

INTRODUCTION TO DEEP LEARNING

Seminar @ UPC TelecomBCN Barcelona (3rd edition). 22-28 January 2020.



Instructors



Xavier
Giró-i-Nieto



Marta R.
Costa-Jussà



Noé
Casas



Verónica
Vilaplana



Ramon
Morros



Javier
Ruiz

Organizers



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH



Supporters



Google Cloud

GitHub Education

+ info: <http://bit.ly/idl2020>



#DLUPC

Day 2 Lecture 2

Loss functions



Javier Ruiz Hidalgo

javier.ruiz@upc.edu

Associate Professor

Universitat Politècnica de Catalunya
Technical University of Catalonia



Me



- **Javier Ruiz Hidalgo**

- Email: j.ruiz@upc.edu
- Office: UPC, Campus Nord, D5-008

- **Teaching experience**

- Basic signal processing
- Project based
- Image processing & computer vision

- **Research experience**

- Master on hierarchical image representations by UEA (UK)
- PhD on video coding by UPC (Spain)
- Interests in image & video coding, 3D analysis and super-resolution

Acknowledgements



[Kevin McGuinness](#)

kevin.mcguinness@dcu.ie

Research Fellow

Insight Centre for Data Analytics
Dublin City University



[Xavier Giró](#)

xavier.giro@upc.edu

Associate Professor

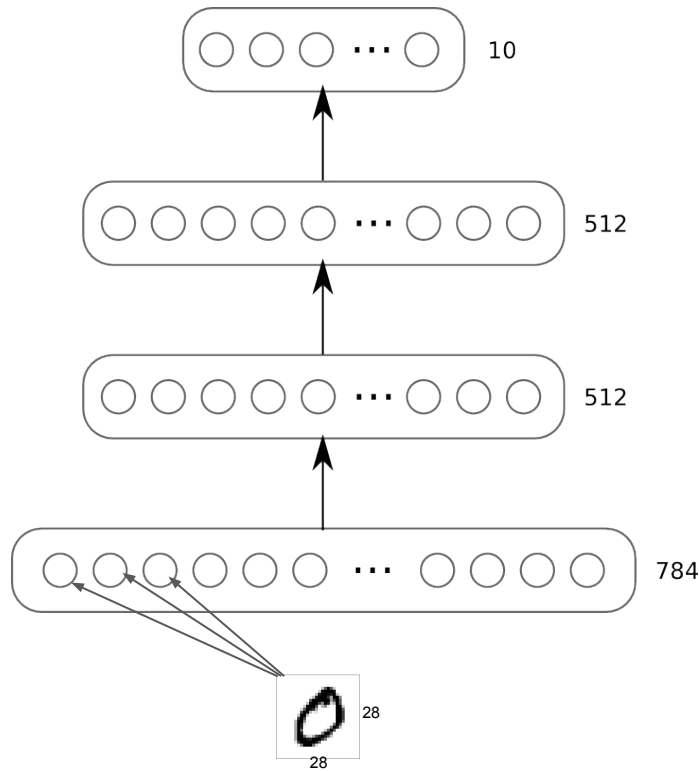
ETSETB TelecomBCN
Universitat Politècnica de Catalunya

Motivation

Model

- 3 layer neural network (2 hidden layers)
- Tanh units (activation function)
- 512-512-10
- **Softmax** on top layer
- **Cross entropy loss** ??

Layer	#Weights	#Biases	Total
1	784 x 512	512	401,920
2	512 x 512	512	262,656
3	512 x 10	10	5,130
			669,706



Outline

- **Introduction**
 - Definition, properties, training process
- **Common types of loss functions**
 - Regression
 - Classification
- **Regularization**

Nomenclature

loss function

=

cost function

=

objective function

=

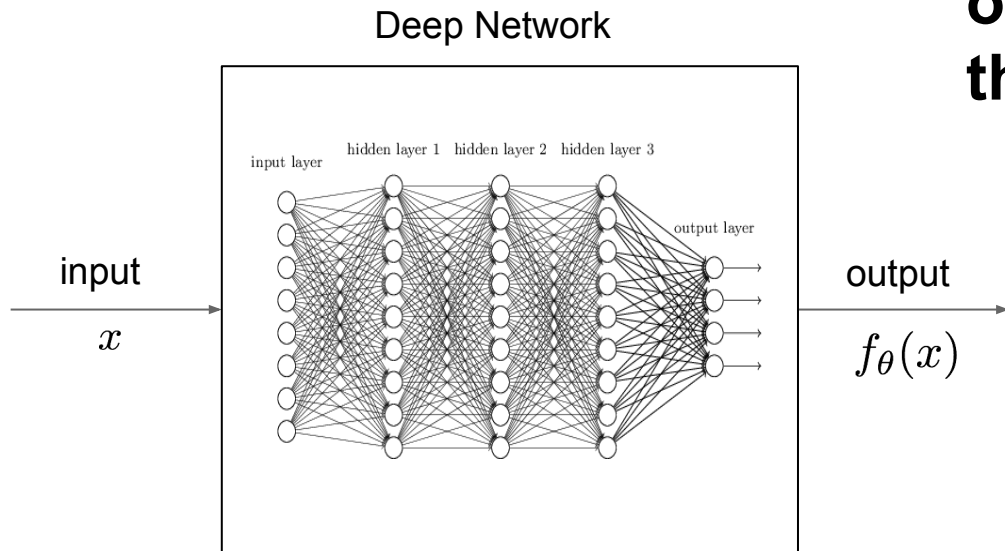
error function

Definition

In a supervised deep learning context the **loss function** measures the **quality** of a particular set of parameters based on how well the output of the network **agrees** with the ground truth labels in the training data.

Loss function (1)

How good does
our network with
the training data?



labels (ground truth)

input

$$\mathcal{L} = distance(f_{\theta}(x), y)$$

error

parameters (weights, biases)

Loss function (2)

- The loss function **does not** want to **measure** the **entire performance** of the network against a validation/test dataset.



Loss function (3)

- The loss function is used to **guide the training process** in order to find a set of parameters that reduce the value of the loss function.



