UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

Deep Learning

Session 5

Training Neural Networks

Applied Data Science

2024/2025

Objective Function: Analogous to Training

Babies to not throw food on the floor

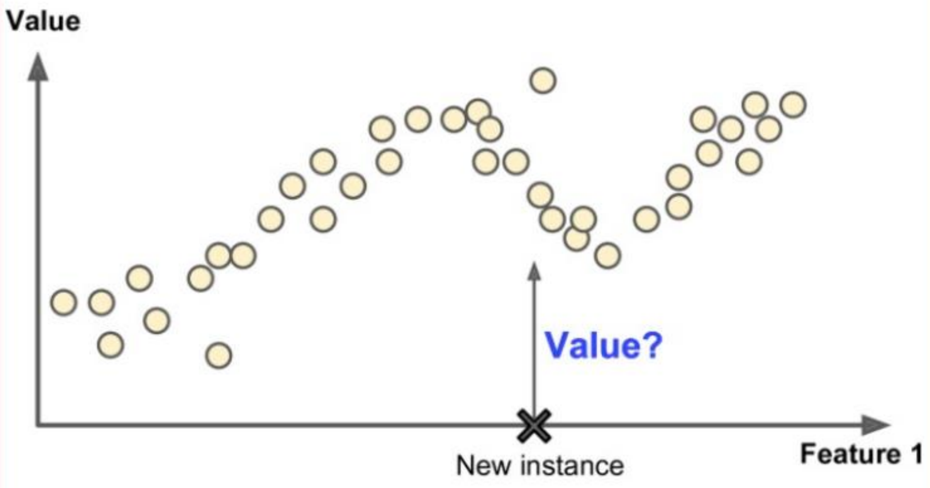
https://www.youtube.com/watch?v=58Pr-QrVNqU

Dogs to learn to sit

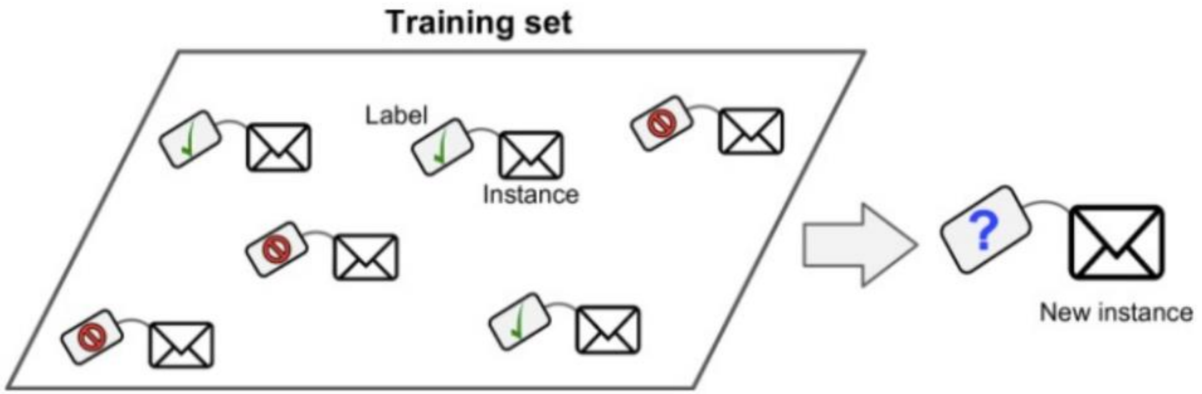
https://www.akc.org/expert-advice/training/how-to-become-a-dog-trainer/

Objective Function: Learn Model Parameters that Achieve a Specified Goal

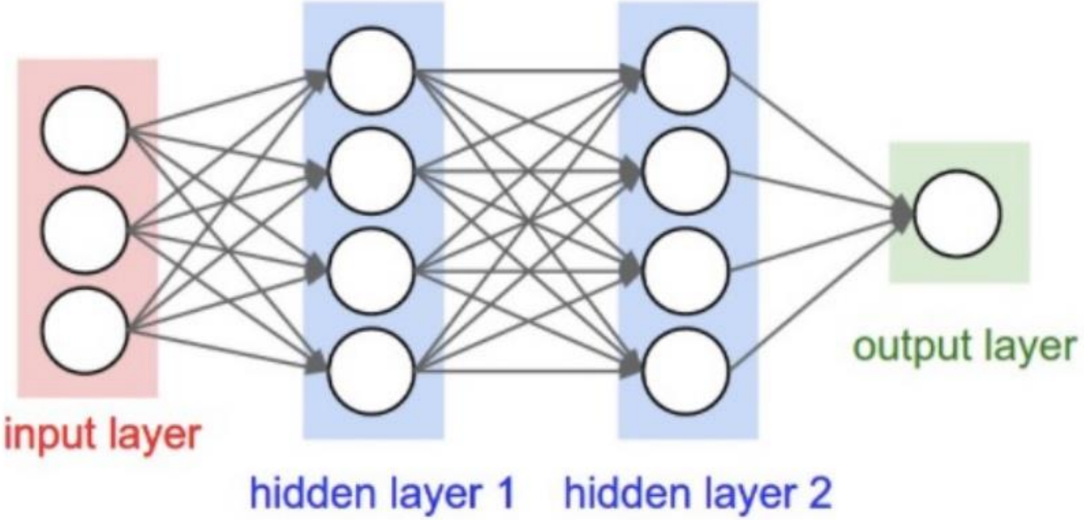
Regression
(predict **continuous** value)



