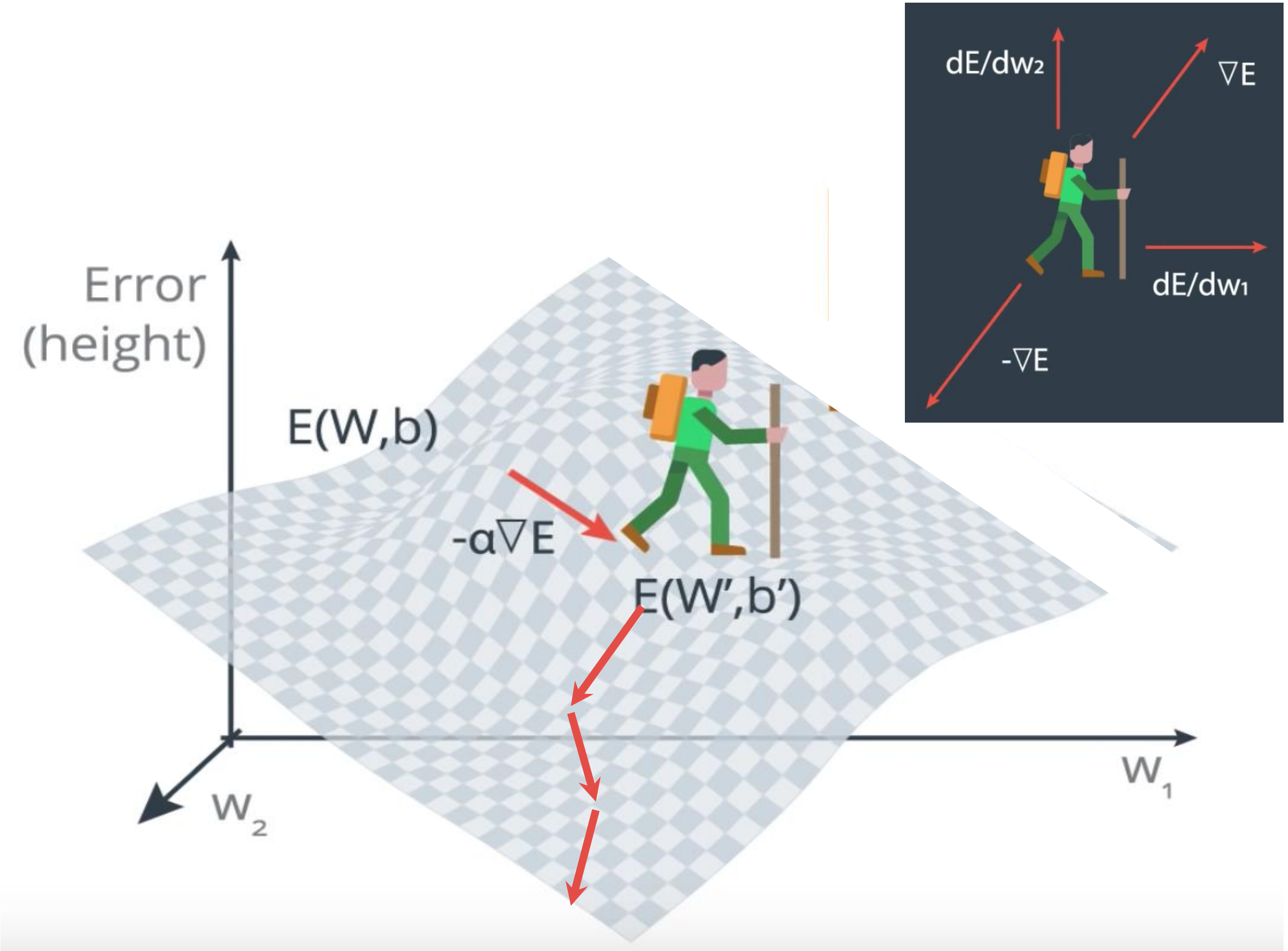
4) Stochastic **gradient descent**

Gradient Descent





Gradient Descent



Udacity.com

$\hat{y} = \sigma(Wx+b) \longleftarrow$ Bad

$\hat{y} = \sigma(w_1x_1 + \ldots + w_nx_n + b)$

$\nabla E = (\partial E/\partial w_1, \ldots, \partial E/\partial w_n, \partial E/\partial b)$

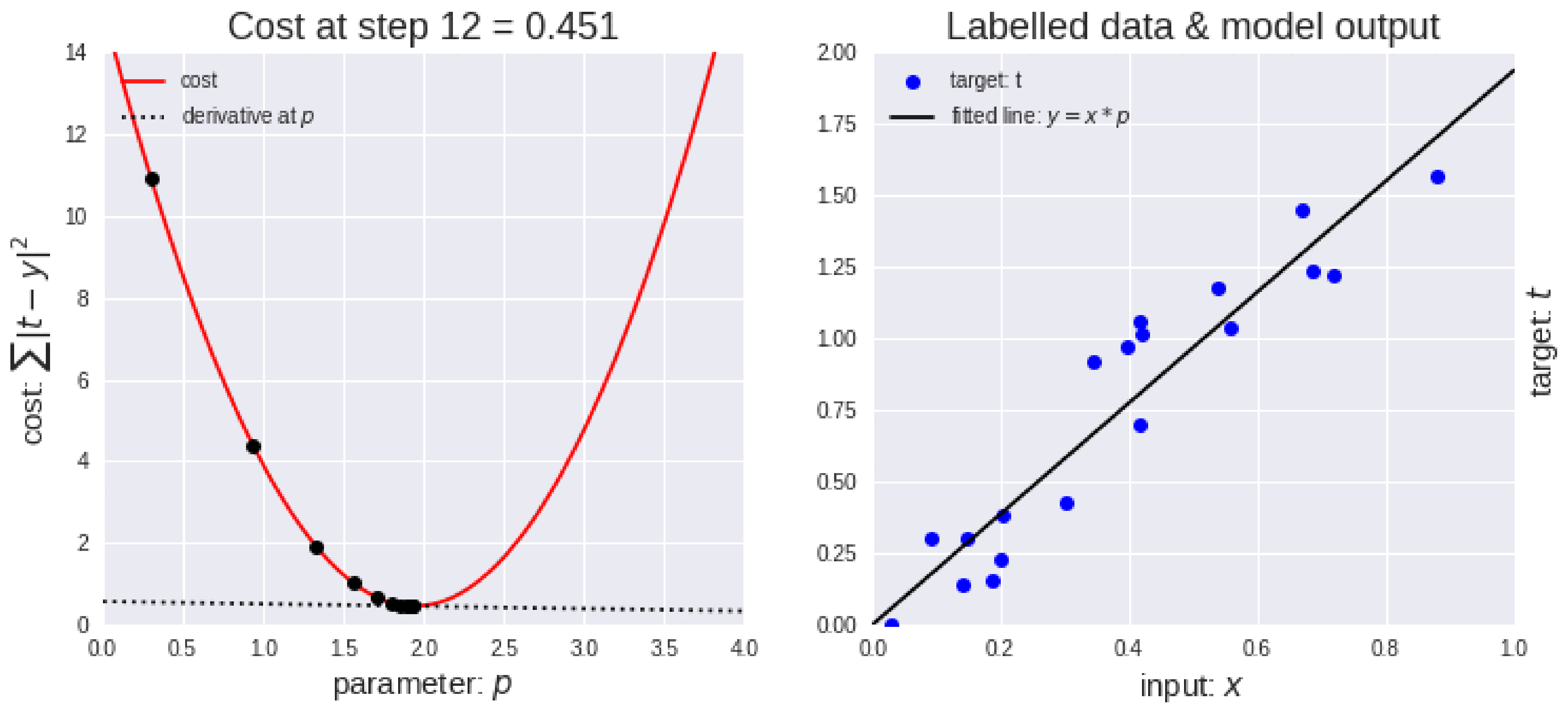
$\alpha = 0.1$ (learning rate)

$w_i' \longleftarrow w_i - \alpha\, \partial E/\partial w_i$

$b' \longleftarrow b - \alpha\, \partial E/\partial b$

$\hat{y} = \sigma(W'x+b') \longleftarrow$ Better

Gradient Descent



Maths under the algorithm

The total error E (for m points) is:

$$E = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i)\right)$$ where the prediction is given by $\hat{y}_i = \sigma(Wx^{(i)} + b)$

The error produced by each point is $E = -y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$

The goal is to calculate the gradient of the total error E, at a point x = (x1, …, xn)

$$\nabla E = \left(\frac{\partial}{\partial w_1}E, \cdots, \frac{\partial}{\partial w_n}E, \frac{\partial}{\partial b}E\right)$$

To simplify the calculations, the derivative of the error of each point is calculated and the total error, then, is the average of the errors at all the points.

$$\nabla E = -(y-\hat{y})(x_1, \ldots, x_n, 1)$$

Maths under the algorithm

First we must calculate the first order derivative of sigmoid function

$$
\begin{aligned}
\sigma'(x) &= \frac{\partial}{\partial x}\frac{1}{1+e^{-x}} \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \sigma(x)(1-\sigma(x))
\end{aligned}
$$

Goal $\Rightarrow$ $\nabla E = \left(\frac{\partial}{\partial w_1}E, \cdots, \frac{\partial}{\partial w_n}E, \frac{\partial}{\partial b}E\right)$

Maths under the algorithm

Let's calculate each term $\frac{\partial}{\partial w_j} E$

Knowing that $E = -y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$

Thus
$$\begin{aligned}
\frac{\partial}{\partial w_j}\hat{y} &= \frac{\partial}{\partial w_j}\sigma(Wx+b) \\
&= \sigma(Wx+b)(1-\sigma(Wx+b)) \cdot \frac{\partial}{\partial w_j}(Wx+b) \\
&= \hat{y}(1-\hat{y}) \cdot \frac{\partial}{\partial w_j}(Wx+b) \\
&= \hat{y}(1-\hat{y}) \cdot \frac{\partial}{\partial w_j}(w_1x_1 + \cdots + w_jx_j + \cdots + w_nx_n + b) \\
&= \hat{y}(1-\hat{y}) \cdot x_j
\end{aligned}$$

As we can see before the last equality, the only term that is not a constant is w_jx_j