Training process

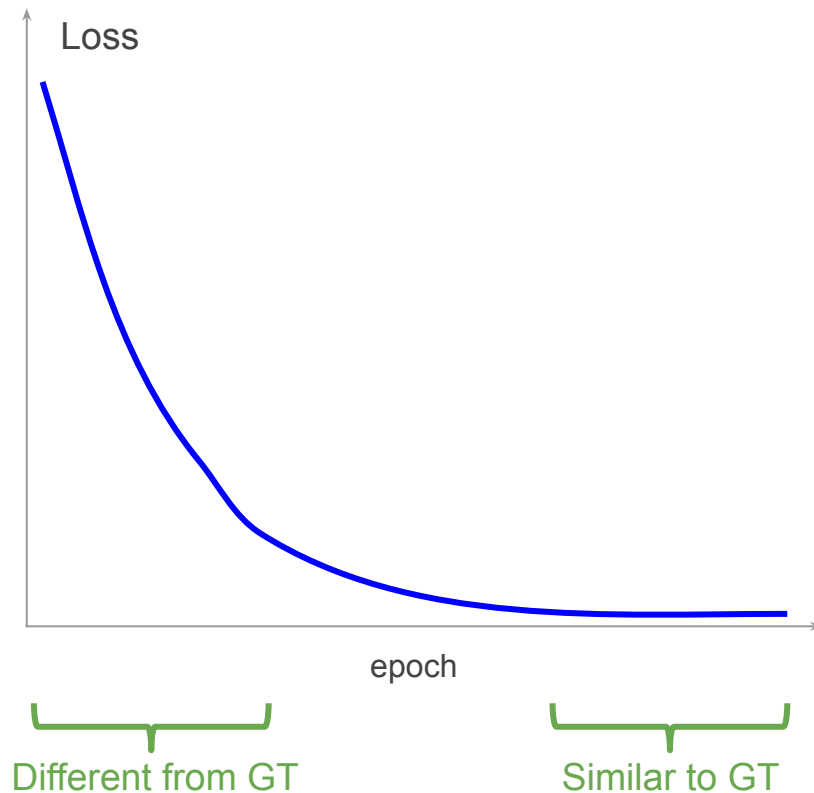
Stochastic gradient descent

- Find a set of parameters which make the loss as small as possible.
- Change parameters at a rate determined by the partial derivatives of the loss function:

$$\frac{\partial \mathcal{L}}{\partial w} \quad \frac{\partial \mathcal{L}}{\partial b}$$

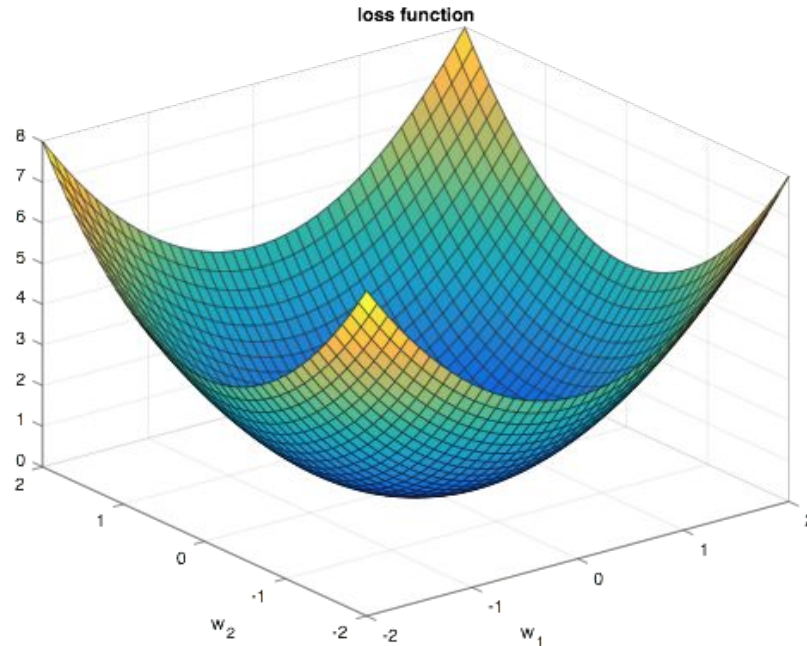
Properties (1)

- Minimum (0 value) when the output of the network is equal to the ground truth data.
- Increase value when output differs from ground truth.



Properties (2)

- Ideally \rightarrow convex function



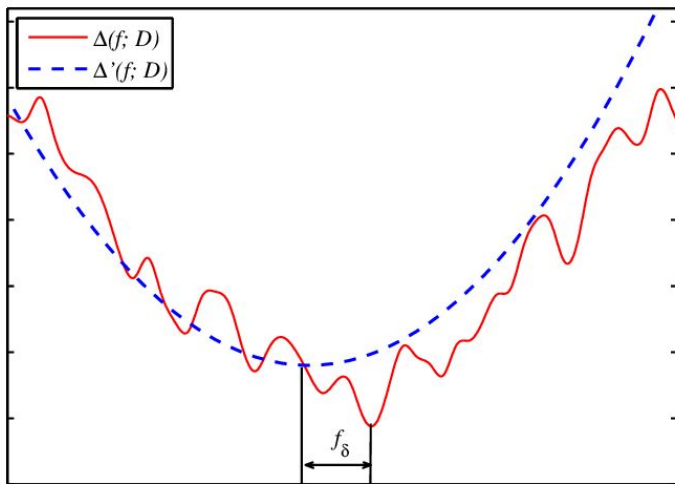
Properties (2)

- In reality → some many parameters (in the order of millions) than it is not convex



Properties (3)

- Varies smoothly with changes on the output
 - Better gradients for gradient descent
 - Easy to compute small changes in the parameters to get an improvement in the loss



Outline

- Introduction
 - Definition, properties, training process
- **Common types of loss functions**
 - **Regression**
 - Classification
- Regularization

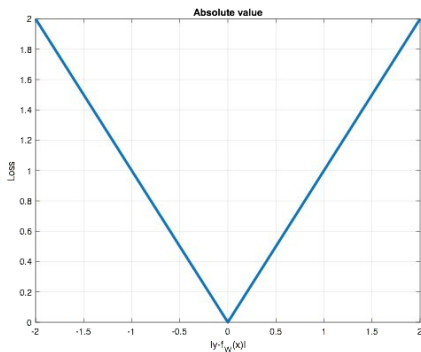
Loss functions for regression (1)