Classification
(predict **discrete** value)



Objective Function: Learn Model Parameters that Achieve a Specified Goal



e.g., make as small as possible the squared error (aka, L2 loss, quadratic loss)

Mean taken over *n* instances

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

True value

Predicted value

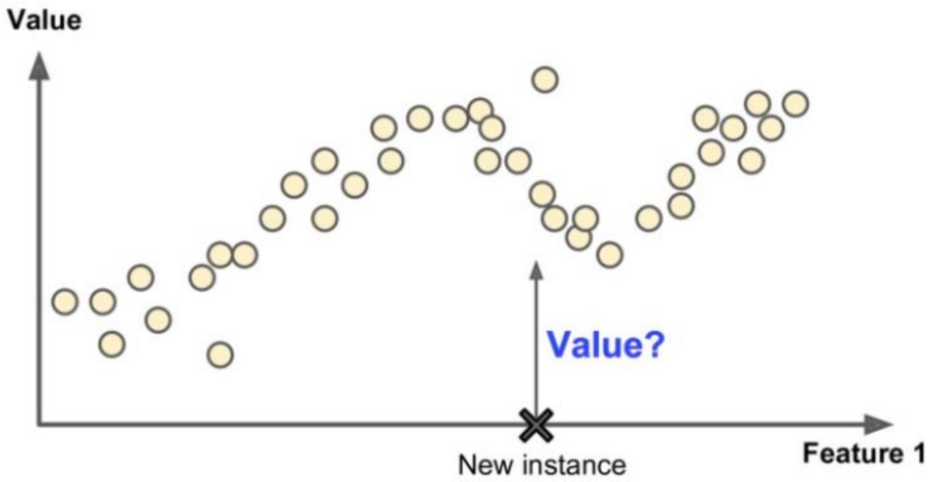
What is the range of possible values?

- Minimum: 0
    - i.e., all correct predictions
- Maximum: Infinity
    - i.e., incorrect predictions

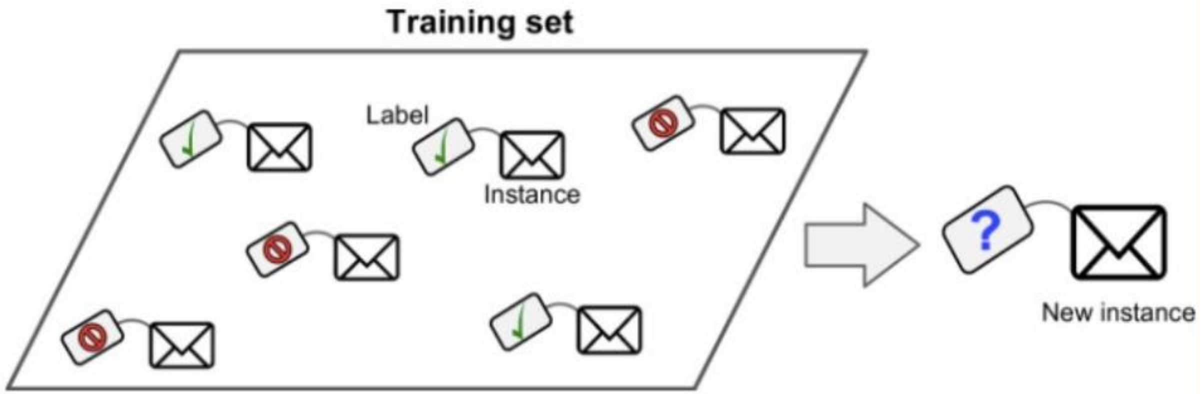
Objective Function: Learn Model Parameters that Achieve a Specified Goal

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

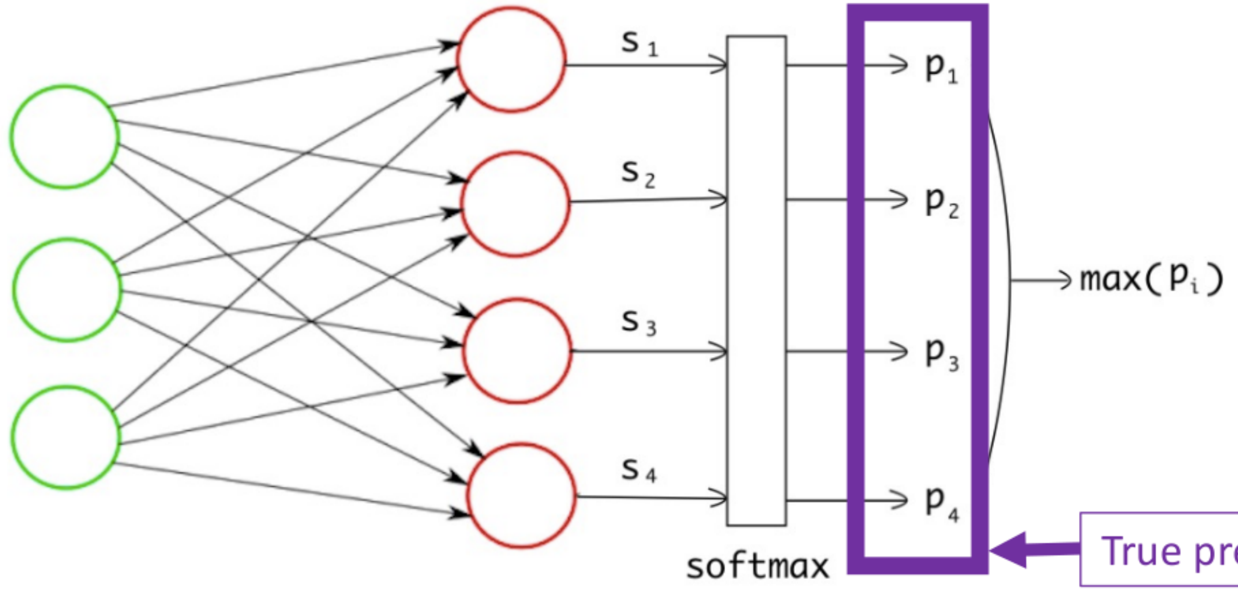
Regression
(predict **continuous** value)



