[course site]

#DLUPC

Day 4 Lecture 1

Loss functions

Javier Ruiz Hidalgo
javier.ruiz@upc.edu

Associate Professor
Universitat Politecnica 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

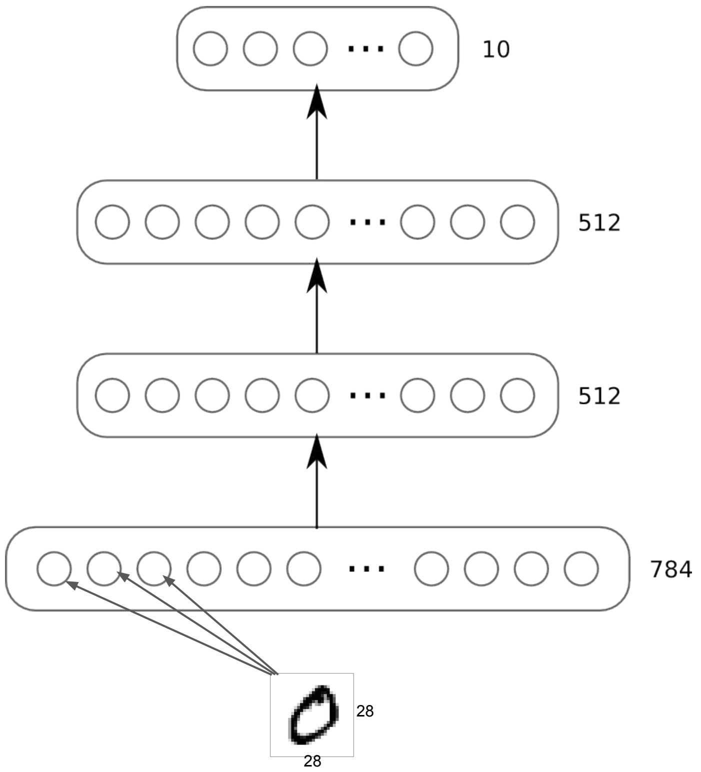
Motivation

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
  - Metric learning
- Example

Nomenclature

loss function

=

cost function

=

objective function

=

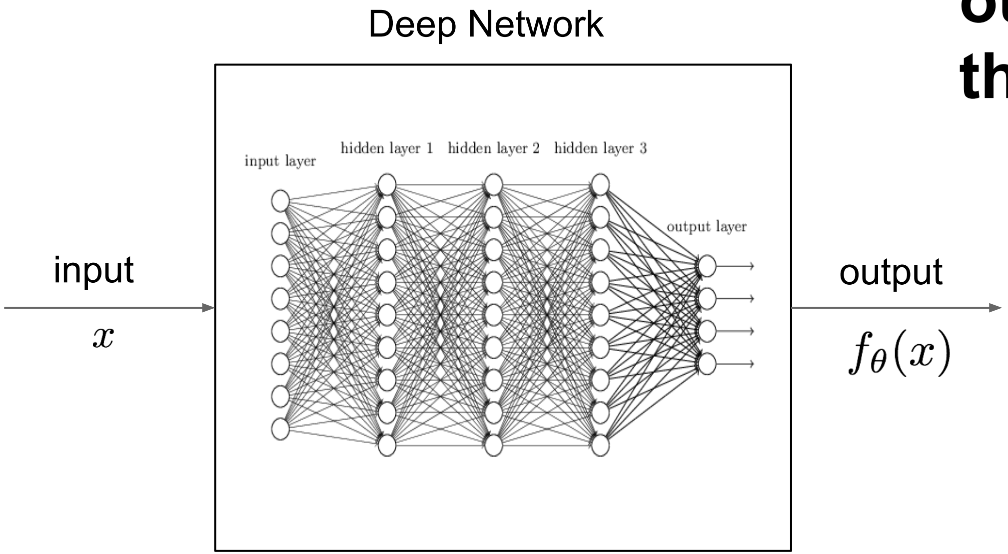
error function

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

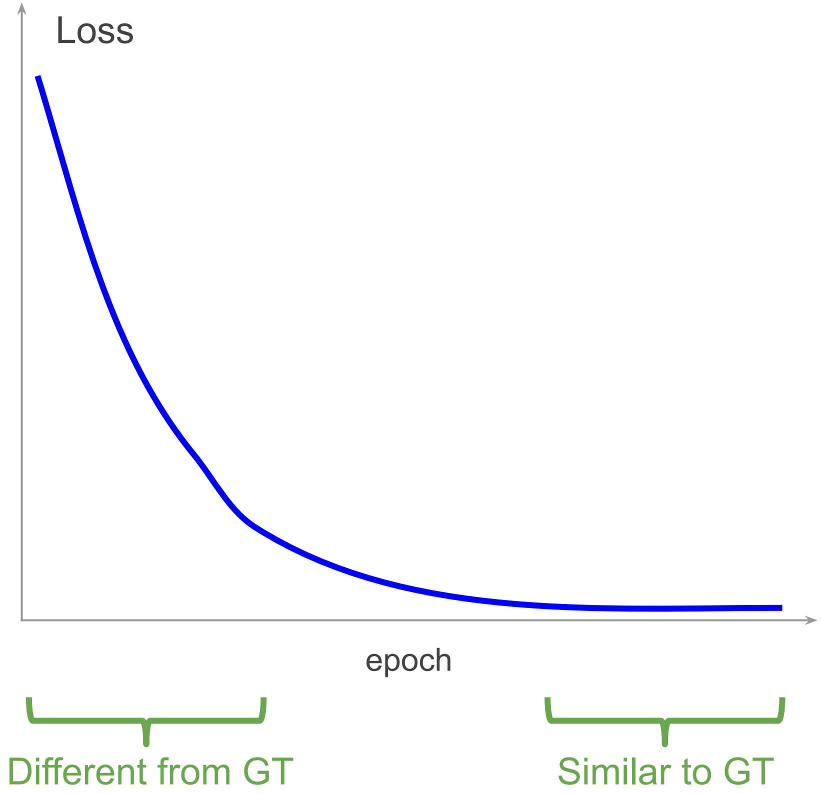
Stochastic gradient descent

- 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:

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

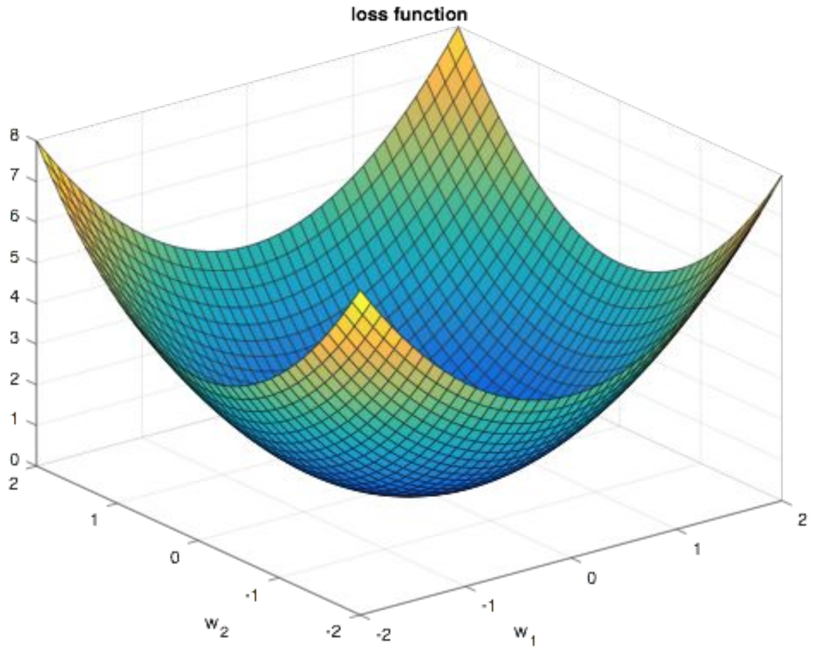
Properties (1)

- 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 → convex function

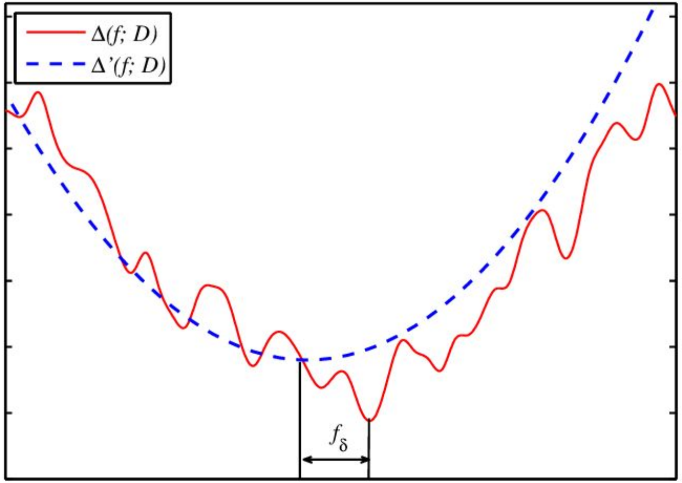


Properties (3)

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

Properties (4)

- 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



Assumptions

- For **backpropagation** to work:
  - Loss function can be written as an average over loss functions for individual training examples:

empirical risk

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_i$$

  - Loss functions can be written as a function of the output activations from the neural network.

Outline

- Introduction
  - Definition, properties, training process
- **Common types of loss functions**
  - **Regression**
  - Classification
  - Metric learning
- Example

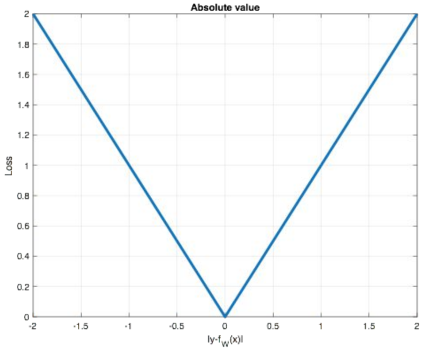
Loss functions for regression (1)

- Loss functions depend on the type of task:
    - Regression: the network predicts **continuous**, **numeric** variables
        - Example: Length of fishes in images, price of a house
        - Absolute error, square error, Huber error

Loss functions for regression (2)

L1 Distance /
Absolute Value

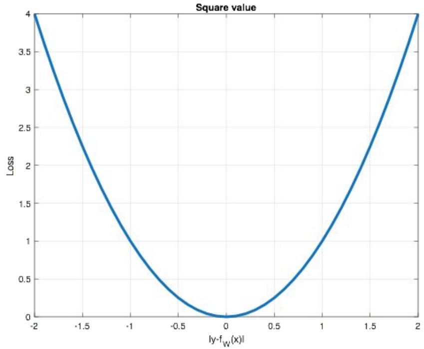
Lighter computation
(only |·|)



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

L2 Distance /
Euclidian Distance / Square Value

Heavier computation
(square)



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

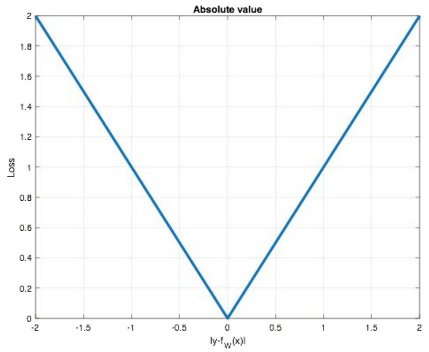
Loss functions for regression (3)

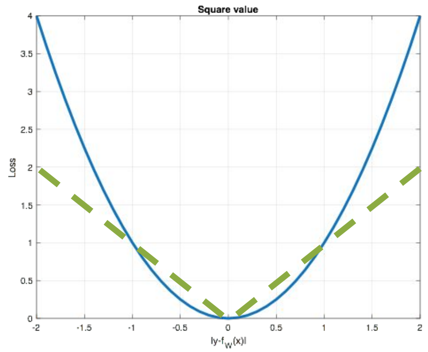
L1 Distance /
Absolute Value

L2 Distance /
Euclidian Distance / Square Value

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

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





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

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

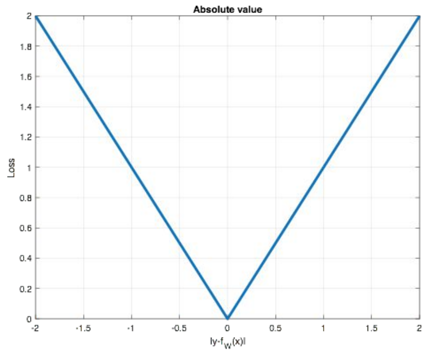
Loss functions for regression (4)