- Loss functions depend on the type of task:
 - Regression: the network predicts **continuous, numeric** variables
 - Example: Length of fishes in images, temperature from latitude/longitude
 - Absolute error, square error, Huber loss

Loss functions for regression (2)

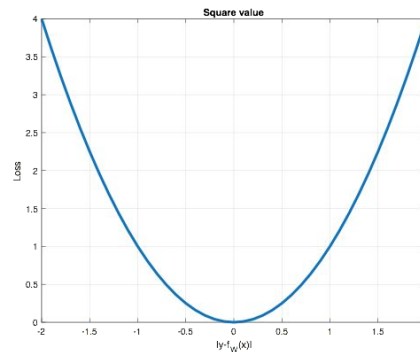
L1 Distance /
Absolute Value

Lighter computation
(only $|\cdot|$)



L2 Distance /
Euclidian Distance / Square Value

Heavier computation
(square)



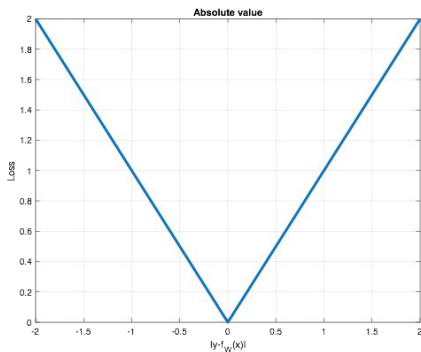
$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n |y_i - f_{\theta}(x_i)|$$

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (y_i - f_{\theta}(x_i))^2$$

Loss functions for regression (3)

L1 Distance /
Absolute Value

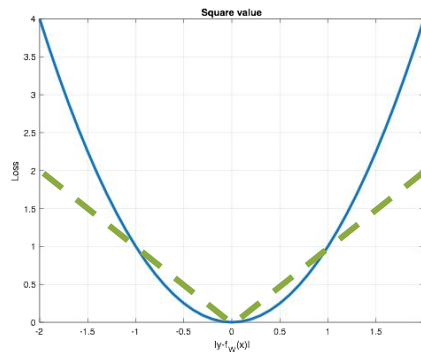
Less penalty to errors > 1
More penalty to errors < 1



$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n |y_i - f_{\theta}(x_i)|$$

L2 Distance /
Euclidian Distance / Square Value

More penalty to error > 1
Less penalty to errors < 1

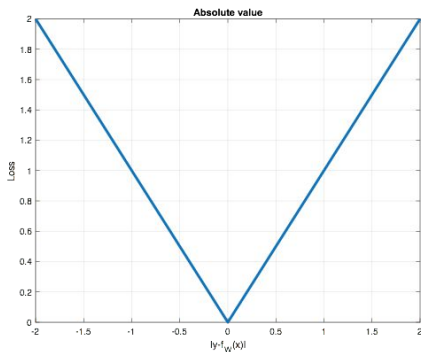


$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (y_i - f_{\theta}(x_i))^2$$

Loss functions for regression (4)

L1 Distance /
Absolute Value

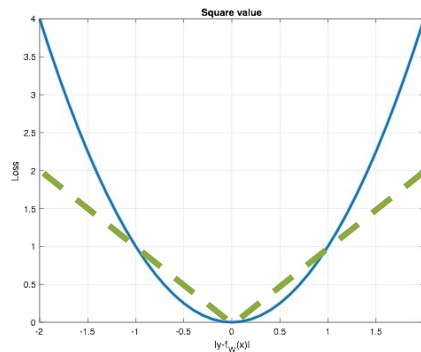
Less sensitive to outliers,
which are very high $y_i - f_{\theta}(x_i)$



$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n |y_i - f_{\theta}(x_i)|$$

L2 Distance /
Euclidian Distance / Square Value

More sensitive to outliers,
which are very high $y_i - f_{\theta}(x_i)$

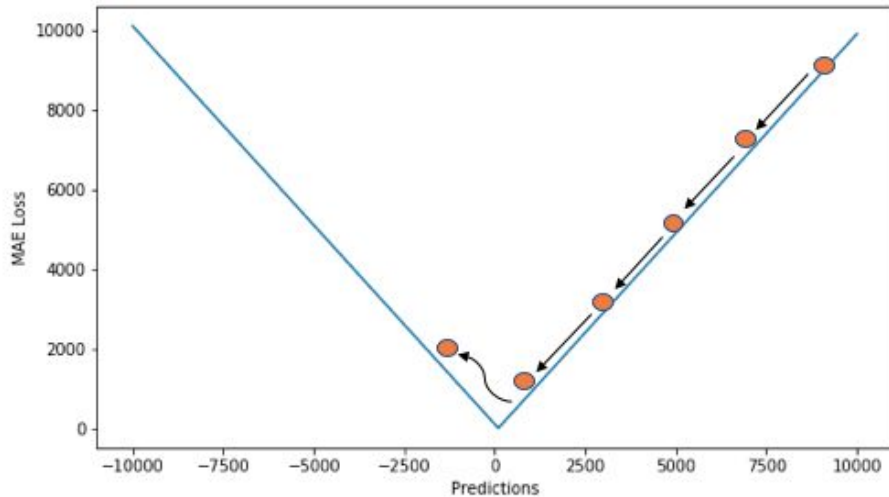


$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (y_i - f_{\theta}(x_i))^2$$

Loss functions for regression (5)