Classification
(predict **discrete** value)



Objective Function: Learn Model Parameters that Achieve a Specified Goal



e.g., make as small as possible the distance between predicted and true class distributions with cross entropy loss

True prediction is 1 hot vector (i.e., one 1 and the rest 0s)

https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede

Objective Function: Learn Model Parameters that Achieve a Specified Goal

Probability distribution of predicted class

Probability distribution of true class

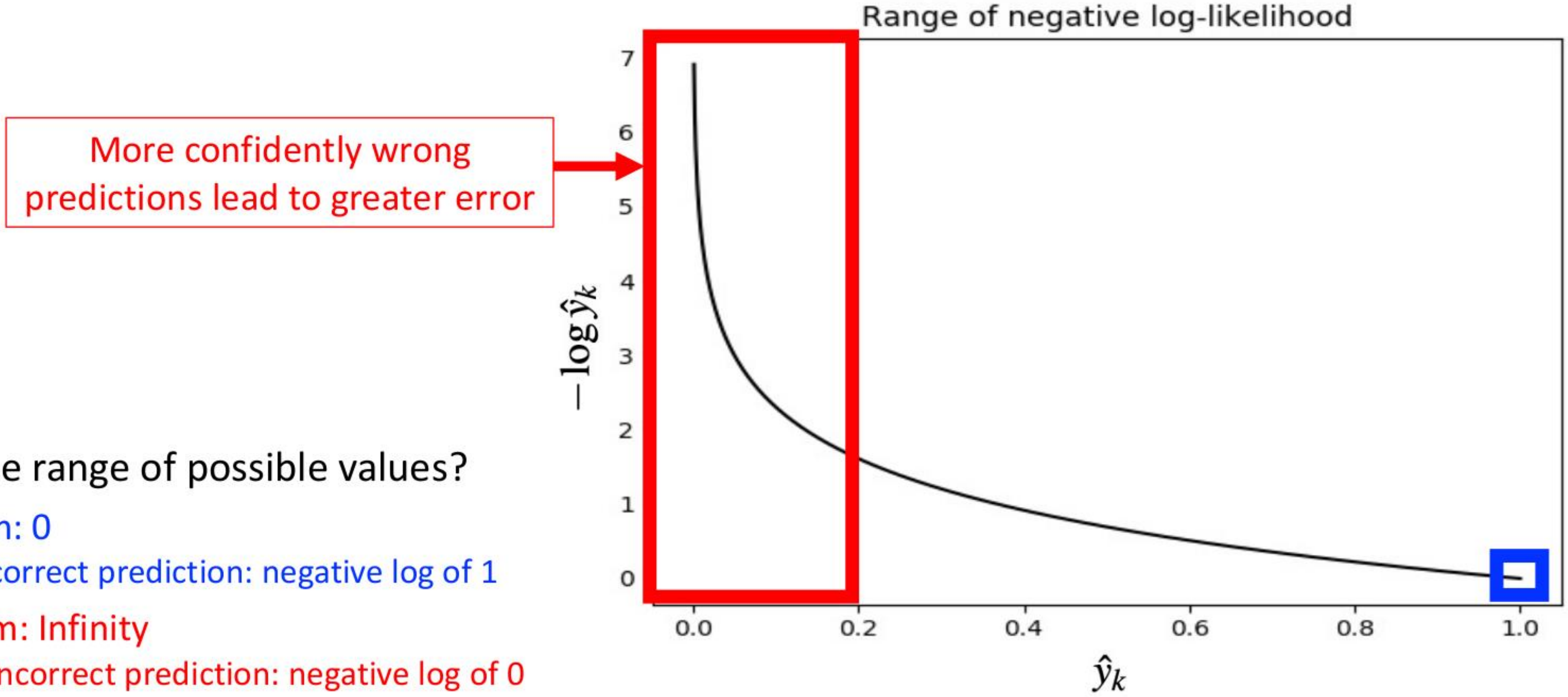
Number of classes?

$$L_{\mathrm{CE}}(\hat{y}, y) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

What is the range of possible values?