So that partial derivative will be x_j

Maths under the algorithm

From here we can calculate,

$$
\begin{aligned}
\frac{\partial}{\partial w_j} E &= \frac{\partial}{\partial w_j}[-y \log(\hat{y}) - (1-y)\log(1-\hat{y})] \\
&= -y \frac{\partial}{\partial w_j} \log(\hat{y}) - (1-y) \frac{\partial}{\partial w_j} \log(1-\hat{y}) \\
&= -y \cdot \frac{1}{\hat{y}} \cdot \frac{\partial}{\partial w_j}\hat{y} - (1-y) \cdot \frac{1}{1-\hat{y}} \cdot \frac{\partial}{\partial w_j}(1-\hat{y}) \\
&= -y \cdot \frac{1}{\hat{y}} \cdot \hat{y}(1-\hat{y})x_j - (1-y) \cdot \frac{1}{1-\hat{y}} \cdot (-1)\hat{y}(1-\hat{y})x_j \\
&= -y(1-\hat{y}) \cdot x_j + (1-y)\hat{y} \cdot x_j \\
&= -(y-\hat{y})x_j
\end{aligned}
$$

Maths under the algorithm

If we calculate in the same way but with respect to *b* we will obtain,

$$\frac{\partial}{\partial b} E = -(y - \hat{y})$$

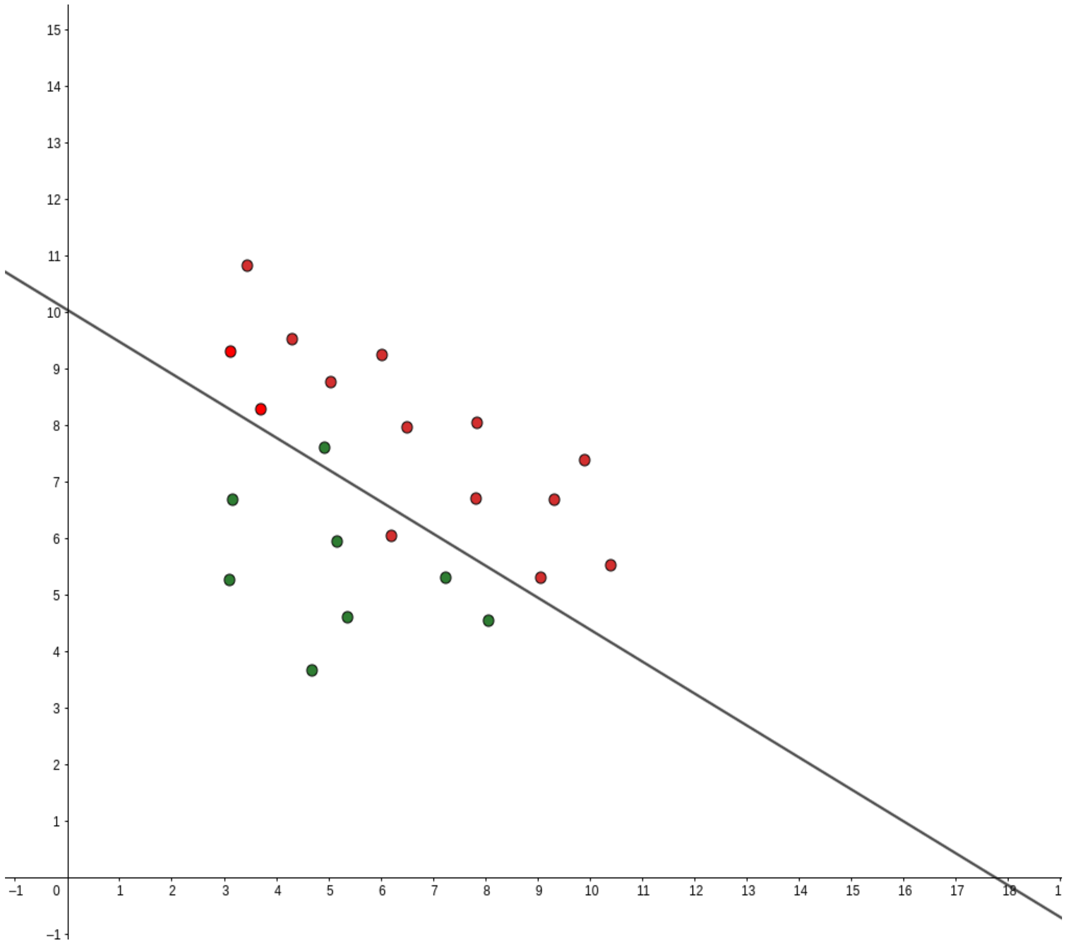
Hence for a set of points, $(x_1, \ldots, x_n)$

The gradient will be:

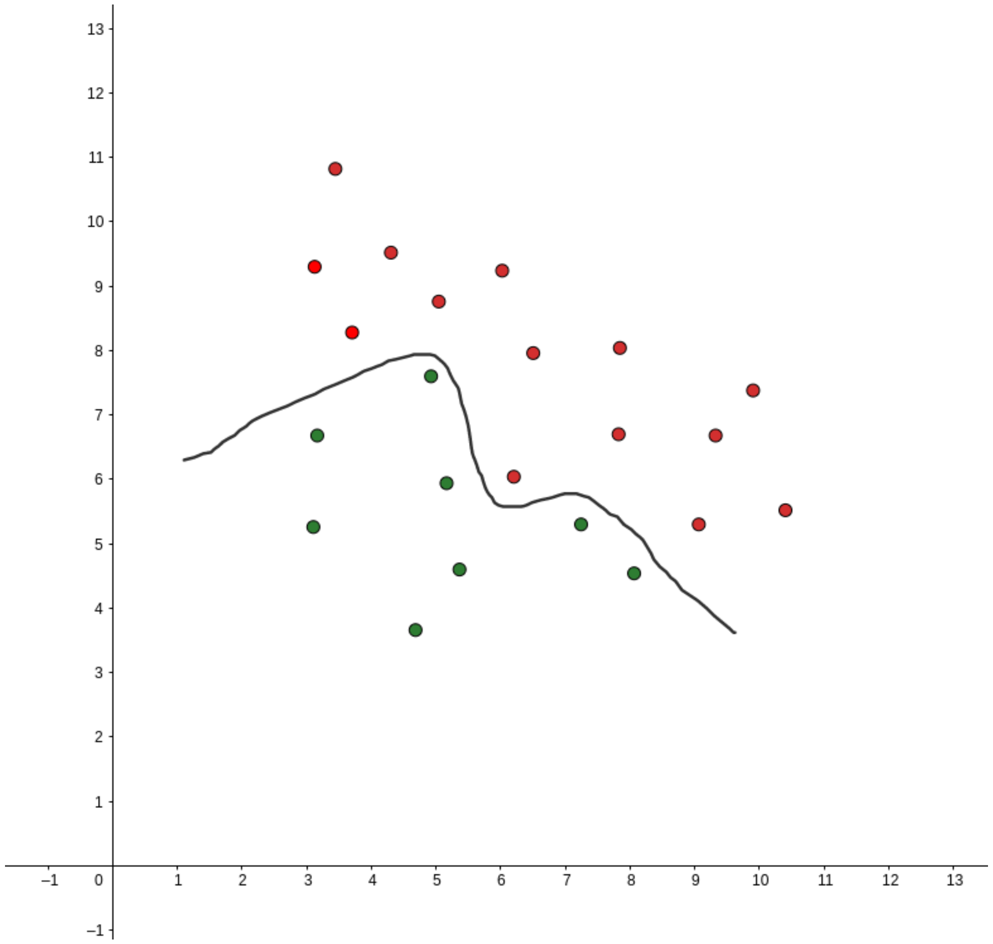
$$(-(y - \hat{y})x_1, \cdots, -(y - \hat{y})x_n, -(y - \hat{y}))$$

$$\nabla E = -(y - \hat{y})(x_1, \ldots, x_n, 1)$$

TEST

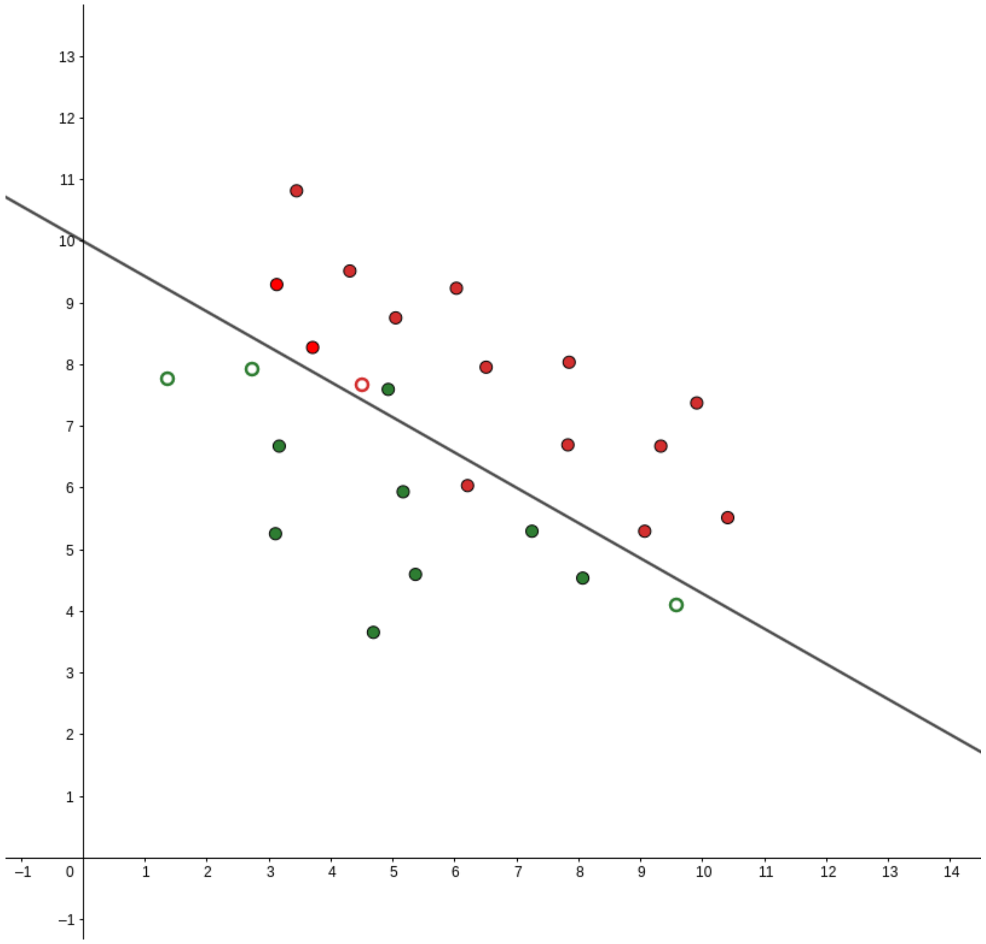


15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
−1
−1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