L1 Distance /
Absolute Value

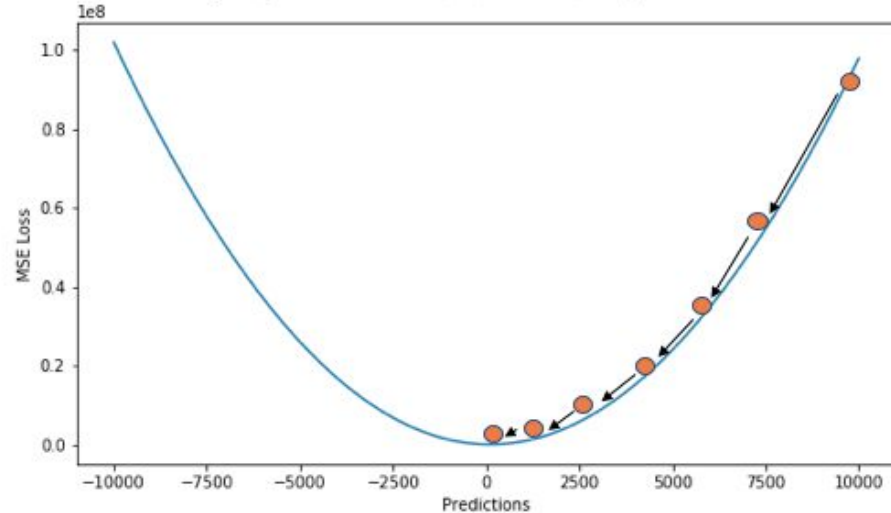
Range of predicted values: (-10,000 to 10,000) | True value: 100



$$\mathcal{L}_i = |y_i - f_{\theta}(x_i)|$$

L2 Distance /
Euclidian Distance / Square Value

Range of predicted values: (-10,000 to 10,000) | True value: 100



$$\mathcal{L}_i = (y_i - f_{\theta}(x_i))^2$$

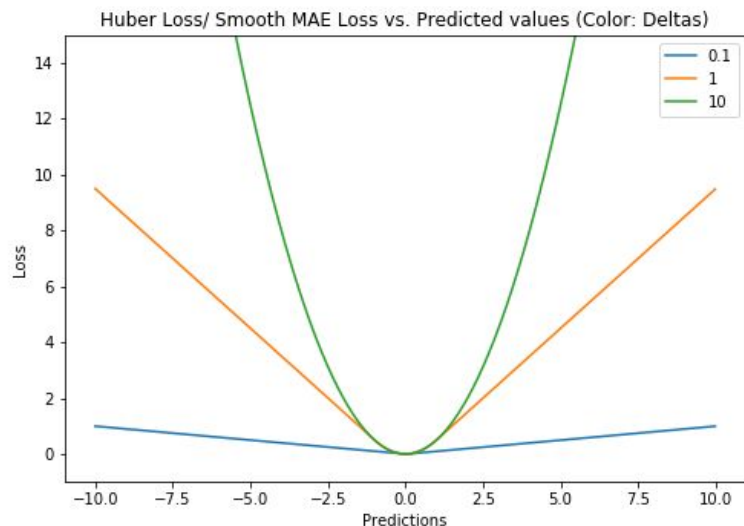
Loss functions for regression (6)

- L1 / absolute loss
 - ✓ More robust to outliers
 - ✗ Large gradients for small loss → Inefficient to find minimum
- L2 / square loss
 - ✗ Sensitive to outliers
 - ✓ More stable and closed form solution (derivatives go to 0)
- In general, **L2 loss is more appropriate**
- Smooth / **Huber loss**

Loss functions for regression (7)

Smooth / Huber loss

$$\mathcal{L}_i = \begin{cases} \frac{1}{2}(y_i - f_{\theta}(x_i))^2 & \text{for } |y_i - f_{\theta}(x_i)| < \delta \\ \delta|y_i - f_{\theta}(x_i)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$



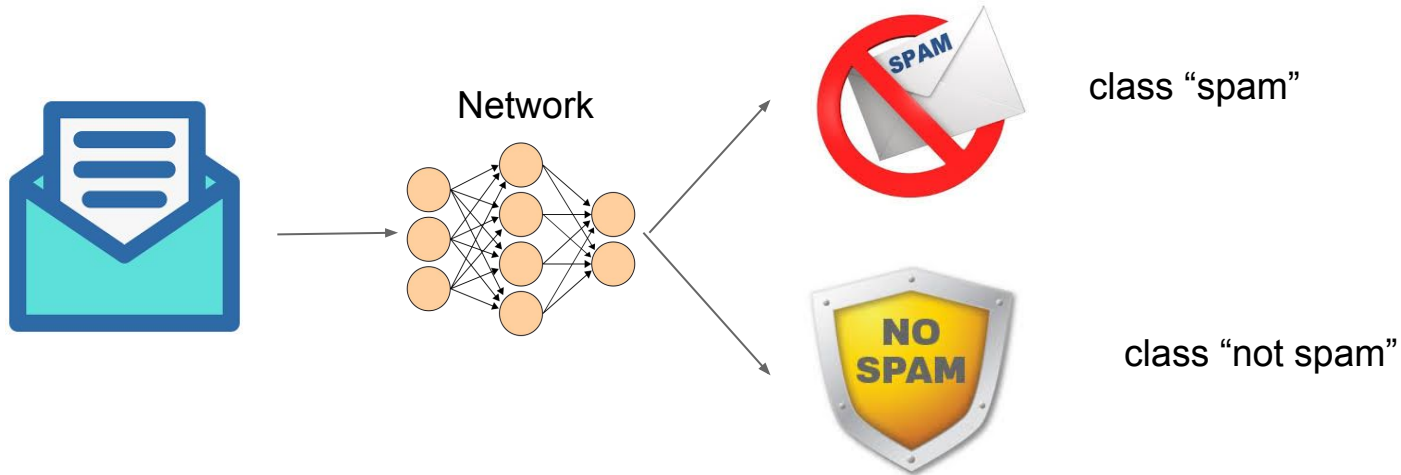
- Behaves like absolute error but becomes quadratic when error is small.
- δ becomes an hyper-parameter (usually $\delta=1$)

Outline

- Introduction
 - Definition, properties, training process
- **Common types of loss functions**
 - Regression
 - **Classification**
- Regularization

Binary classification (1)

- We want the network to classify the input into two classes
 - Spam / Not spam



Binary classification (2)