- Minimum: 0
  - i.e., correct prediction: negative log of 1
- Maximum: Infinity
  - i.e., incorrect prediction: negative log of 0

Objective Function: Learn Model Parameters that Achieve a Specified Goal

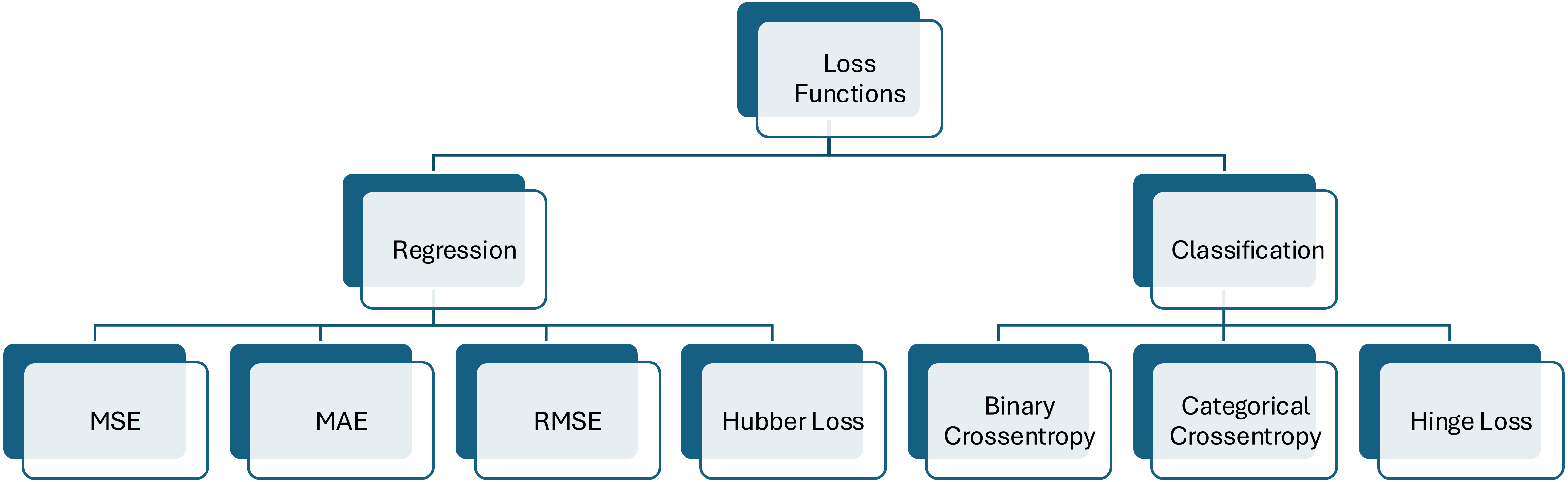


What is the range of possible values?

- Minimum: 0
  - i.e., correct prediction: negative log of 1
- Maximum: Infinity
  - i.e., incorrect prediction: negative log of 0

Objective Function

- Many objective functions exist, and we will explore popular ones in this course!



Loss Functions

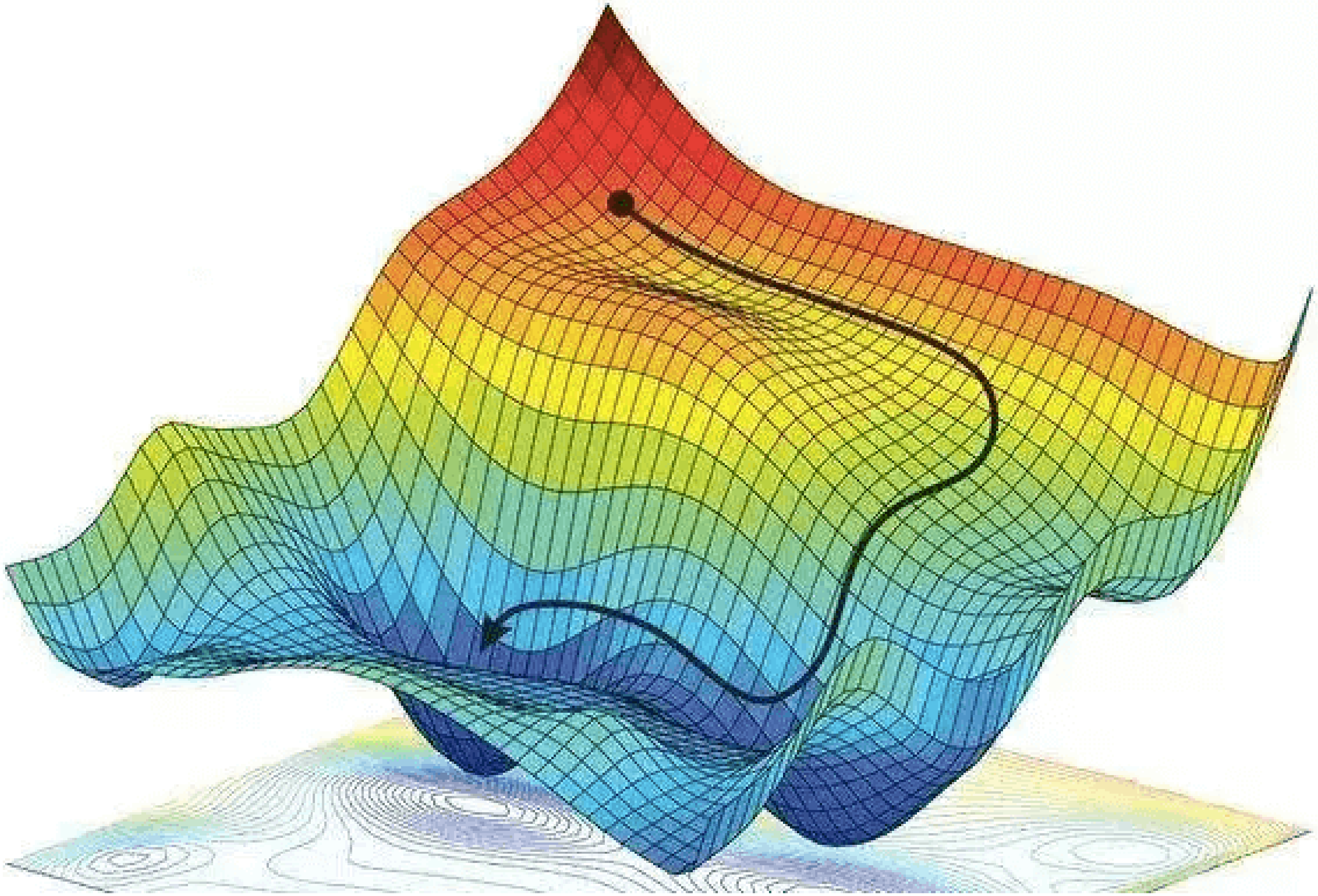
10 Most Common Loss Functions in Machine Learning