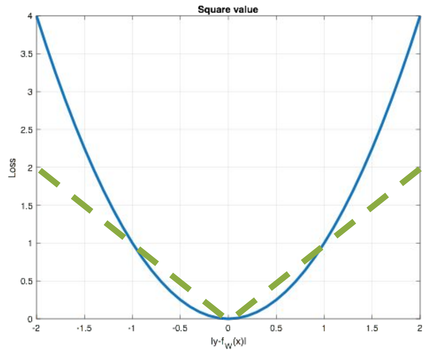
L1 Distance /
Absolute Value

L2 Distance /
Euclidian Distance / Square Value

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

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



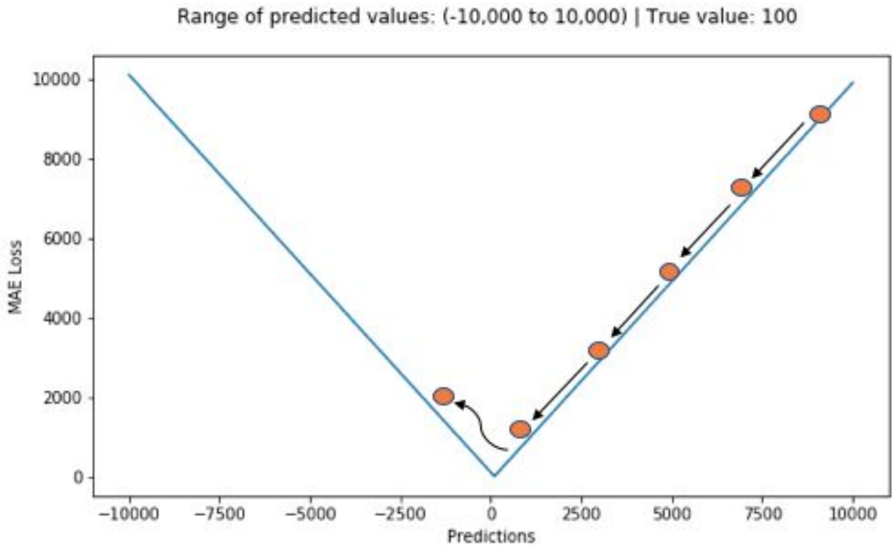


$$\mathcal{L}_i = |y_i - f_\theta(x_i)|$$

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

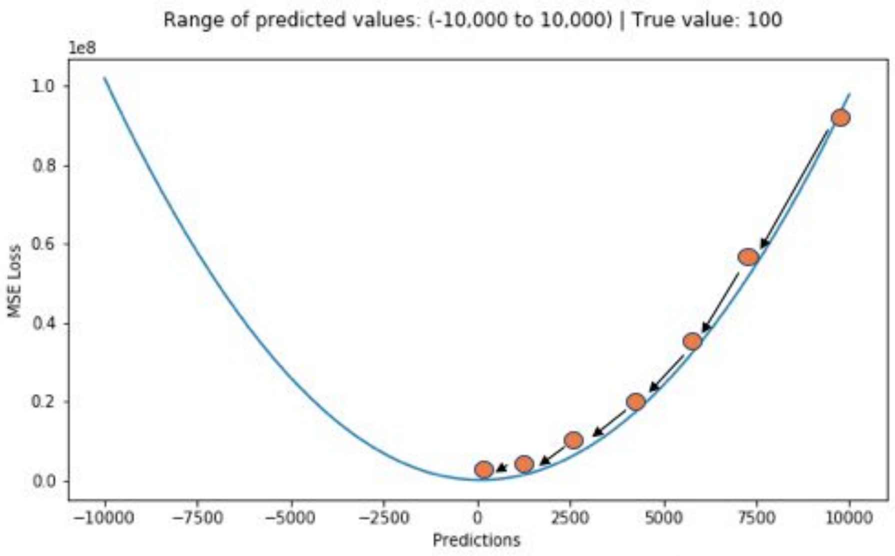
Loss functions for regression (5)

L1 Distance /
Absolute Value



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

L2 Distance /
Euclidian Distance / Square Value



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

[Choosing the right loss function for fitting a model, Prince Grover]

Loss functions for regression (6)

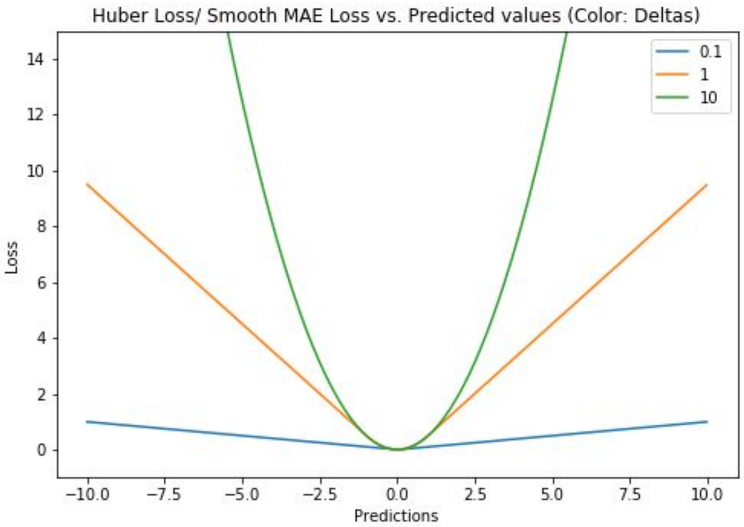
- L1 / absolute loss
  - ❎ More robust to outliers
  - ❌ Large gradients for small loss → Inefficient to find minimum
- L2 / square loss
  - ❌ Sensitive to outliers
  - ❎ More stable and closed form solution (derivatives go to 0)
- In general, **L2 loss is more appropriate**

- 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 δ=1)