- How can we create a loss function to improve the scores?
 - Assign a number to each class
 - +1 for “spam” and “-1” for “not spam” and use any regression based loss (square loss)
 - Errors in between labels are treated equal than errors “outside” labels

$$\mathcal{L}_i = (y_i - f_{\theta}(x_i))^2$$

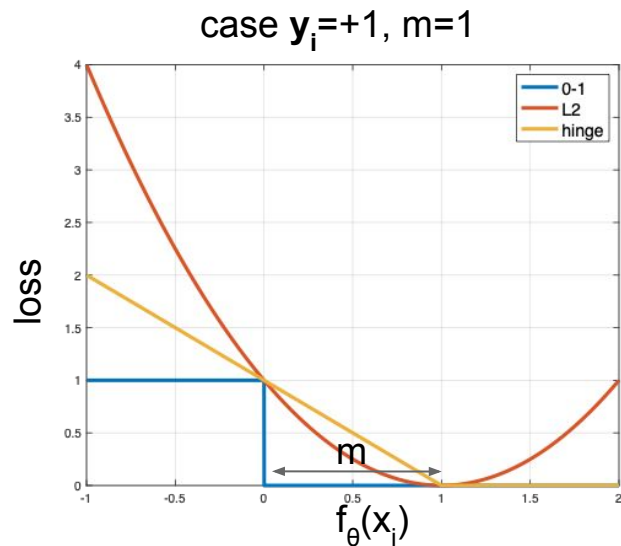
- Non-probabilistic interpretation → **hinge loss**
- Probabilistic interpretation → **cross-entropy loss**



Hinge loss (1)

$$\mathcal{L}_i = \max(0, m - y_i f_{\theta}(x_i))$$

- Labels are $y_i = \pm 1$, m is the margin (usually $m=1$)
- Correct and confident predictions not penalized
- Penalizes incorrect predictions but also correct but not confident (margin)
- Maximizes the margin between our decision boundary and input data



Loss functions for classification

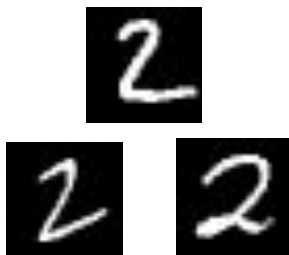
- How can we create a loss function to improve the scores?
 - Non-probabilistic interpretation → hinge loss
 - Probabilistic interpretation → **cross-entropy loss**
 - Transform output of the network into probabilities (softmax)
 - Measure the loss to expected label probabilities
 - We'll review cross-entropy loss in the context of single-label multiclass classification
 - Cross-entropy can also be applied to binary classification
 - Hinge loss could also be extended to multiclass classification

Single-label multi-class classification (1)

We want the network to classify the input into a fixed number of k classes



class "1"



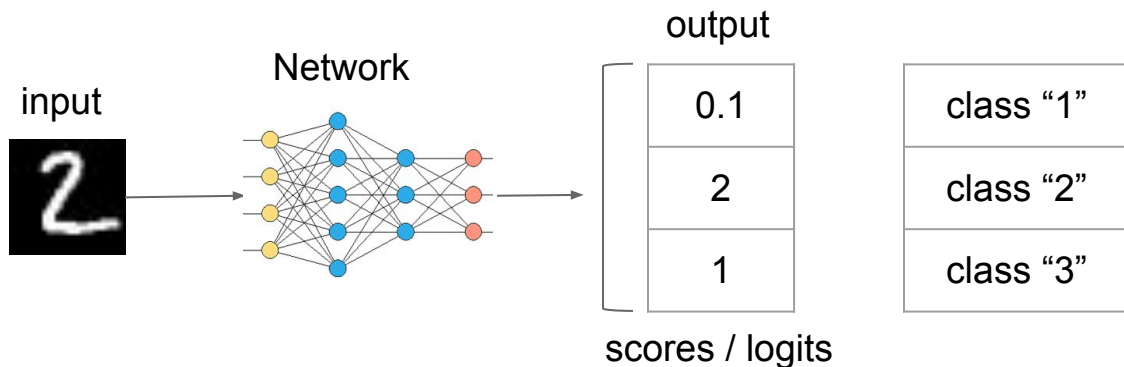
class "2"



class "3"

Single-label multi-class classification (2)

- Each input can have only one label (single-label)
 - One prediction per output class
 - The network will have “k” outputs (number of classes / multi-class)



Cross-entropy loss (1)

- Write multi-class labels (ground truth) into a vector
 - **One-hot encoding**
- Transform scores (output of the network) into probabilities
 - **Softmax**
- Measure the difference between expected probabilities
 - **Negative log likelihood**

One-hot encoding

- Transform each label into a vector (with only 1 and 0)
 - Length equal to the total number of classes “k”
 - Value of 1 for the correct class and 0 elsewhere

class “1”

1
0
0

class “2”

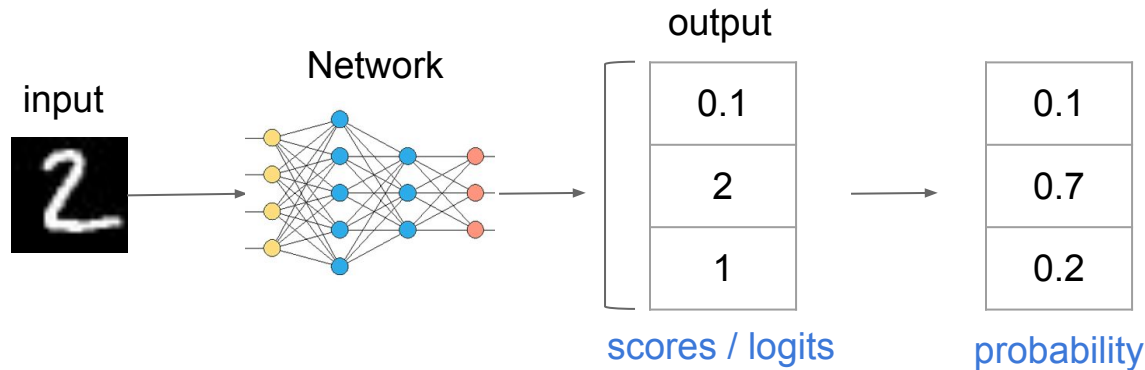
0
1
0

class “3”

0
0
1

Softmax (1)

- Convert scores into confidences/probabilities
 - From 0.0 to 1.0
 - Probability for all classes adds to 1.0