blog.DailyDoseofDS.com

Loss Function Name	Description	Function
Regression Loss Functions		
Mean Bias Error	Captures average bias in prediction. But is rarely used for training.	$\mathcal{L}_{MBE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))$
Mean Absolute Error	Measures absolute average bias in prediction. Also called L1 Loss.	$\mathcal{L}_{MAE} = \frac{1}{N}\sum_{i=1}^{N}\lvert y_i - f(x_i)\rvert$
Mean Squared Error	Average squared distance between actual and predicted. Also called L2 Loss.	$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$
Root Mean Squared Error	Square root of MSE. Loss and dependent variable have same units.	$\mathcal{L}_{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2}$
Huber Loss	A combination of MSE and MAE. It is parametric loss function.	$\mathcal{L}_{\text{Huberloss}} = \begin{cases} \frac{1}{2}(y_i - f(x_i))^2 & : \lvert y_i - f(x_i)\rvert \le \delta \\ \delta(\lvert y_i - f(x_i)\rvert - \frac{1}{2}\delta) & : otherwise \end{cases}$
Log Cosh Loss	Similar to Huber Loss + non-parametric. But computationally expensive.	$\mathcal{L}_{\text{LogCosh}} = \frac{1}{N}\sum_{i=1}^{N} log(cosh(f(x_i) - y_i))$
Classification Loss Functions (Binary + Multi-class)		
Binary Cross Entropy (BCE)	Loss function for binary classification tasks.	$\mathcal{L}_{BCE} = \frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(x_i)) + (1 - y_i) \cdot log(1 - p(x_i))$
Hinge Loss	Penalizes wrong and right (but less confident) predictions. Commonly used in SVMs.	$\mathcal{L}_{\text{Hinge}} = max(0, 1 - (f(x) \cdot y))$
Cross Entropy Loss	Extension of BCE loss to multi-class classification.	$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}.log(f(x_{ij}))$ $N: samples; M: classes$
KL Divergence	Minimizes the divergence between predicted and true probability distribution	$\mathcal{L}_{KL} = \sum_{i=1}^{N} y_i \cdot log(\frac{y_i}{f(x_i)})$

https://blog.dailydoseofds.com/p/10-regression-and-classification

Gradient Descent: How to Learn?



Loss Optimization

- We want to find the network weights that achieve the lowest loss!

$$W^* = \underset{W}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x^{(i)}; W), y^{(i)})$$

$$W^* = \underset{W}{\operatorname{argmin}} J(W)$$

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. Calculate error

- e.g., learn linear model for converting kilometers to miles when only observing the input “miles” and output “kilometers”

Miles → Kilometers = miles x constant → Kilometers

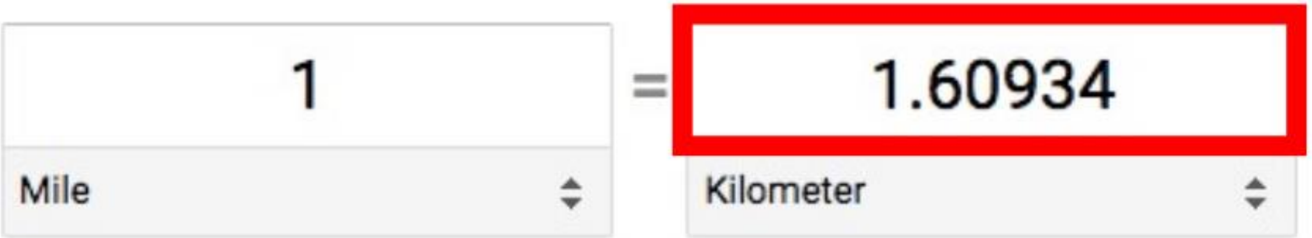
Gradient Descent: Intuition

- Repeat:
  1. **Guess**
  2. Calculate error

1 → Kilometers = miles x constant

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. **Calculate error**



1 → Kilometers = miles x constant → Error = Guess - Correct

Gradient Descent: Intuition

- Repeat:
  1. **Guess**
  2. Calculate error

1.5 ⟶ Kilometers = miles x constant

Gradient Descent: Intuition

- Repeat:
  1. Guess
  2. **Calculate error**

1.5 → Kilometers = miles x constant → Error = Guess - Correct

- Idea: iteratively adjust the constant (i.e., model parameter) to try to reduce the error

Gradient Descent: Intuition

- Iteratively search for values that solve optimization problem (i.e., minimize or maximize an objective function)