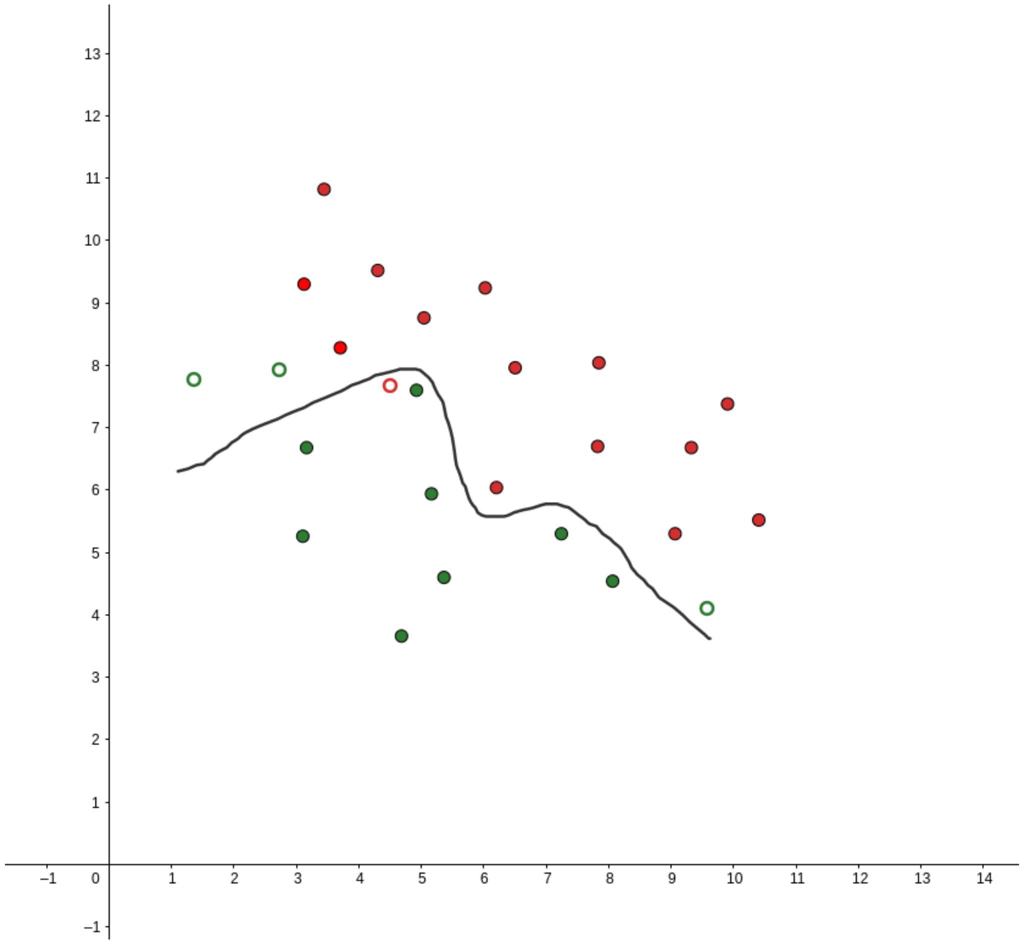


13
12
11
10
9
8
7
6
5
4
3
2
1
−1
−1 0 1 2 3 4 5 6 7 8 9 10 11 12 13

TEST

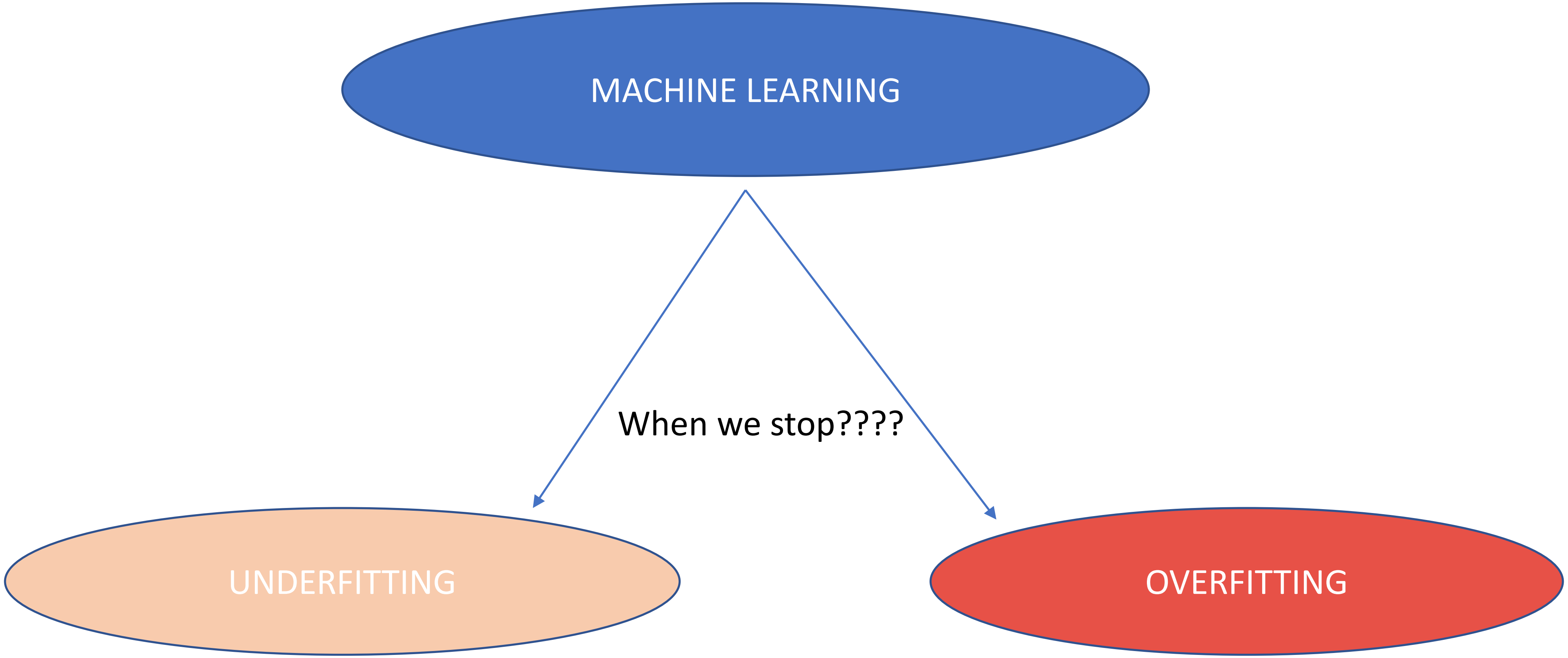


13
12
11
10
9
8
7
6
5
4
3
2
1
0
−1
−1 1 2 3 4 5 6 7 8 9 10 11 12 13 14

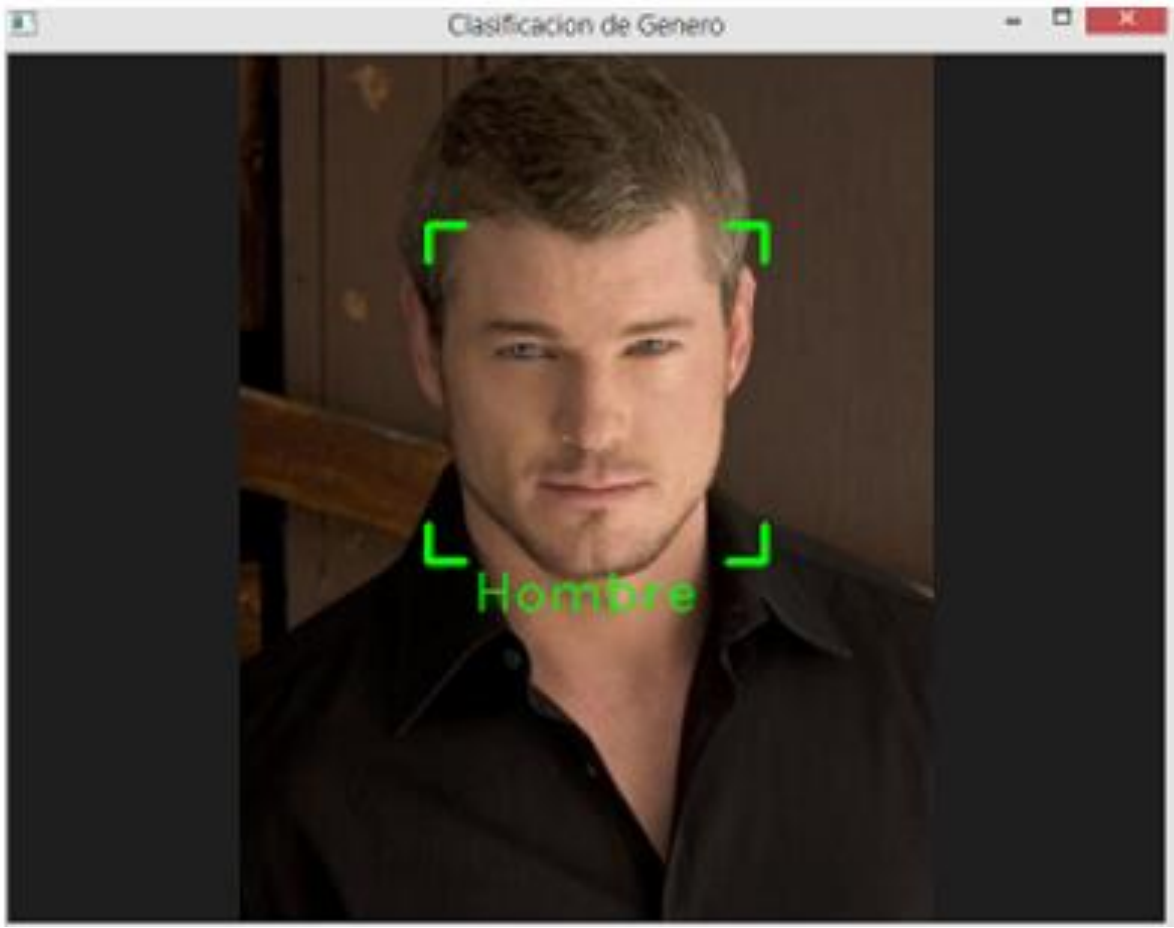


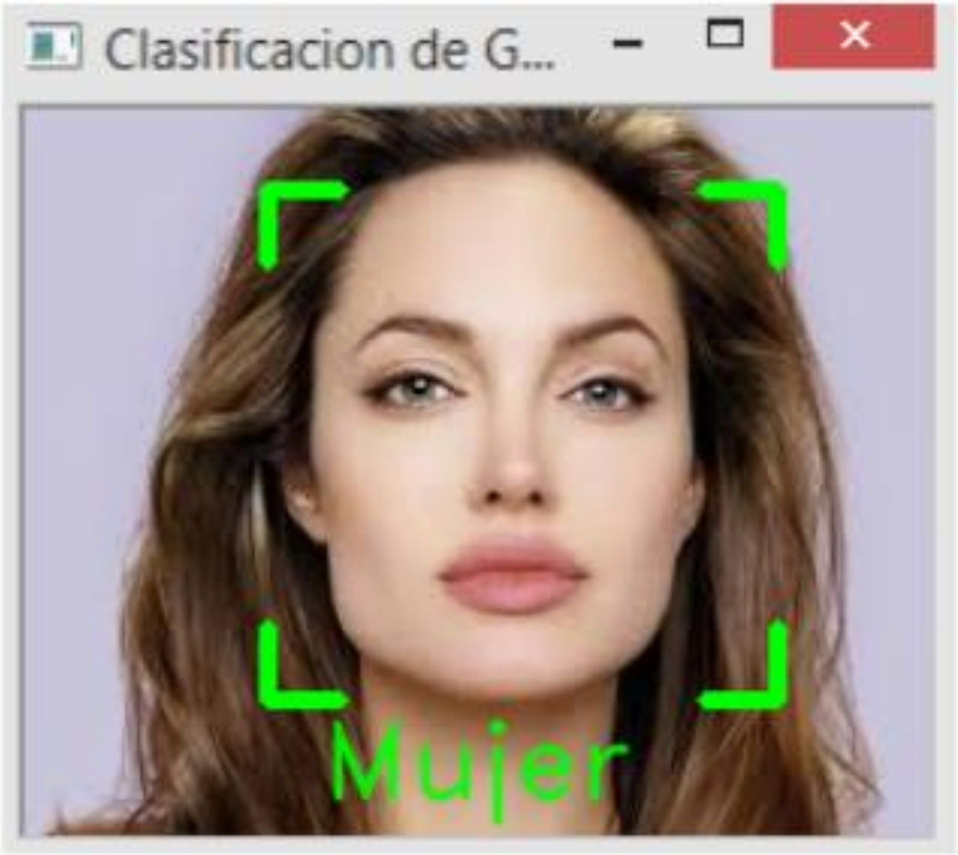
13
12
11
10
9
8
7
6
5
4
3
2
1
0
−1
−1 1 2 3 4 5 6 7 8 9 10 11 12 13 14

UNDERFITTING vs OVERFITTING



UNDERFITTING vs OVERFITTING





UNDERFITTING vs OVERFITTING

- Two different models:

1st) MAN → SHORT HAIR / BLUE EYES / NOT EARRINGS/ BLACK SHIRT

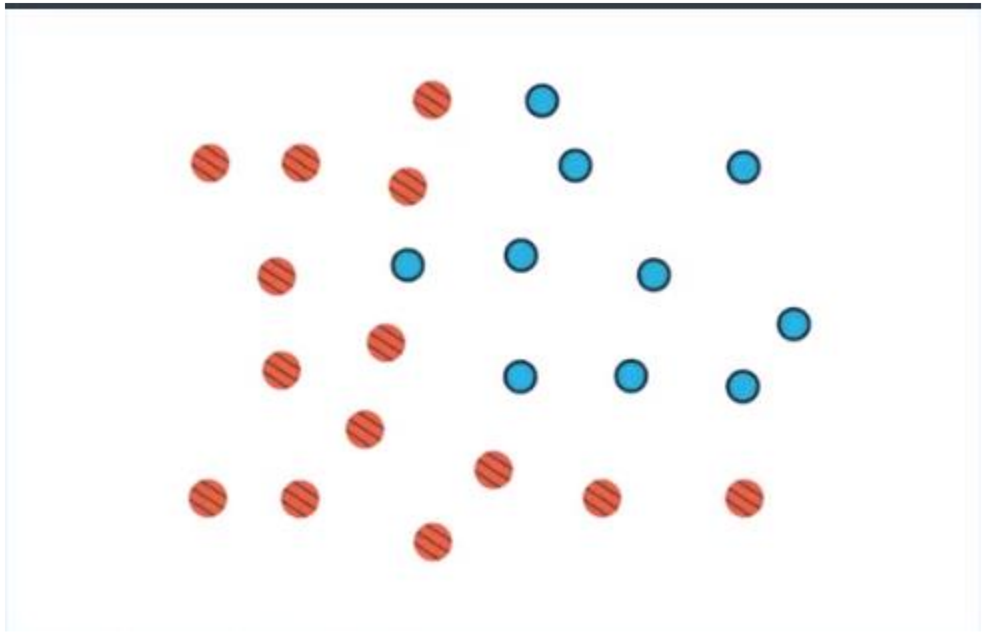
FIRST IS TOO SPECIFIC

2nd) WOMAN → LONG HAIR

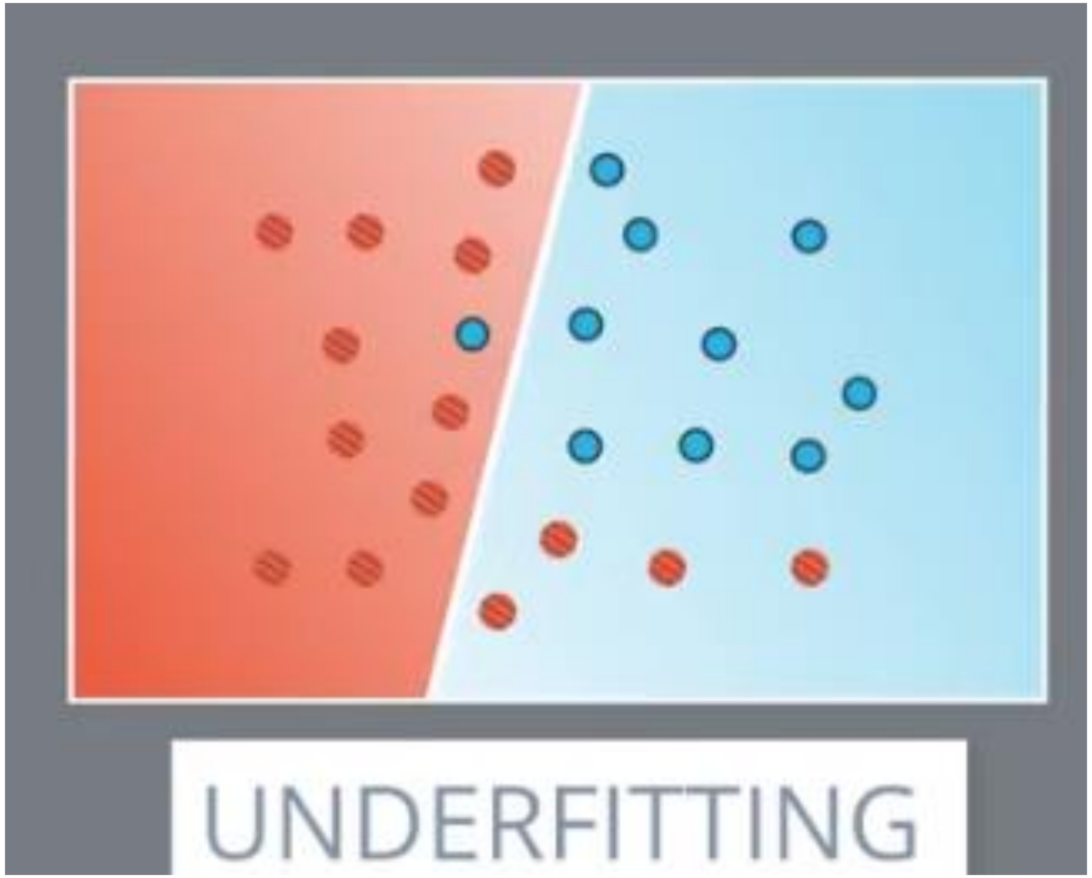
SECOND IS TOO SIMPLE

UNDERFITTING vs OVERFITTING

LET’S EXPLAIN WITH OTHER EXAMPLE