Choosing the right loss function for fitting a model, Prince Grover

Outline

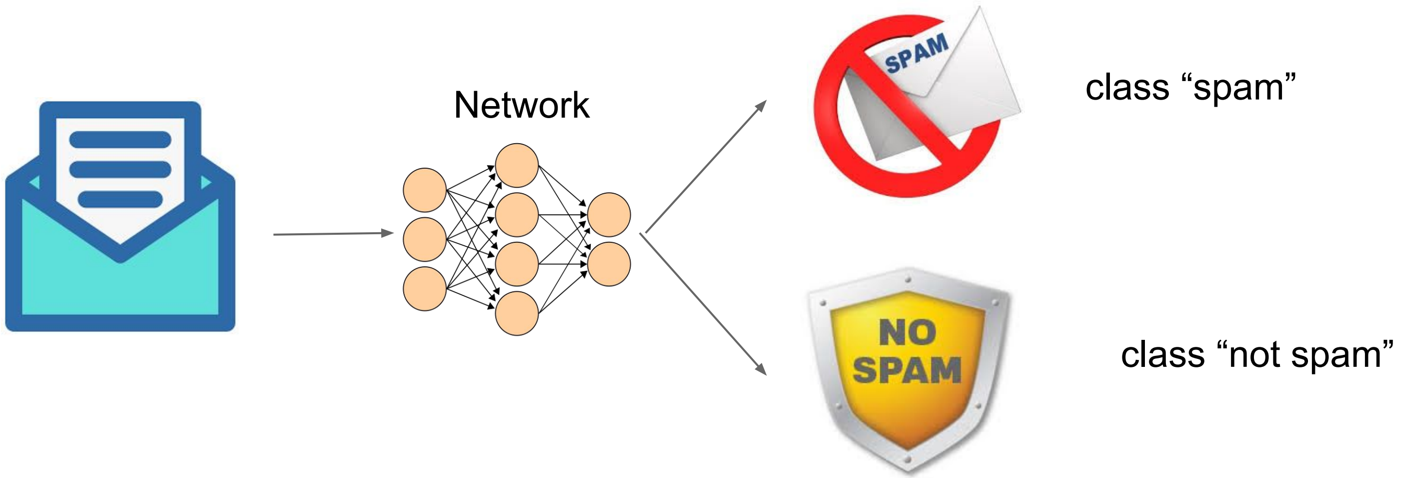
- Introduction
  - Definition, properties, training process
- **Common types of loss functions**
  - Regression
  - **Classification**
  - Metric learning
- Example

Loss functions for classification

- Loss functions depend on the type of task:
  - Classification: the network predicts **categorical** variables (fixed number of classes)
    - Example
      - Classify email as spam (binary classification)
      - Classify images of numbers (single-label multiclass classification)
      - Describe images with labels (multi-label classification)
    - Hinge loss, Cross-entropy loss, Focal loss, ...

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

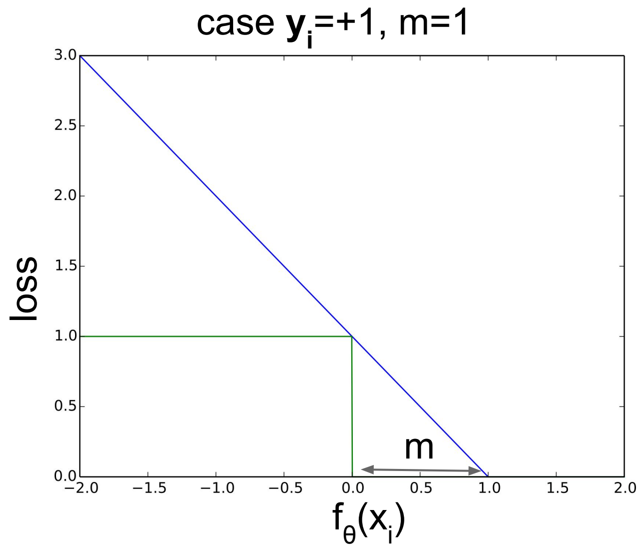
$$\mathcal{L}_i = (y_i - f_\theta(x_i))^2$$

  - 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 $\mathbf{y_i} = \pm 1$, **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

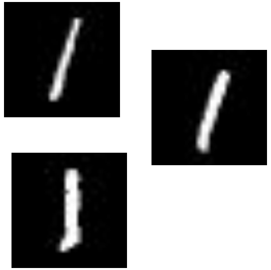


Loss functions for classification

- 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



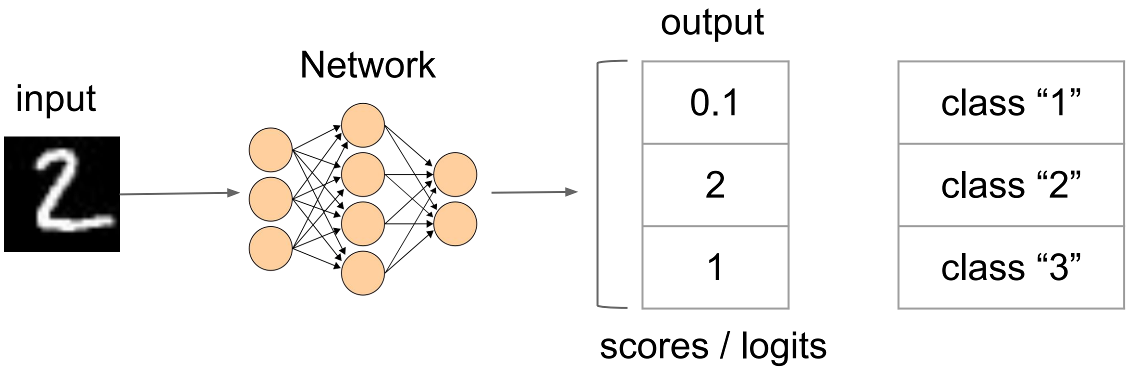
class “1”

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**

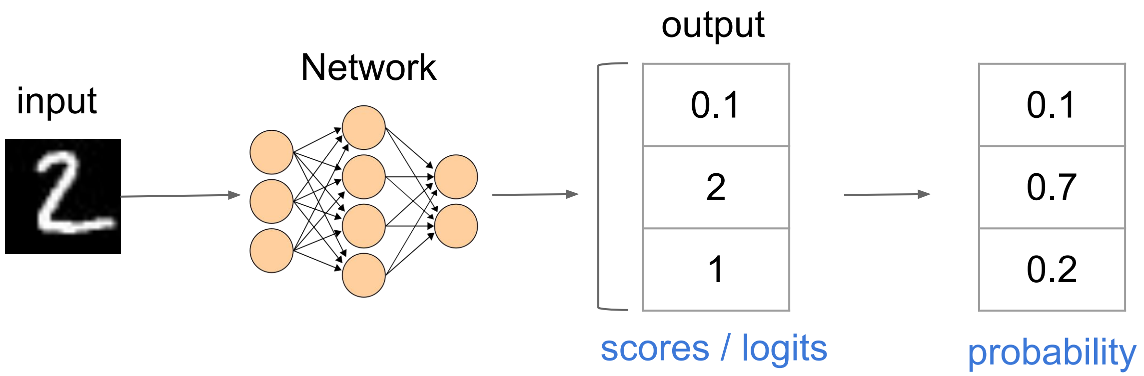
One-hot encoding

- 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”	class “2”	class “3”
1	0	0
0	1	0
0	0	1