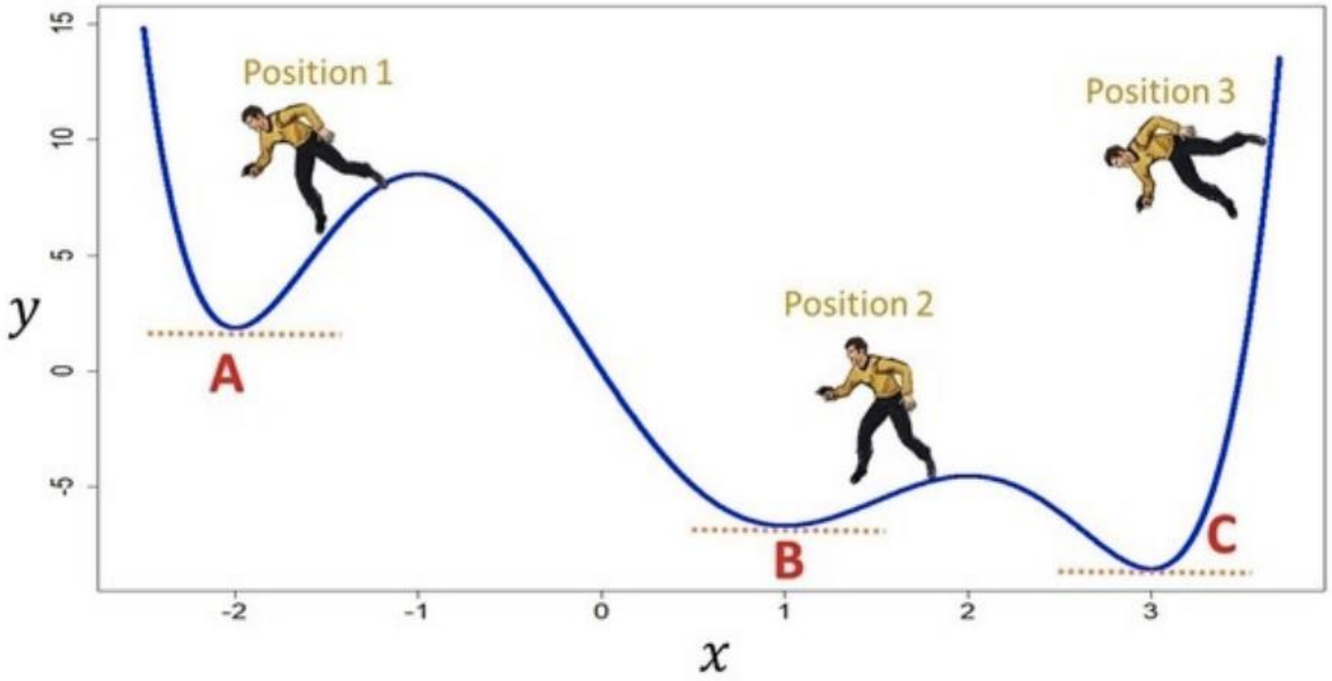
- When **minimizing** the objective function, it also is often called interchangeably the **cost function**, **loss function**, or **error function**.

Gradient Descent: Intuition

- Idea: use derivatives!
  - Derivatives tell us how a small change in the input x affects the output $f(x)$.
  - Specifically, the derivative $f'(x)$ represents the rate of change of the function at a given point, indicating how much $f(x)$ will change in response to a small change in x.
  - Gradient is a vector that indicates how $f(x)$ changes as each function variable changes (i.e., partial derivatives).

Gradient Descent: Intuition

- Gradient descent:
  - Iteratively take steps in the opposite direction of the gradient to minimize the function