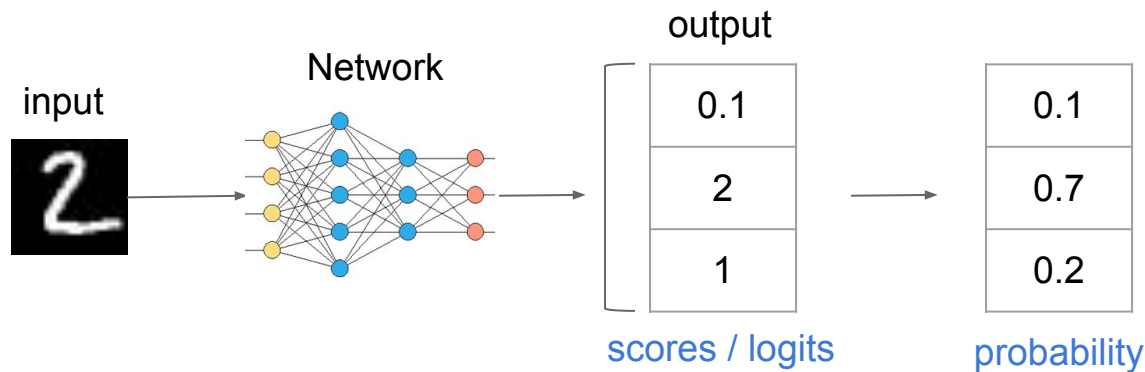


Softmax (2)

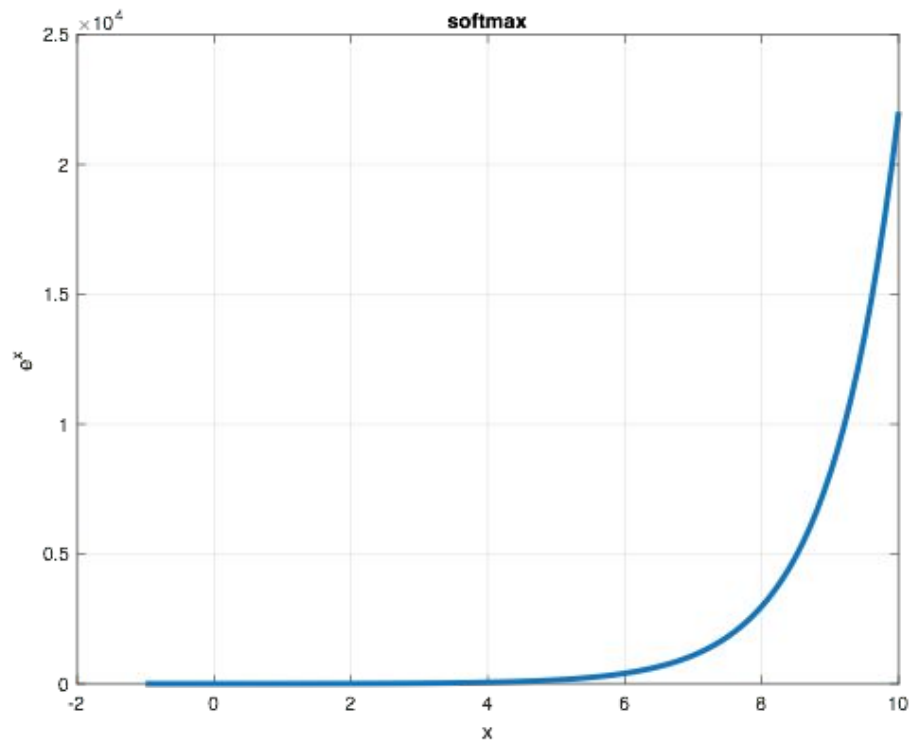
- Softmax function

scores (logits)

$$S(l_i) = \frac{e^{l_i}}{\sum_k e^{l_k}}$$

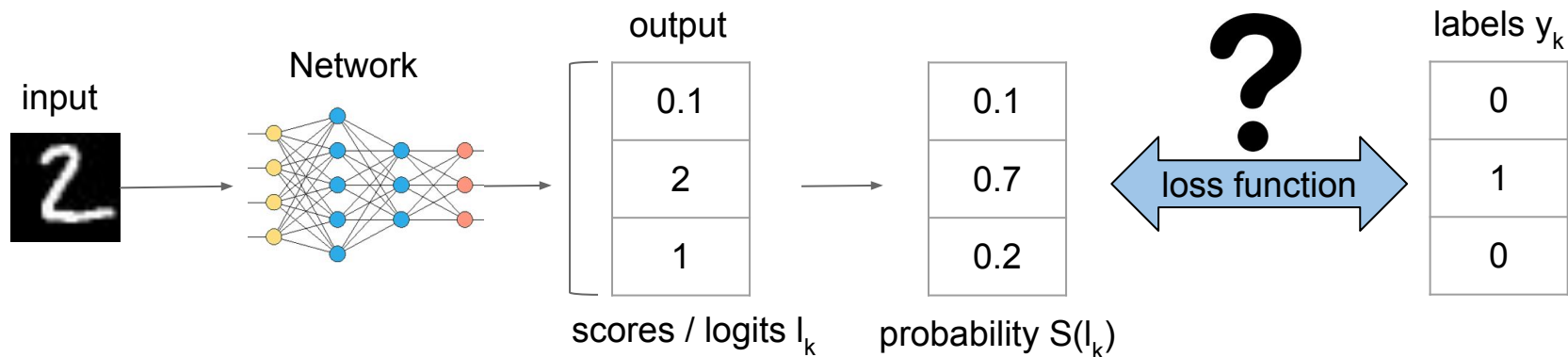


Softmax (3)