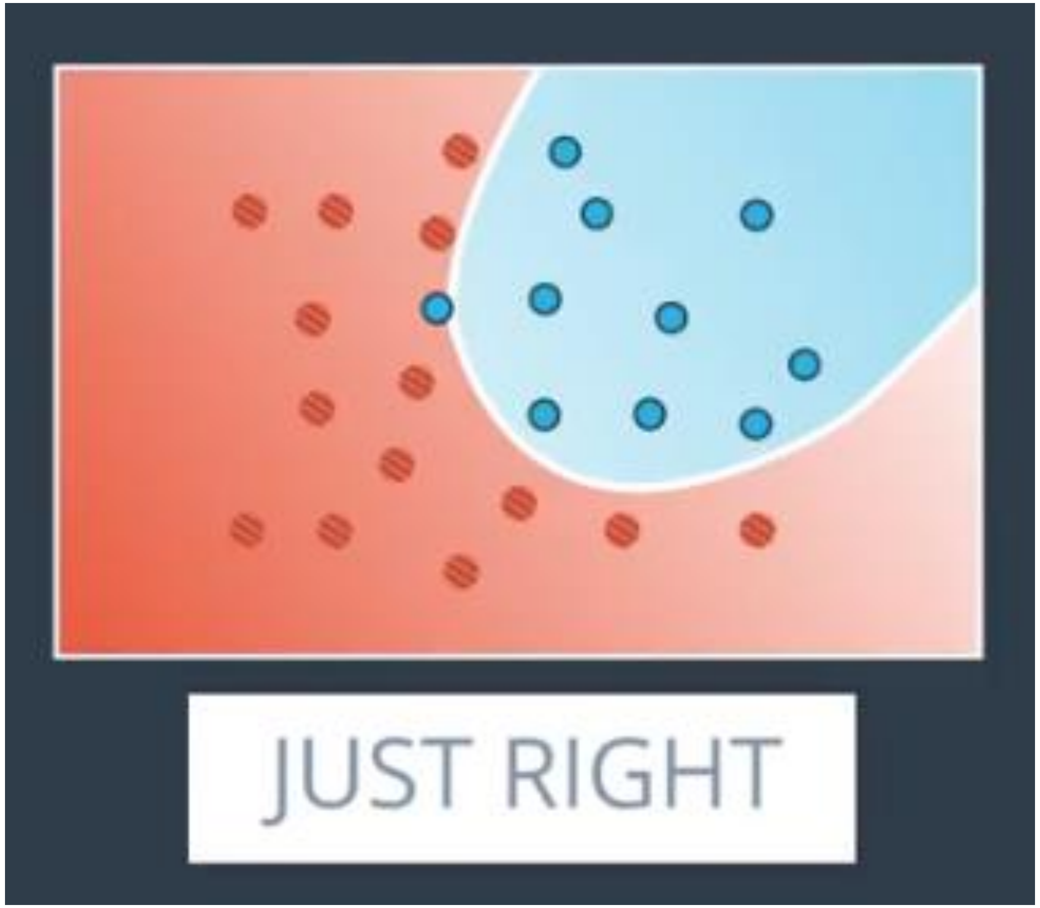


UNDERFITTING vs OVERFITTING



UNDERFITTING vs OVERFITTING

UNDERFITTING vs OVERFITTING



EARLY STOPPING

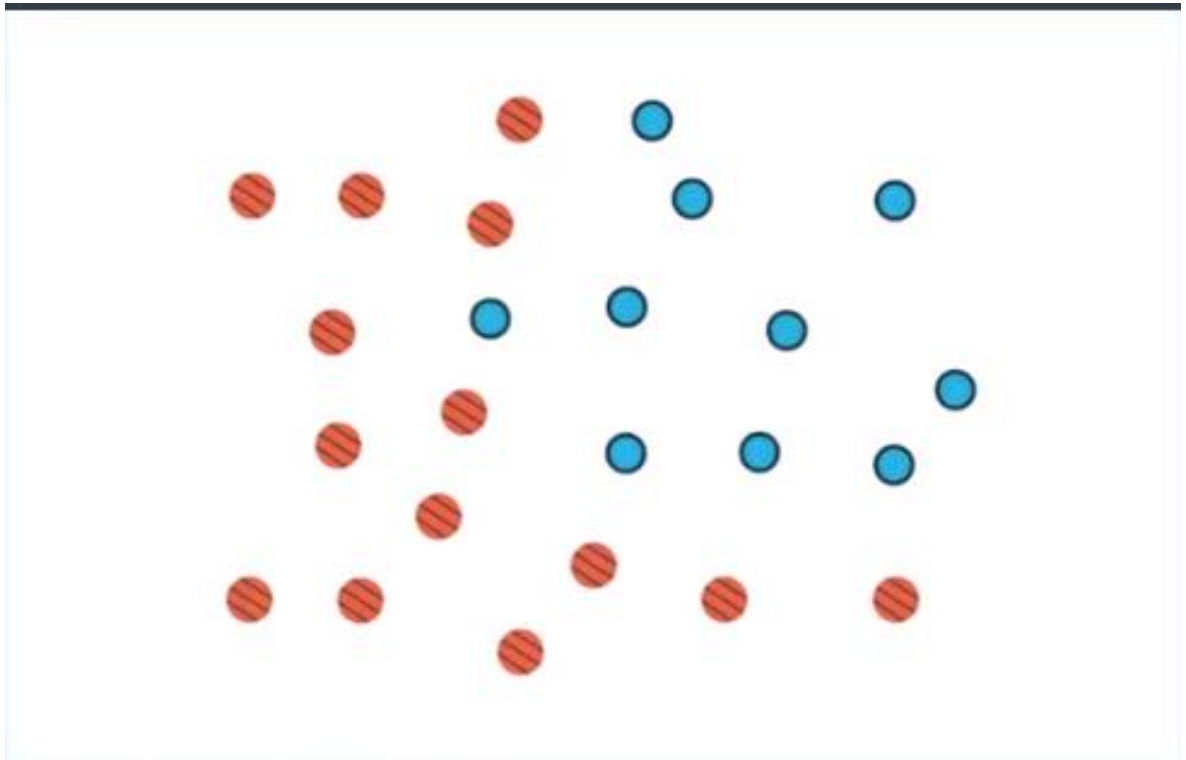
WHEN SHOULD WE STOP TRAINING ???

FEW EPOCHS ???

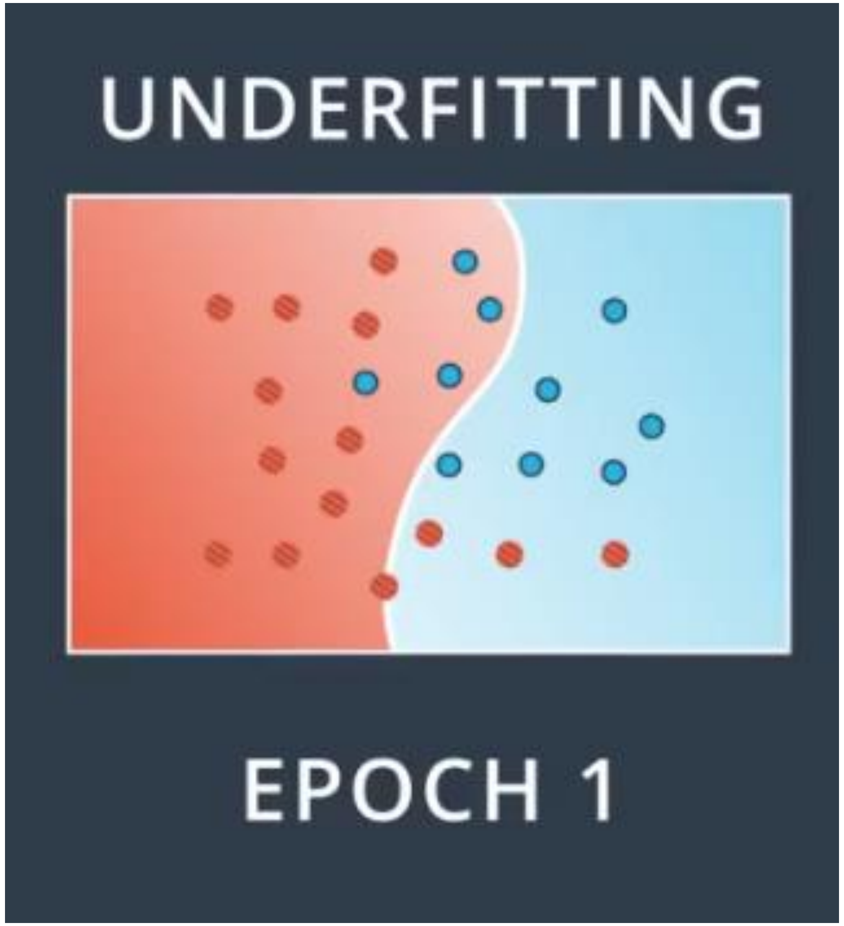
MANY EPOCHS ???

HOW MANY???