Softmax (1)

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

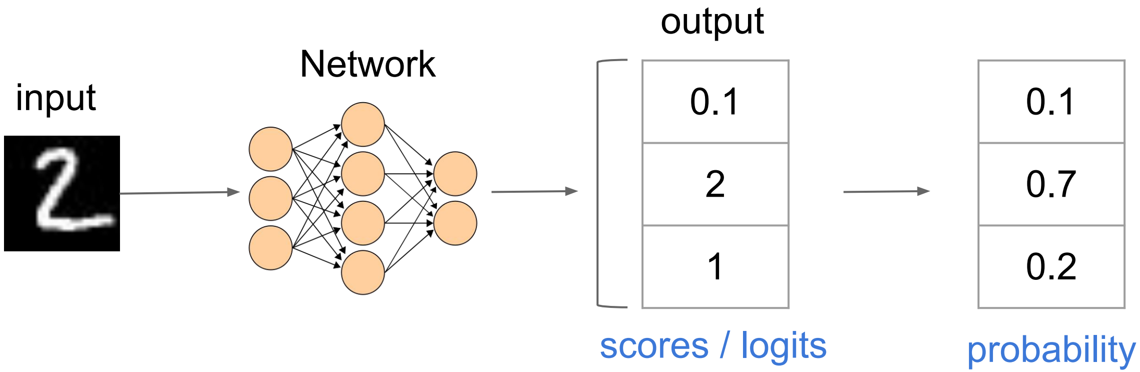


Softmax (2)

- Softmax function

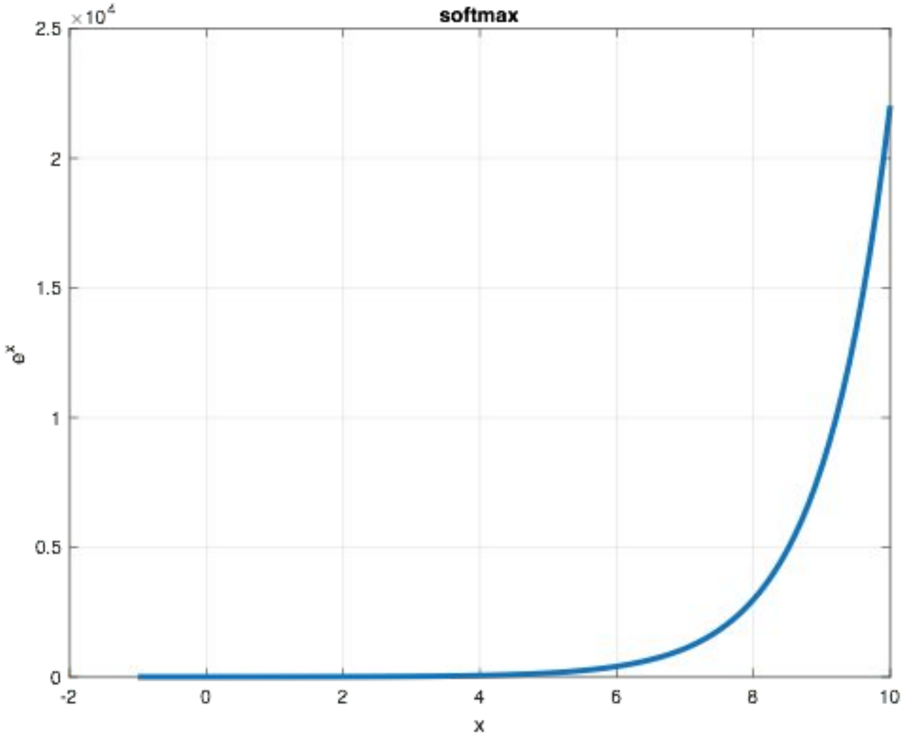
scores (logits)

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



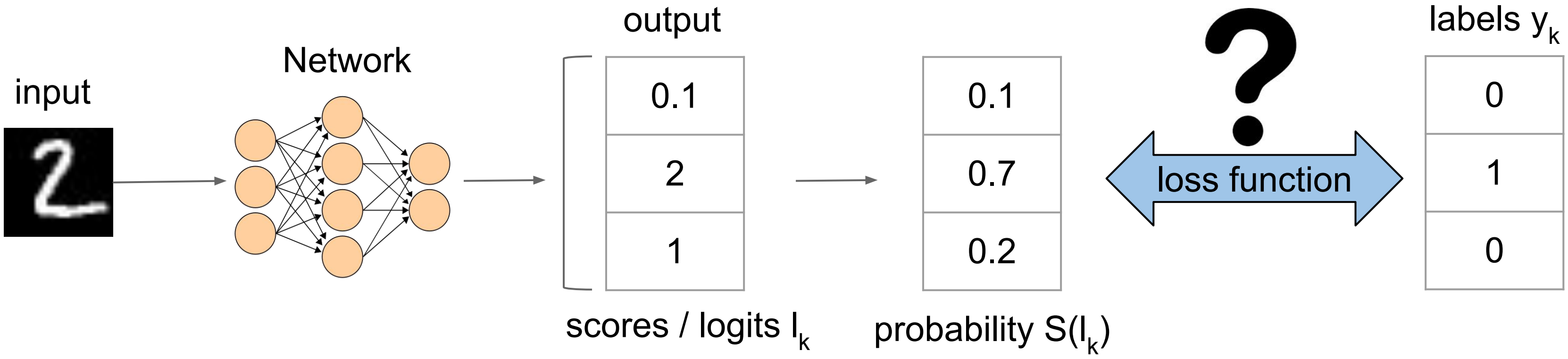
Neural Networks and Deep Learning (softmax)