exponential function

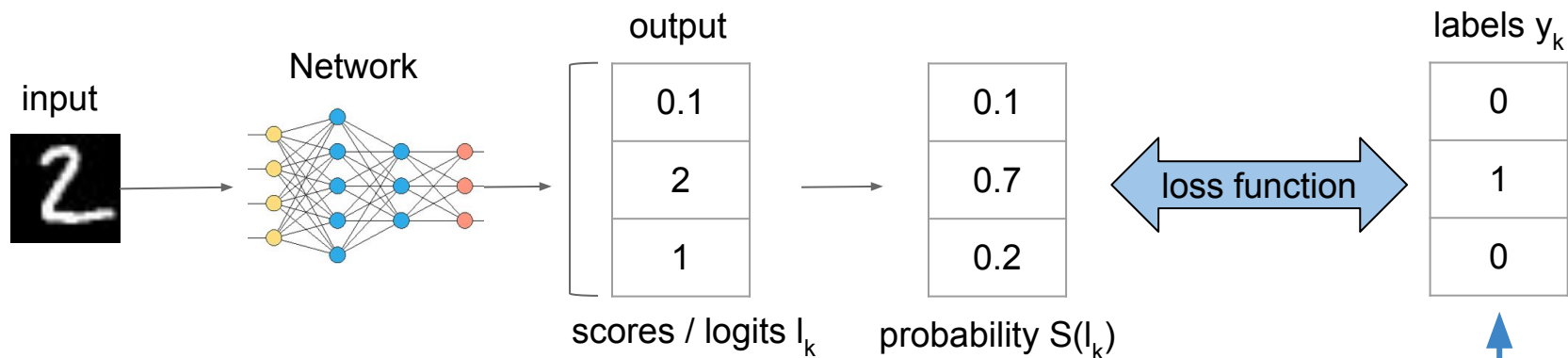
Cross-entropy loss (2)



Loss
contribution
from input x_i
in a training
batch

$$\mathcal{L}_i = - \sum_k y_k \log(S(l_k))$$

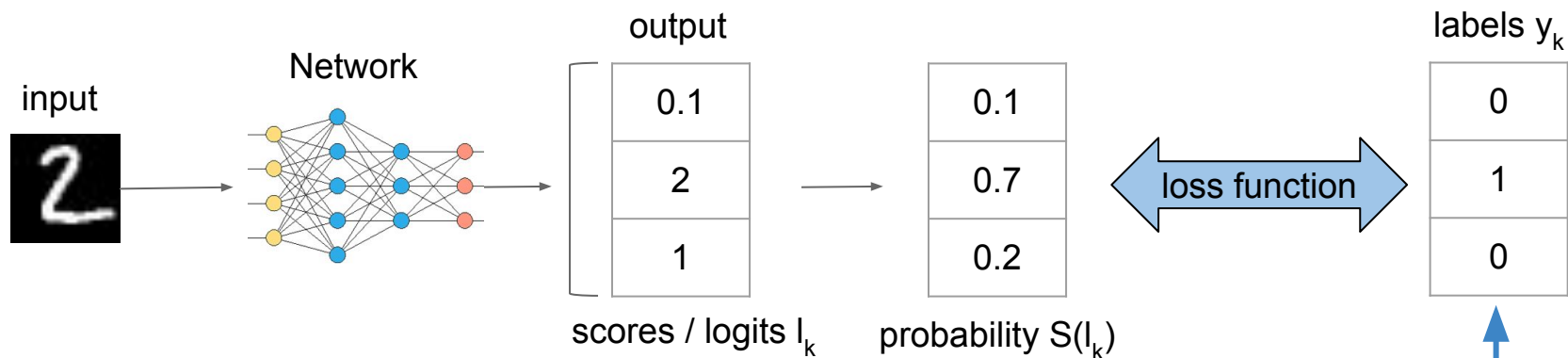
Cross-entropy loss (3)



$$\mathcal{L}_i = - \sum_k y_k \log(S(l_k))$$

In a one-hot vector, only one $y_k=1$ (rest are 0) !!

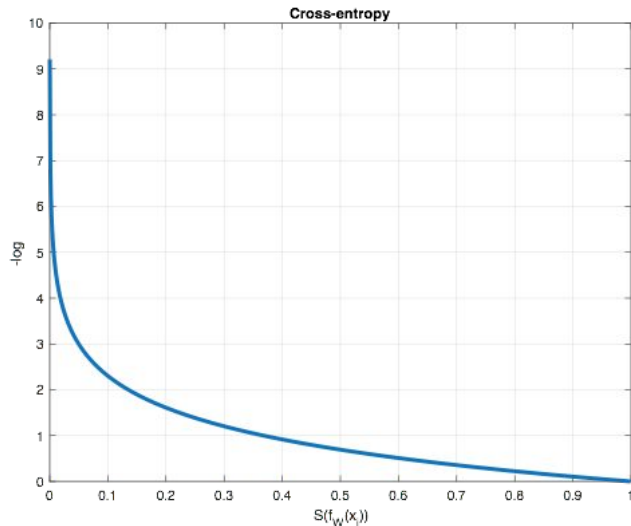
Cross-entropy loss (4)



$$\mathcal{L}_i = - \sum_k y_k \log(S(l_k)) = - \log(S(l))$$

Only the confidence for the GT class is considered.

Cross-entropy loss (5)




$$\mathcal{L}_i = - \sum_k y_k \log(S(l_k)) = -\log(S(l))$$


↑
-log penalizes small confidence scores for GT class

Cross-entropy loss (6)

- For a set of n inputs $\mathcal{L} = \frac{1}{n} \sum_{i=1}^n \mathcal{L}_i$

$$\mathcal{L} = - \sum_{i=1}^n \mathbf{y}_i \log(S(f_{\theta}(\mathbf{x}_i)))$$

labels (one-hot) 

Softmax 

Cross-entropy loss (7)

$$\mathcal{L} = - \sum_{i=1}^n y_i \log(S(f_{\theta}(\mathbf{x}_i)))$$

Confidences $S(l_k)$

0.1	0.3	0.3
0.2	0.4	0.3
0.7	0.3	0.4

Classification accuracy = $2/3 = 0.66$
cross-entropy loss = 1.3784

Ground Truth

1	0	0
0	1	0
0	0	1

Predictions

0	0	0
0	1	0
1	0	1

Classification accuracy = $2/3 = 0.66$
cross-entropy loss = 0.6391

Confidences $S(l_k)$

0.3	0.1	0.1
0.3	0.7	0.2
0.4	0.2	0.7

Weighted cross-entropy

- Dealing with class imbalance
 - Different number of samples for different classes
 - Introduce a weighing factor (related to inverse class frequency or treated as hyperparameter)



class “ant”



class “flamingo”