EARLY STOPPING



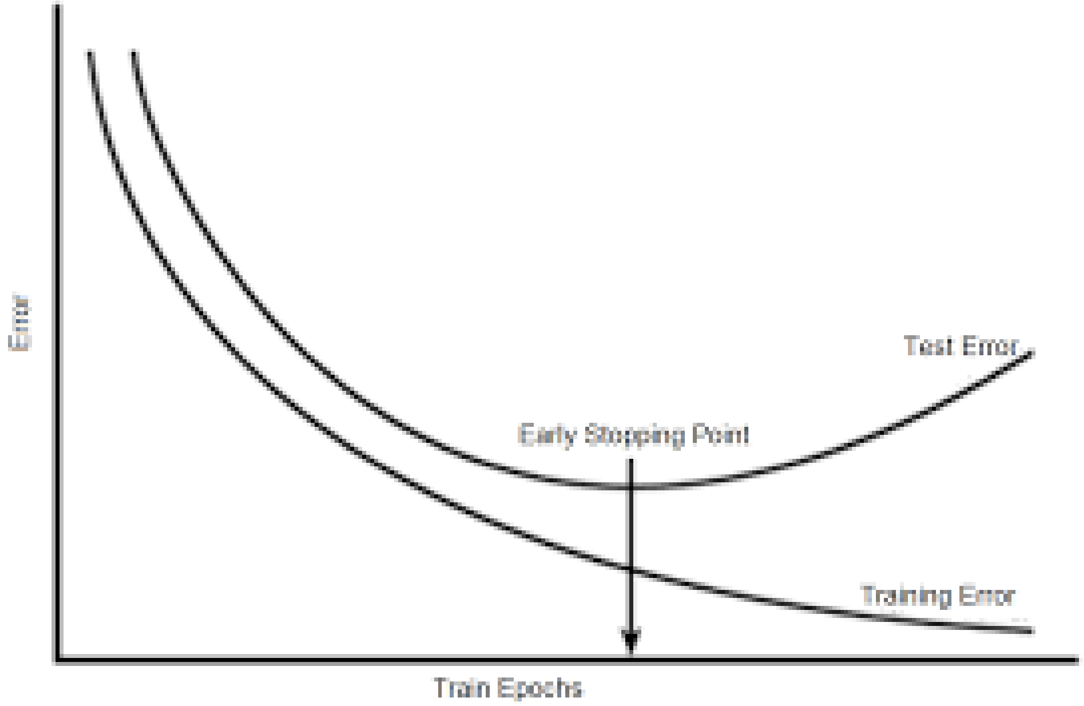
EARLY STOPPING



EARLY STOPPING

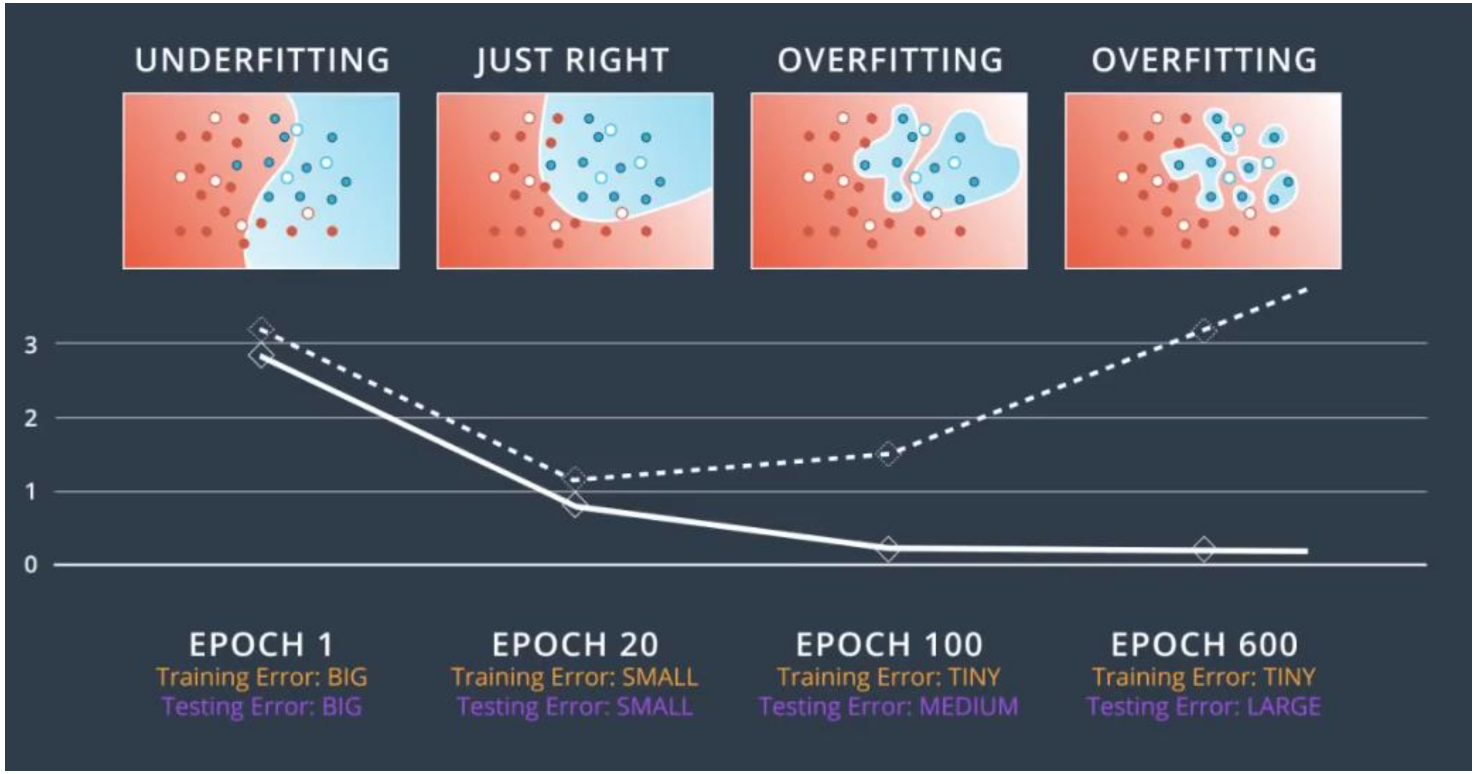
EARLY STOPPING

When do we stop training? How many iterations do we use in our training?



EARLY STOPPING

When do we stop training? How many iterations do we use in our training?



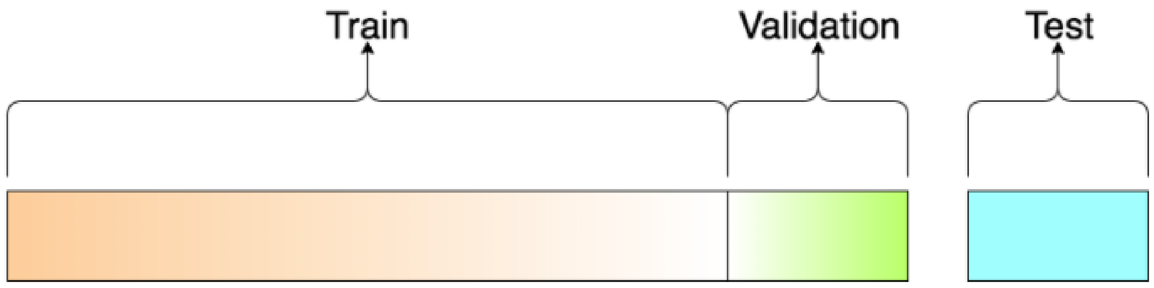
EARLY STOPPING

If we focus both on the error on the training data and on the test or validation data of the model we can observe:

1) With very few epochs the model does not work with either training data or test data. Both errors are high.
2) If we train more times than necessary, even though the error in the training data will be low, we will see how the error in the test data increases.
3) JUST AT THE POINT BEFORE THE ERROR IN THE TEST DATA STARTS TO INCREASE WILL BE WHERE WE MAKE THE EARLY STOPPING OF THE TRAINING.

EARLY STOPPING

How can we distribute the data?



A visualization of the splits

DROPOUT

Why dropout?

DROPOUT

Why dropout?

DROPOUT

How avoid?

Epoch 1

DROPOUT

How avoid?

Epoch 2

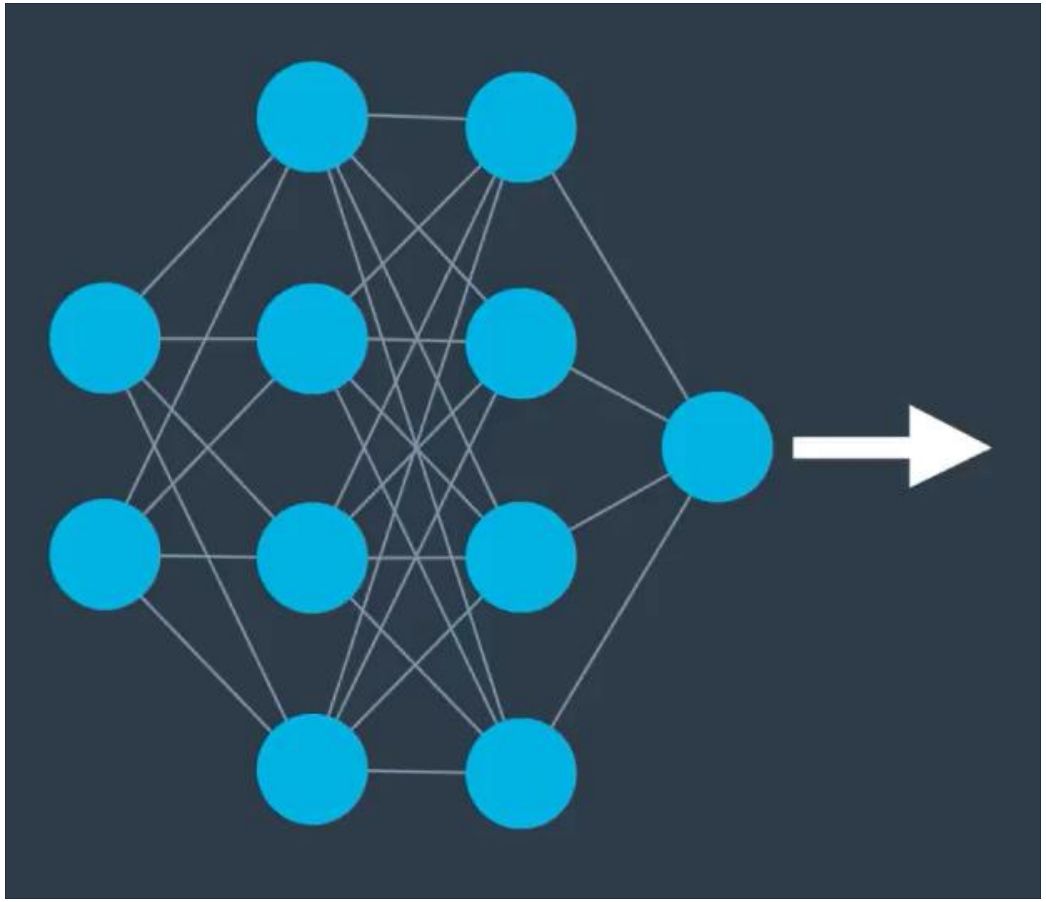
DROPOUT

GOAL?