Softmax (3)



exponential function

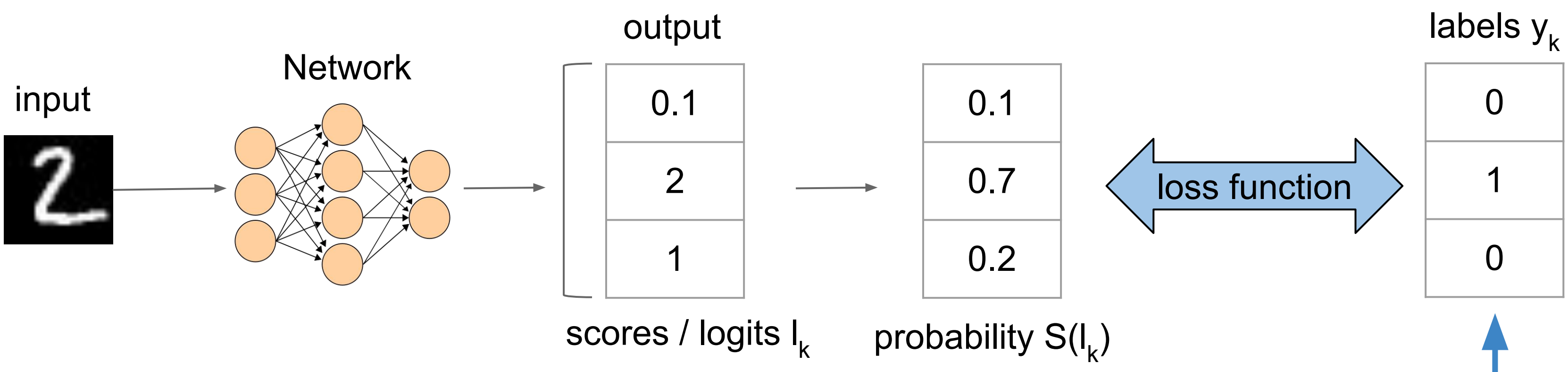
Cross-entropy loss (2)



Loss contribution from input x_i in a training batch

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

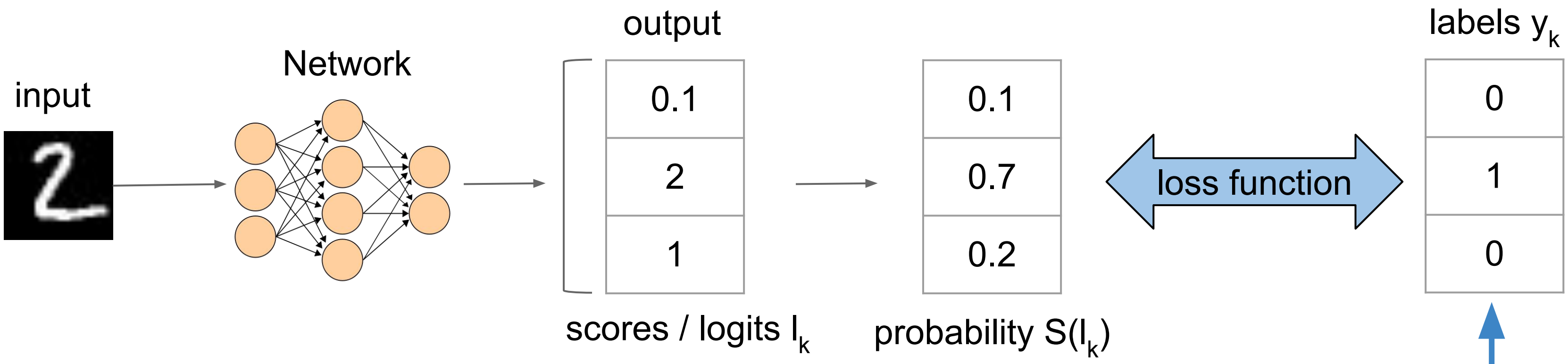
Cross-entropy loss (3)



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

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

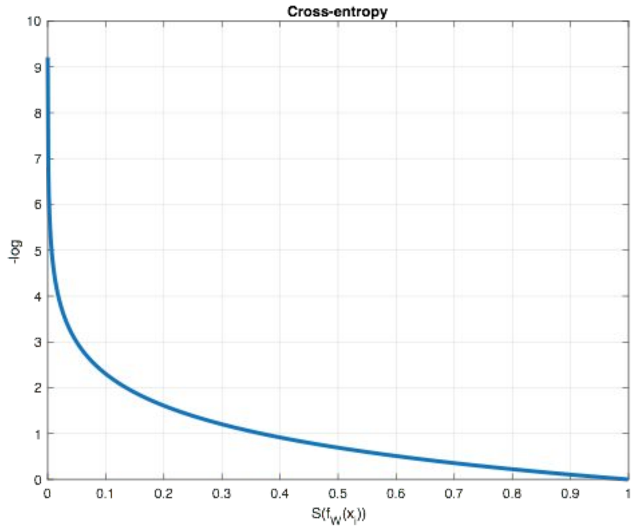
Cross-entropy loss (4)



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

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

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_i$$

labels (one-hot)

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

Softmax

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"

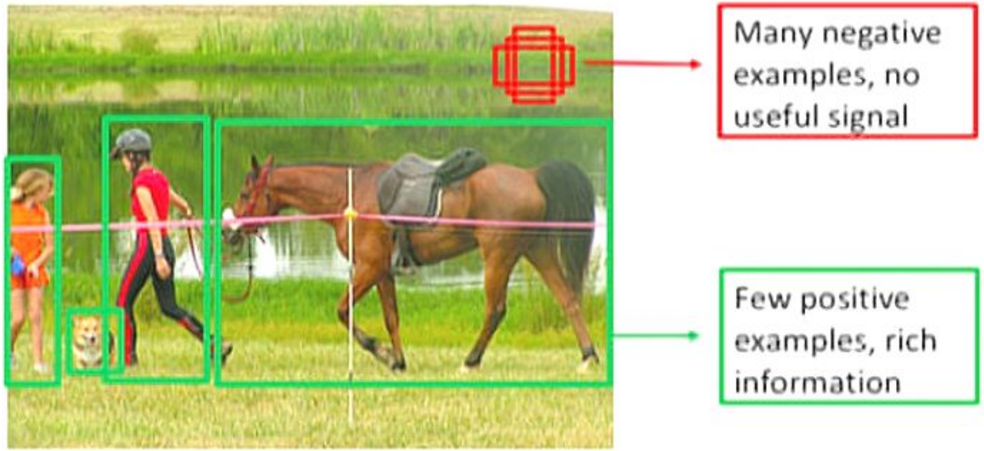
class "flamingo"