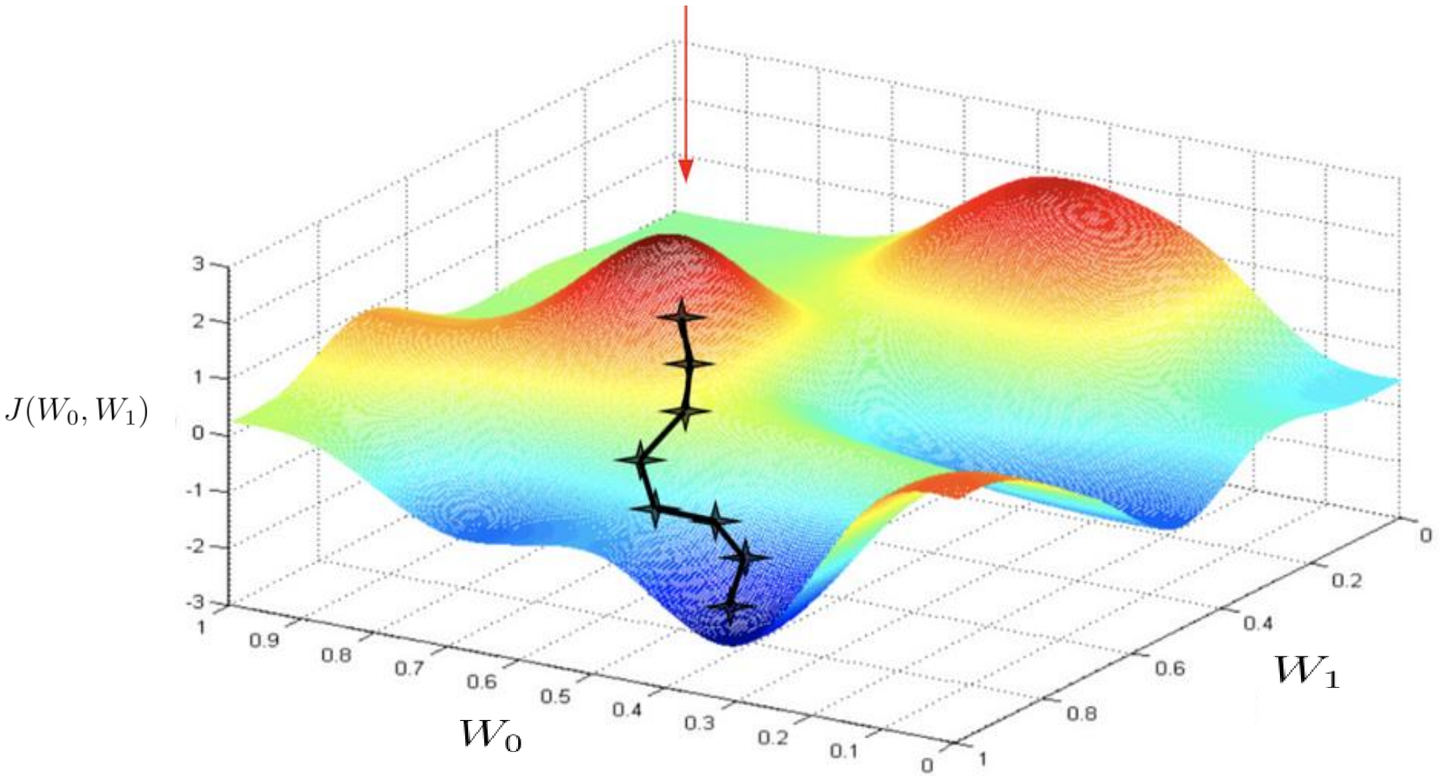
Which letter(s) are the global minima?

Which letter(s) are local minima?

Gradient Descent

$$W^* = \underset{W}{\operatorname{argmin}} J(W)$$

1. Randomly pick an initial (W_0, W_1)



2. Compute the gradiente: $\dfrac{\partial J(W)}{\partial W}$

3. Take a small step in the opposite direction of the gradient.

4. Repeat until convergence.

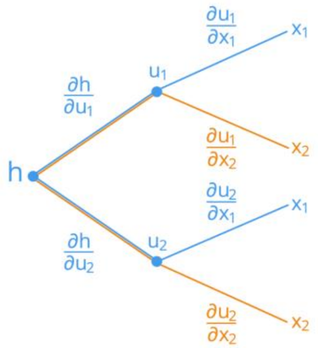
Gradient Descent

- Algorithm:
  1. Initialize weights randomly $\sim N(0, \sigma^2)$
  2. Loop until convergence
  3. Compute gradient $\frac{\partial J(W)}{\partial W}$
  4. Update weights $W \leftarrow W - \eta \frac{\partial J(w)}{\partial w}$
  5. Return weights

Computing Gradients: Backpropagation

- How does a small change in one weight (e.g.,w_2) affect the final loss $J(W)$?

Computing Gradients: Backpropagation



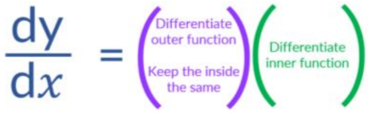
$$\frac{\partial h}{\partial x_1} = \frac{\partial h}{\partial u_1} \cdot \frac{\partial u_1}{\partial x_1} + \frac{\partial h}{\partial u_2} \cdot \frac{\partial u_2}{\partial x_1}$$

$$\frac{\partial h}{\partial x_2} = \frac{\partial h}{\partial u_1} \cdot \frac{\partial u_1}{\partial x_2} + \frac{\partial h}{\partial u_2} \cdot \frac{\partial u_2}{\partial x_2}$$

$$\frac{\partial J(W)}{\partial w_2} =$$

Let's use the chain rule!

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$



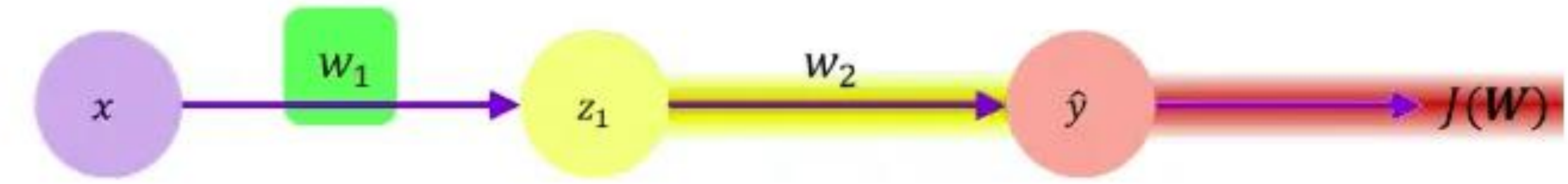
$$y(x) = (x^2 + 1)^3 \longrightarrow ?$$

Computing Gradients: Backpropagation

$$\frac{\partial J(W)}{\partial w_2} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_2}$$