Epoch X

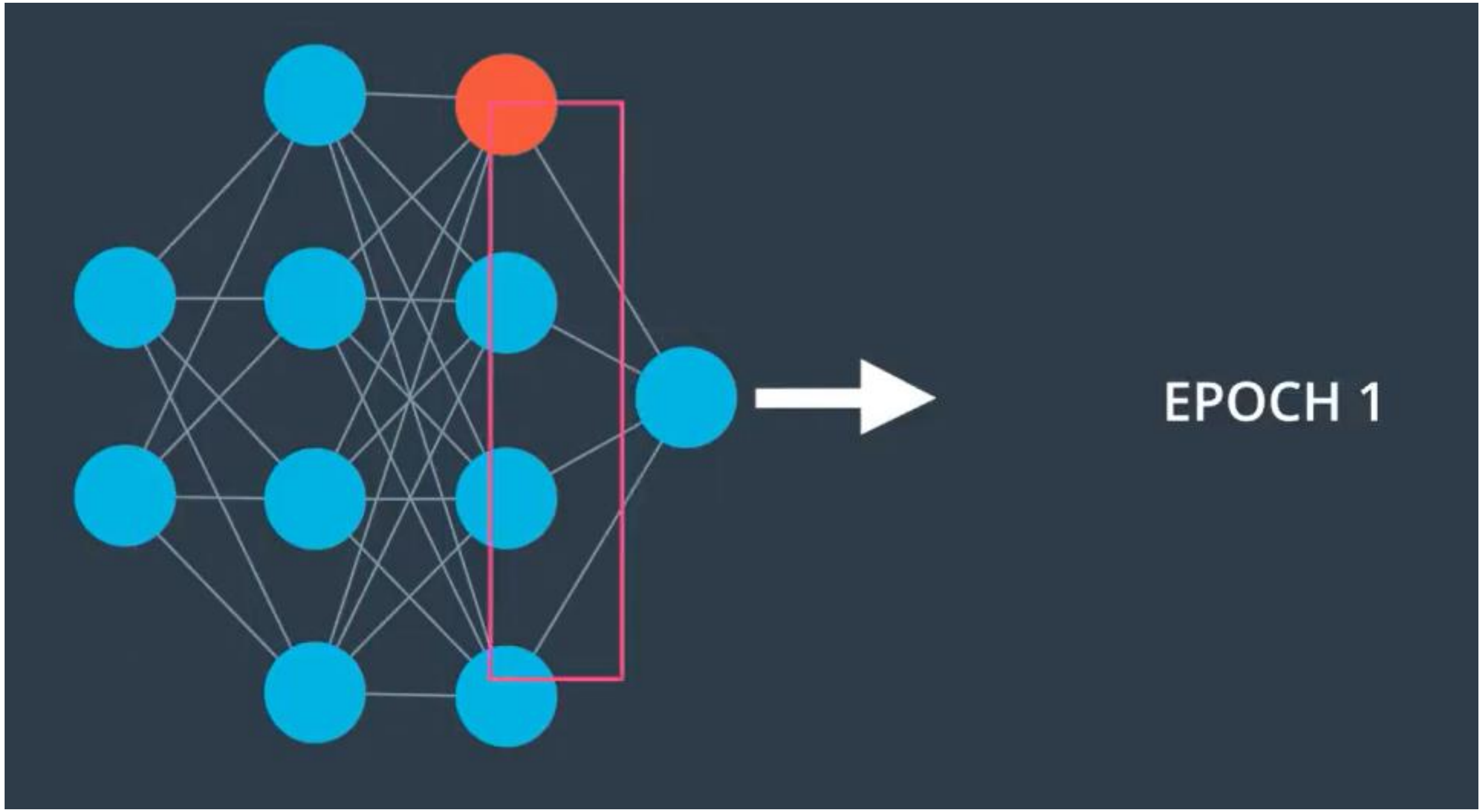
DROPOUT

IN NEURAL NETWORKS!



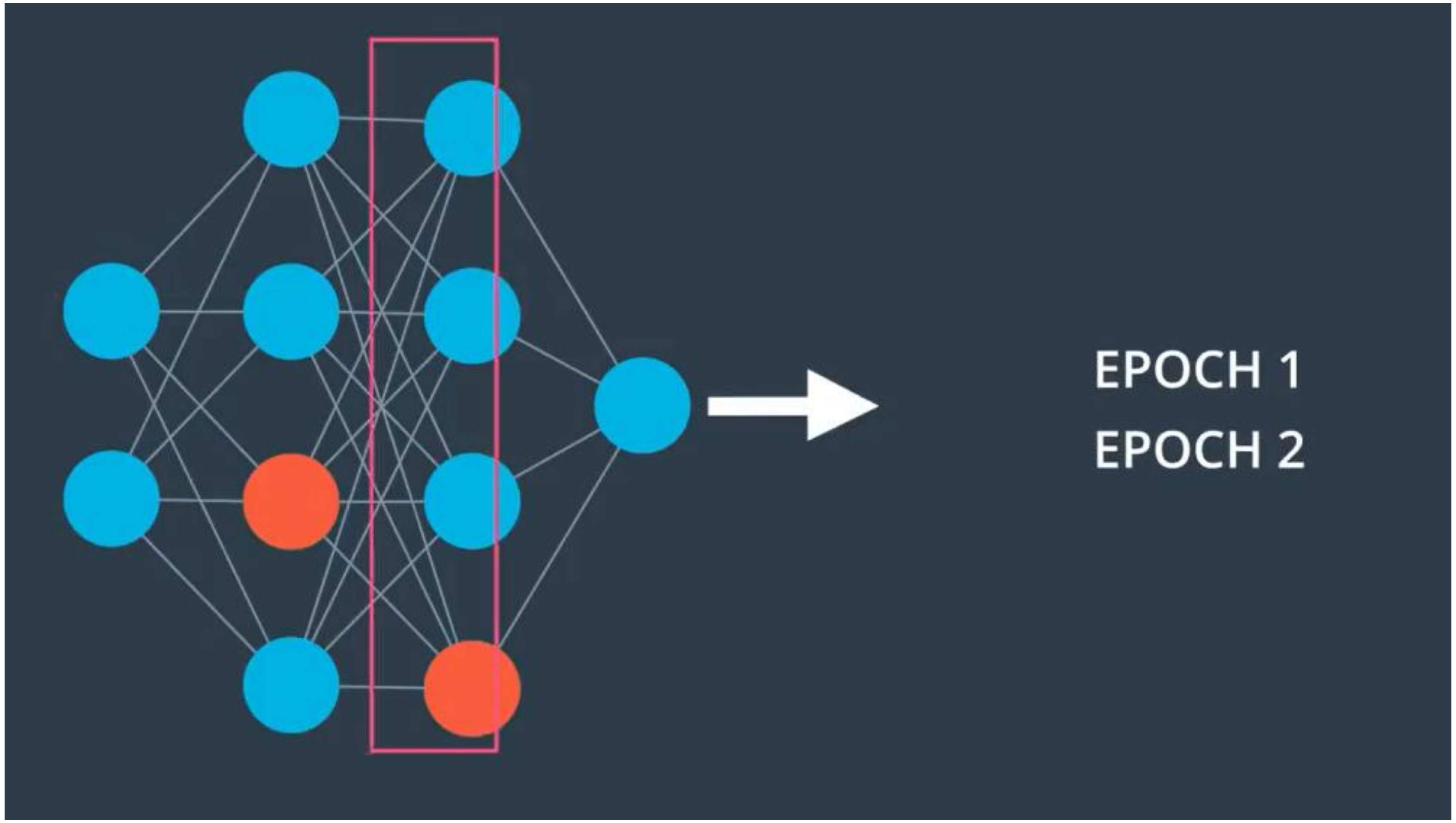
DROPOUT

IN NEURAL NETWORKS!



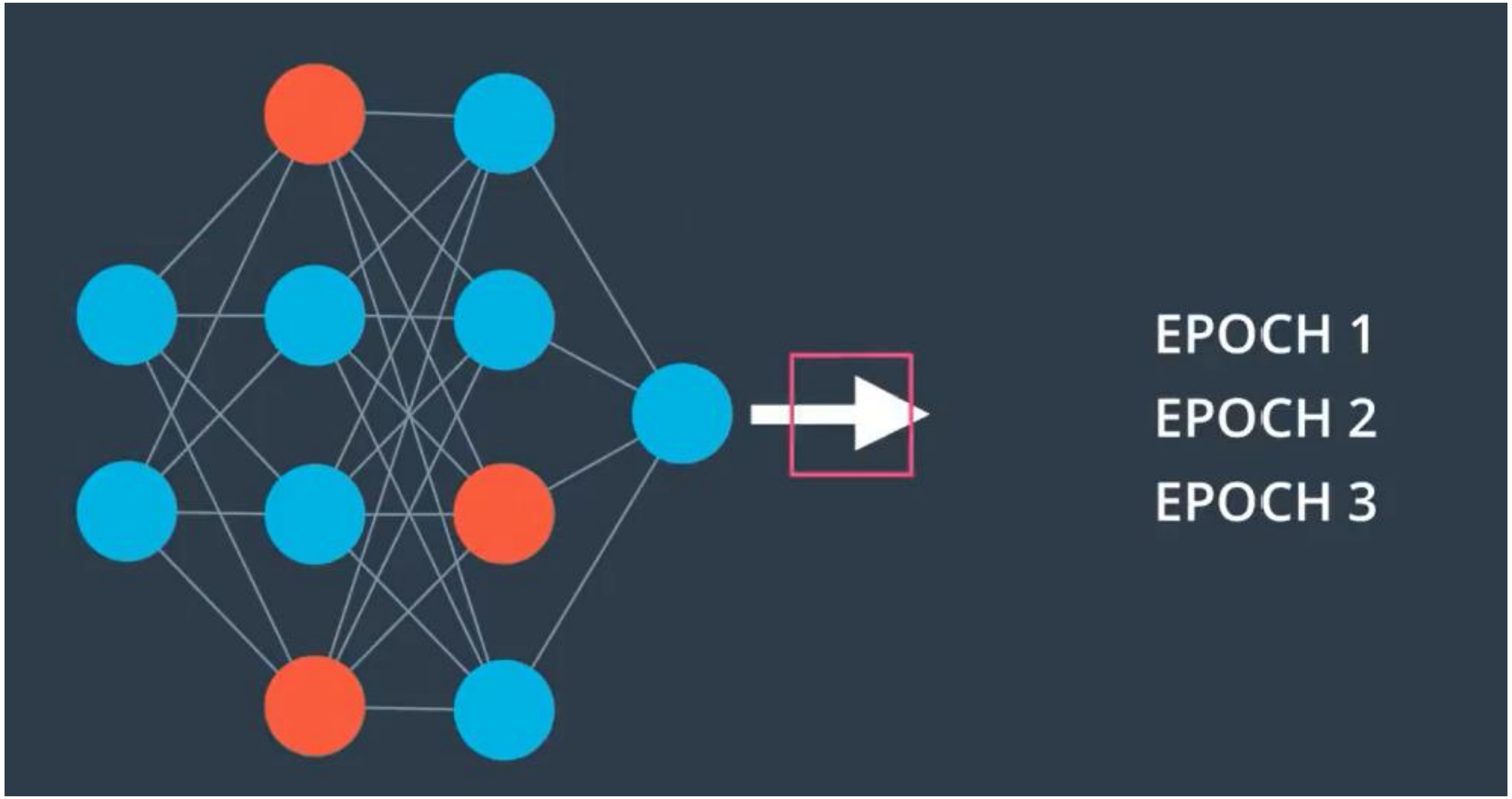
DROPOUT

IN NEURAL NETWORKS!