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



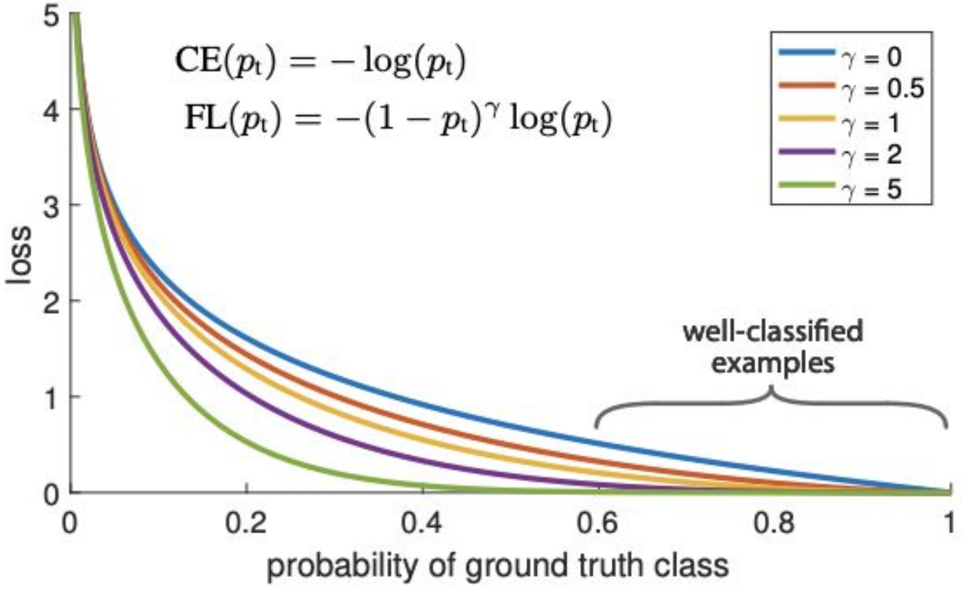
  - Introduce a factor related to the **estimated** probability

T. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, ["Focal Loss for Dense Object Detection"](), 2018.

Focal loss (2)

estimated probability of sample x_i

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



T. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, ["Focal Loss for Dense Object Detection"](), 2018.

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.

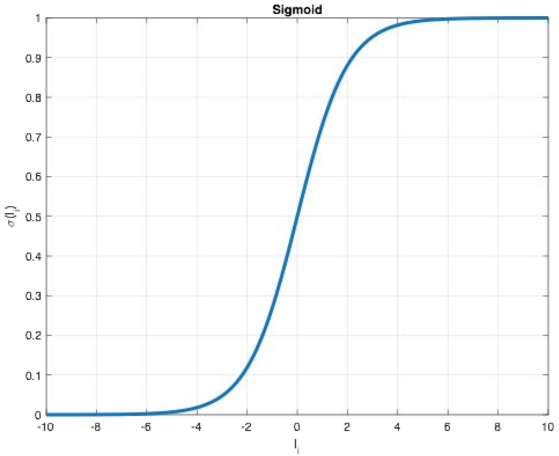
Images			
Labels	tree water black picture drawing sea art blue boat green city	man woman people hair girl picture smile group photo kid family	sky tree water white house window wood sea ocean cloud blue door

Bucak, Serhat Selcuk, Rong Jin, and Anil K. Jain. "Multi-label learning with incomplete class assignments." CVPR 2011.

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

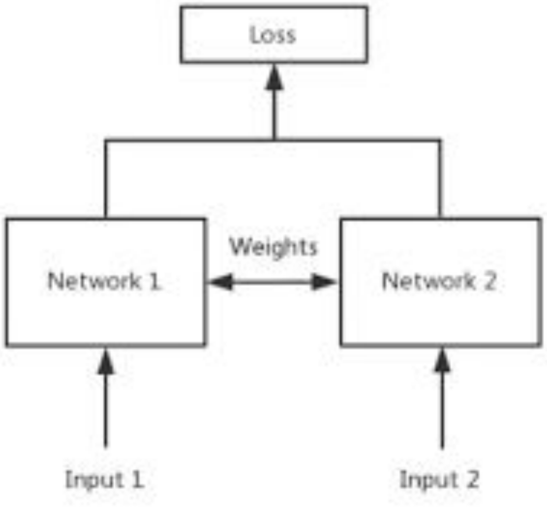
$$\mathcal{L}_i = -\sum_k y_k \log(\sigma(l_i)) + (1 - y_k) \log(1 - \sigma(l_i))$$

Outline

- Introduction
  - Definition, properties, training process
- Common types of loss functions
  - Regression
  - Classification
  - **Metric learning**
- Example