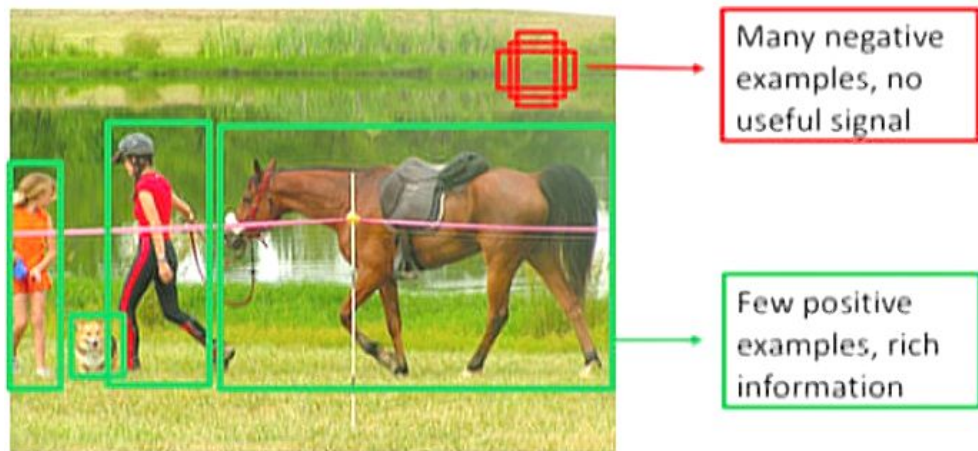


class “panda”

$$\mathcal{L}_i = -\alpha_i \log(S(f_{\theta}(\mathbf{x}_i)))$$

Focal loss (1)

- Deals with hard vs. easy examples
 - Very useful in one-shot object detection where there is extreme imbalance between classes
 - Easy examples (background) will have a lower contribution to the loss
 - Hard/Difficult examples (foreground) will contribute more

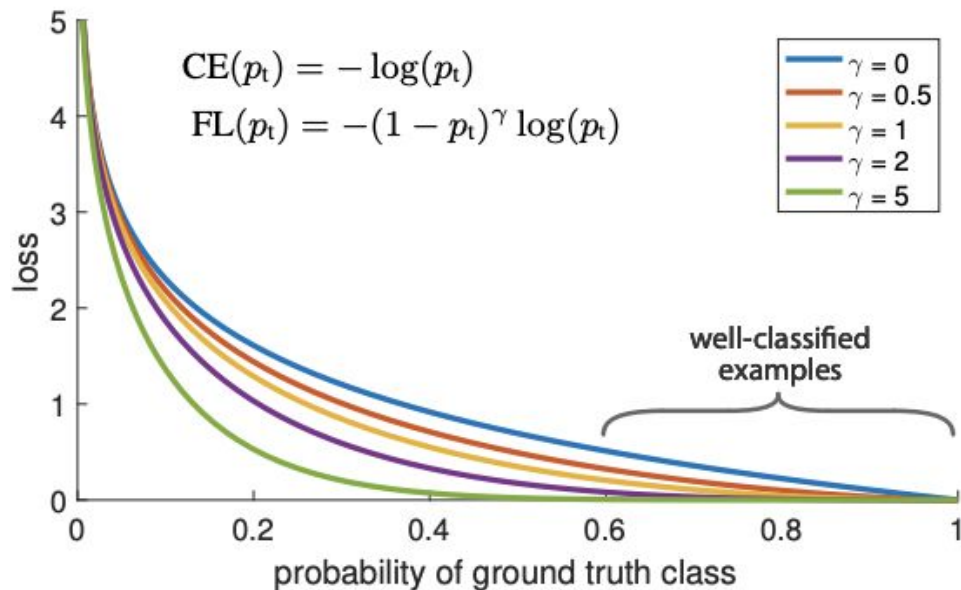


- Introduce a factor related to the **estimated** probability

Focal loss (2)

estimated probability of
sample \mathbf{x}_i

$$\mathcal{L}_i = -(1 - S(f_\theta(\mathbf{x}_i)))^\gamma \log(S(f_\theta(\mathbf{x}_i)))$$



Multi-label classification (1)

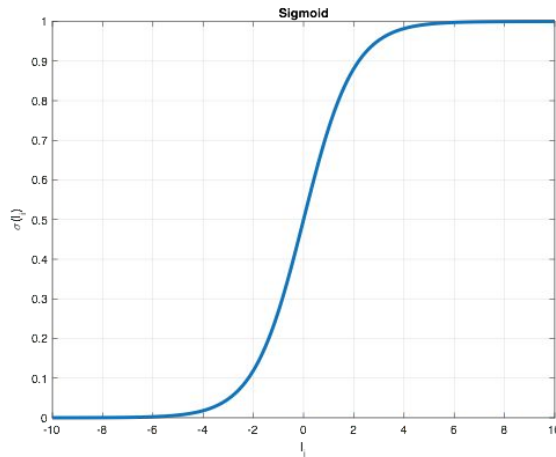
- Outputs can be matched to more than one label
 - “car”, “automobile”, “motor vehicle” can be applied to a same image of a car.

Images			
Labels	tree water black picture drawing sea art blue boat green city	man woman people hair girl picture smile group photo kid family	sky tree water white house window wood sea ocean cloud blue door

Multi-label classification (2)

- Outputs can be matched to more than one label
 - “car”, “automobile”, “motor vehicle” can be applied to a same image of a car.
- .. directly use sigmoid at each output independently instead of softmax

$$\sigma(l_i) = \frac{1}{1 + e^{-l_i}}$$



Cross-entropy for multi-label classification (3)

- Sigmoid + Binary cross-entropy

$$\mathcal{L}_i = - \sum_k y_k \log(\sigma(l_i)) + (1 - y_k) \log(1 - \sigma(l_i))$$

Summary

- Loss functions **measure** the **distance / error** of a particular set of parameters in the network
- Loss functions **drive** the **learning process**
- Regression
 - Absolute loss, **square loss**, huber loss
- Classification
 - Hinge loss, **cross-entropy loss**, focal loss

References

- [About loss functions](#)
- [Neural networks and deep learning](#)
- [Are loss functions all the same?](#)
- [Convolutional neural networks for Visual Recognition](#)
- [Deep learning book, MIT Press, 2016](#)
- [On Loss Functions for Deep Neural Networks in Classification](#)

Thanks! Questions?



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

**Department of Signal Theory
and Communications**

Image Processing Group

<https://imatge.upc.edu/web/people/javier-ruiz-hidalgo>