Computing Gradients: Backpropagation

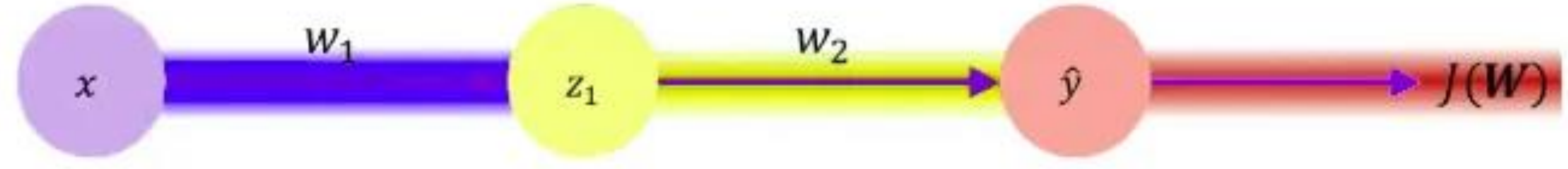
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA



$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_1}$$

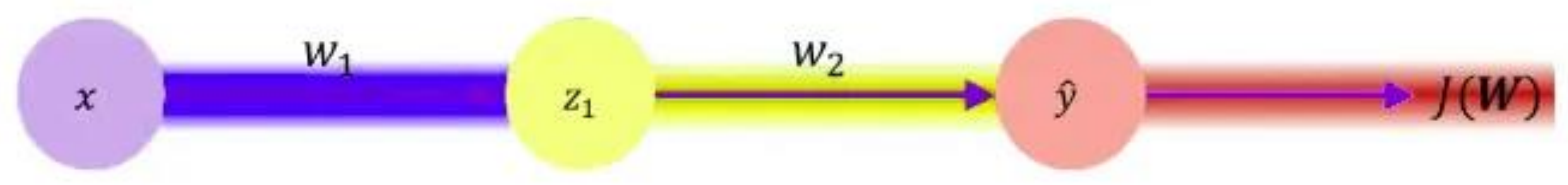
Apply the chain rule! Apply the chain rule!

Computing Gradients: Backpropagation



$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

Computing Gradients: Backpropagation



$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

Repeat this for **every weight in the network** using gradients from later layers.

Gradient Descent: How Often to Update?

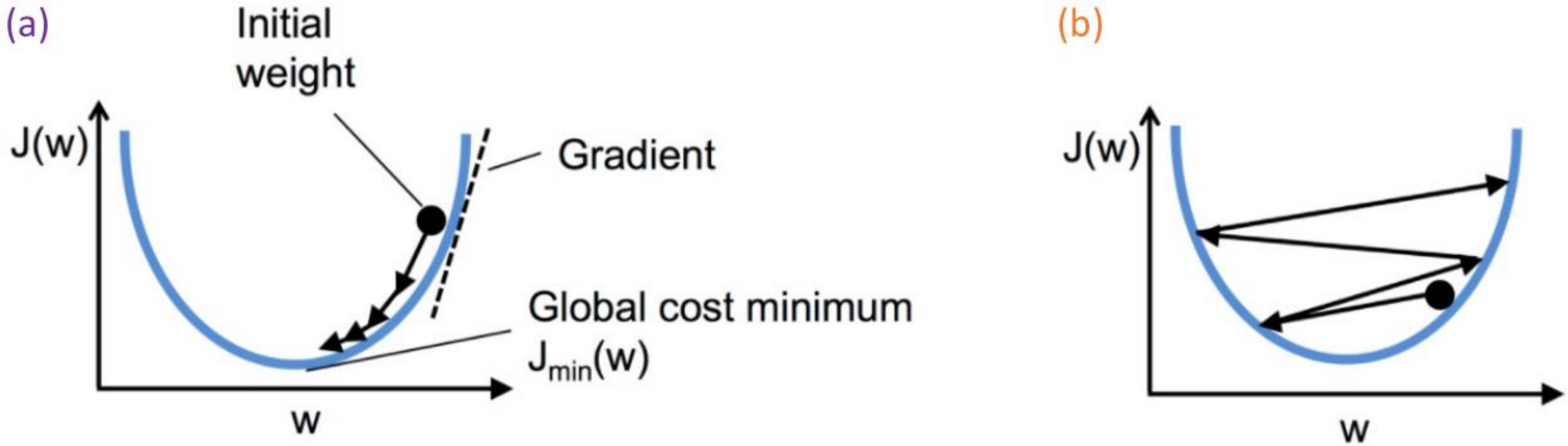
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **Batch Gradient Descent:**
  - Uses all training examples in each update
  - Less noisy updates but slow or infeasible for large datasets
- **Stochastic Gradient Descent (SGD):**
  - Updates with a single example per iteration
  - Fast and memory-efficient for large datasets, but updates can be noisy
- **Mini-batch Gradient Descent:**
  - Updates with a subset of examples per iteration
  - Reduces noise compared to SGD, scalable to large datasets, but can still be slow with large data

Gradient Descent: How Much to Update?

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Step size = learning rate
  - (a) When learning rate is too small, convergence to good solution will be slow!
  - (b) When learning rate is too large, convergence to a good solution is not possible!



Gradient Descent: How to Choose the Learning Rate?

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

Idea 1

Try lots of different learning rates and see what works better!

Gradient Descent: How to Choose the Learning Rate?

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

Idea 3

Do something smarter!

Design na adaptive learning rate that "adapts" to the landscape!

Adaptive Learning Rates

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Learning rates are no longer fixed!

- Can be made larger or smaller depending on:
  - How large the gradient is.
  - How fast learning is happening.
  - The size of particular weights.
  - etc...