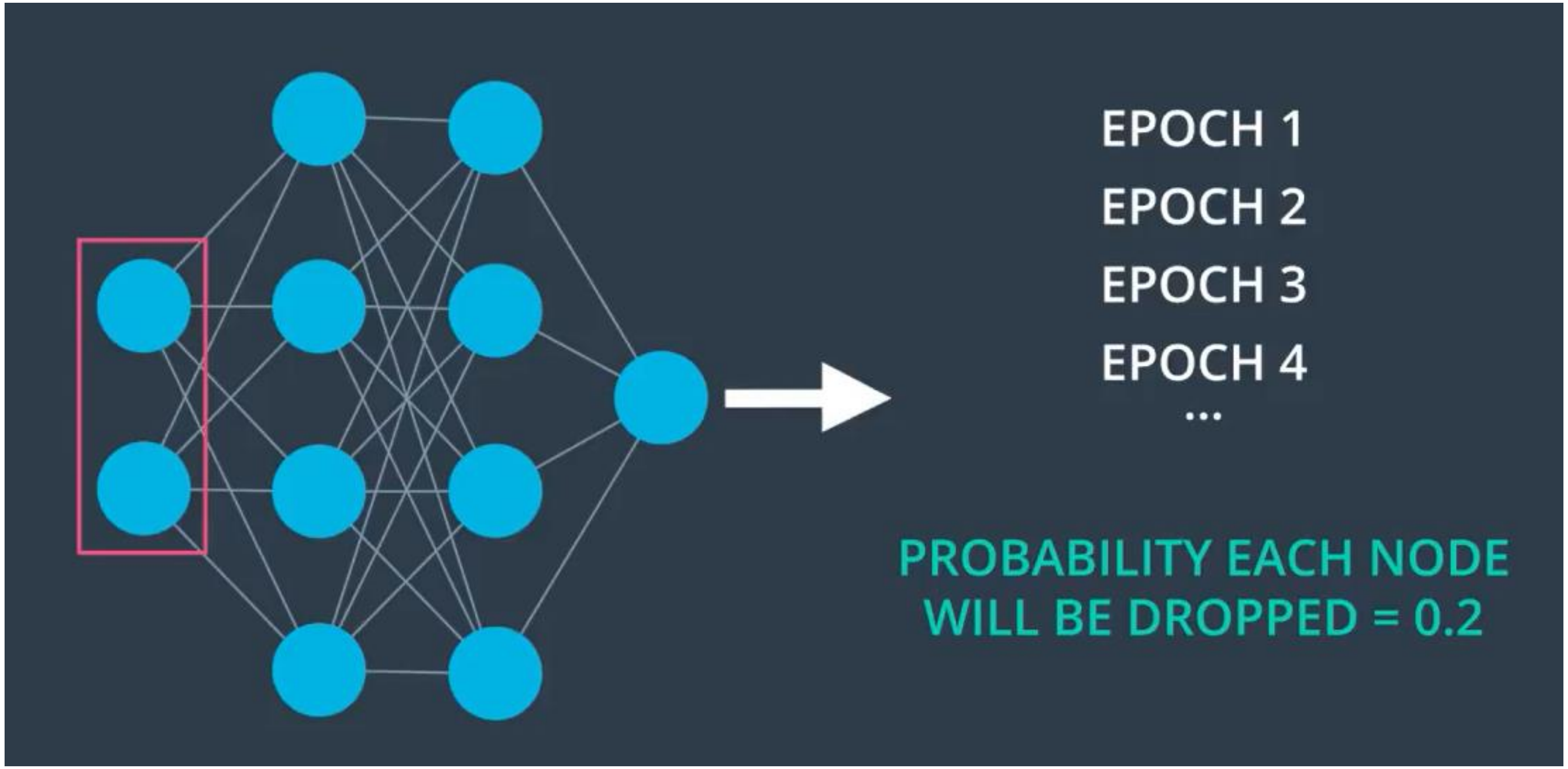
DROPOUT

IN NEURAL NETWORKS!



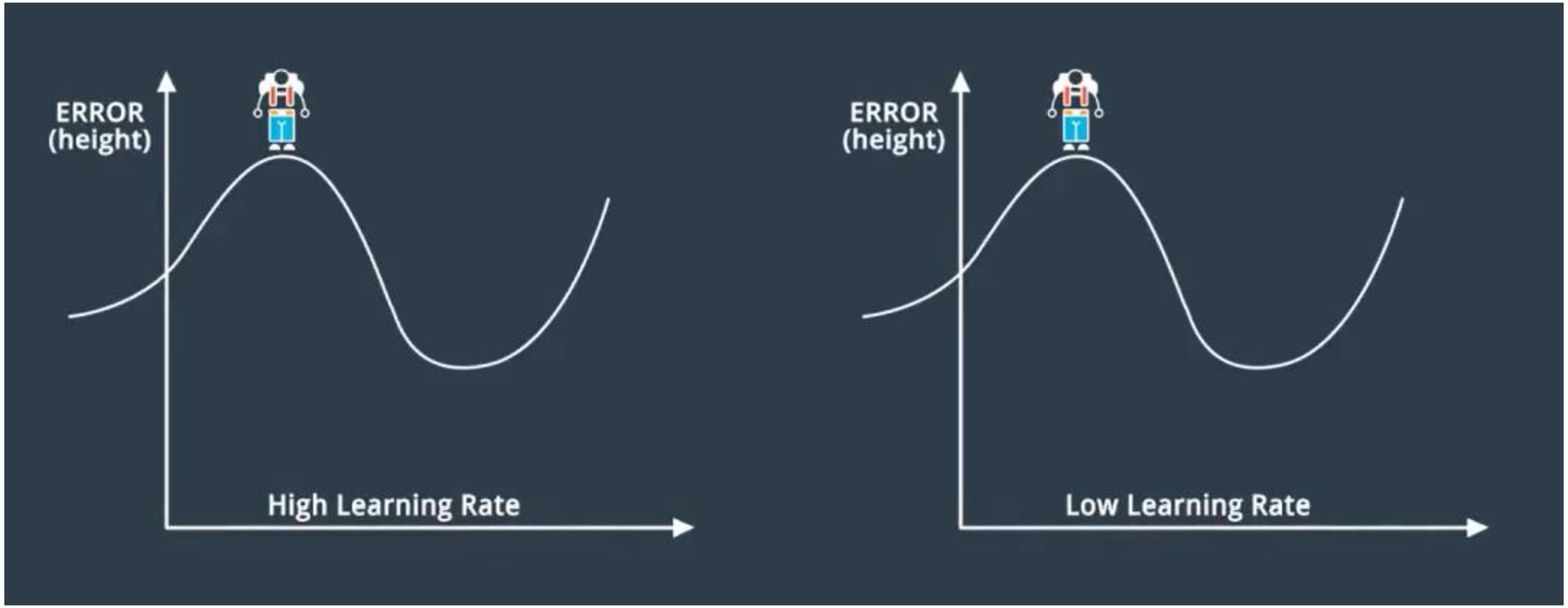
DROPOUT

What does **dropout = 0.2** mean?



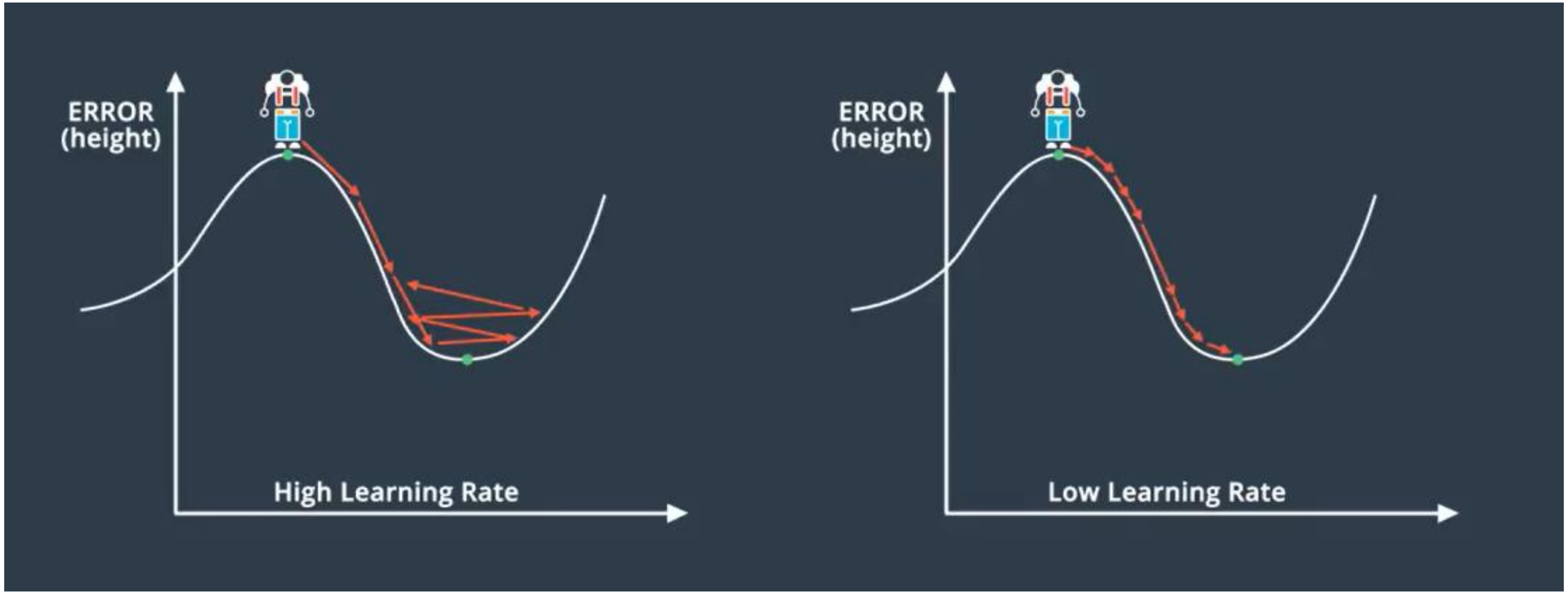
LEARNING RATE

What learning rate should we use?



LEARNING RATE

What learning rate should we use?



LEARNING RATE

What learning rate should we use?