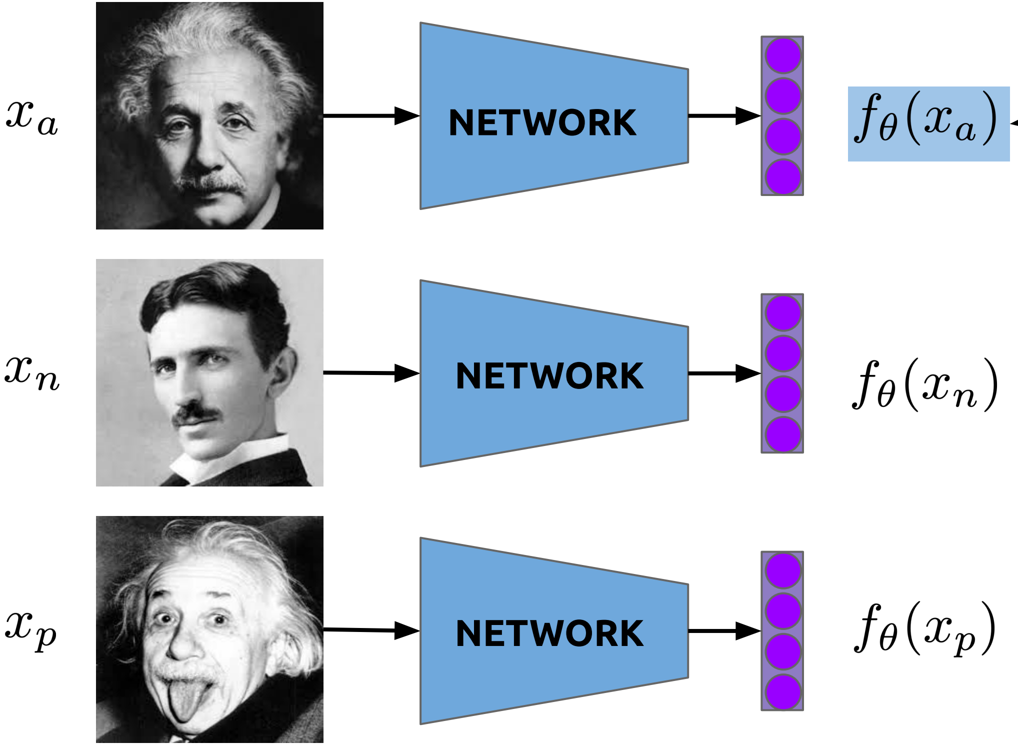
Metric learning (1)

- We want the network to generate encodings (embeddings) of the input to learn distances on the data
- Given two inputs x1 and x2 we want the encodings to be close if similar and further apart if dissimilar
  - **Siamese networks**



Siamese Network (1)

- Siamese networks in the context of face recognition

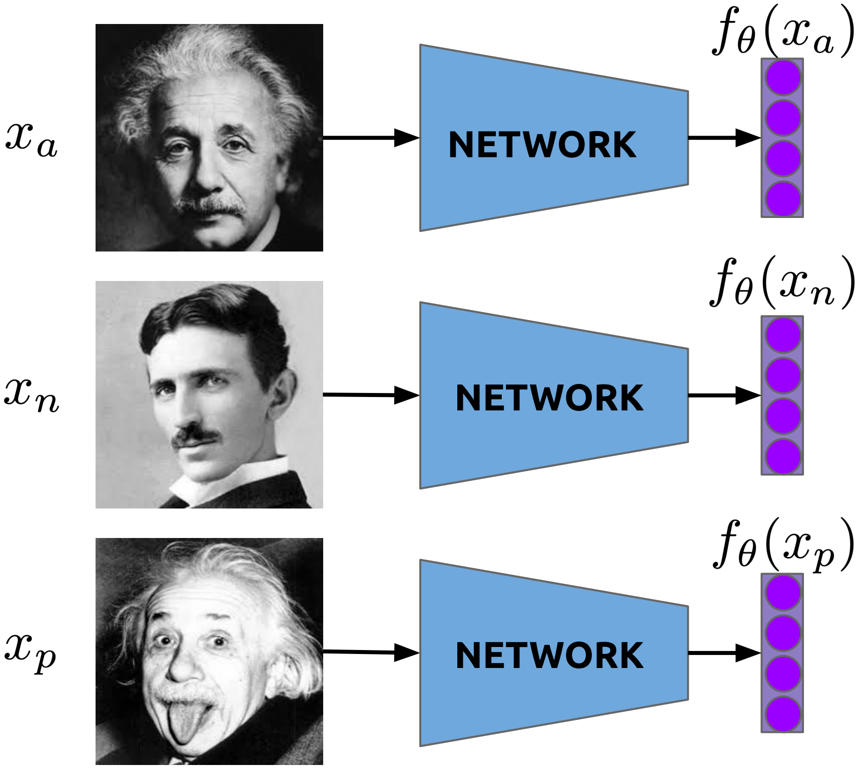


encoding of x_a
embedding
feature vector
descriptor

We want the embeddings of x_a and x_p to be close and the embeddings of x_a and x_n to be farther away!

Siamese Network (2)

- Siamese networks in the context of face recognition



Distances (square error) between **negative pairs** to be **large**

$$d^2(a, n) = ||f_\theta(x_a) - f_\theta(x_n)||^2$$

Distances (square error) between **positive pairs** to be **small**

$$d^2(a, p) = ||f_\theta(x_a) - f_\theta(x_p)||^2$$

Contrastive loss

- Input pairs (x_a, x_b) are fed into the network during training
  - y=+1 if inputs are similar: positive pair (x_a, x_p)
  - y=-1 if inputs otherwise: negative pair (x_a, x_n)

$$\mathcal{L}_i = y_i d^2 + (1 - y_i) max(m - d, 0)^2$$

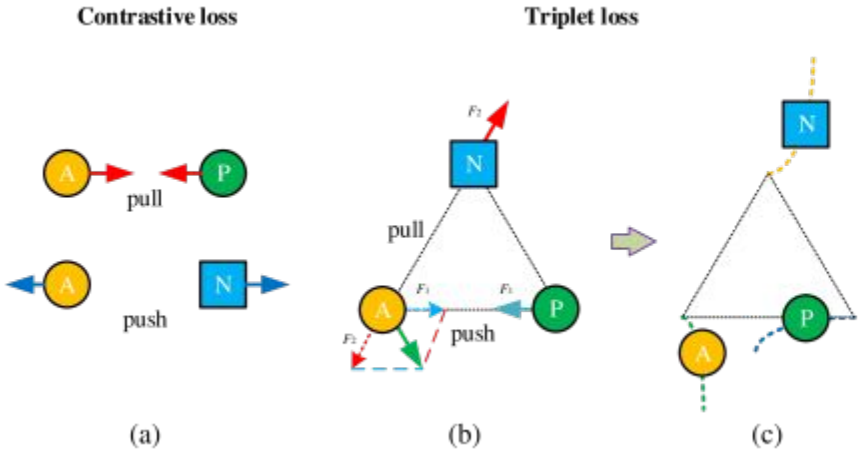
- m is the margin to “tighten” the constraint
  - negative pairs should have distances of at least m

Raia Hadsell, Sumit Chopra, Yann LeCun, Dimensionality reduction by learning an invariant mapping, CVPR 2006

Triplet loss

- Learn embeddings that **jointly** put anchors closer to positive examples than negative examples

$$\mathcal{L}_i = max(m + d^2(a,p) - d^2(a,n), 0)$$

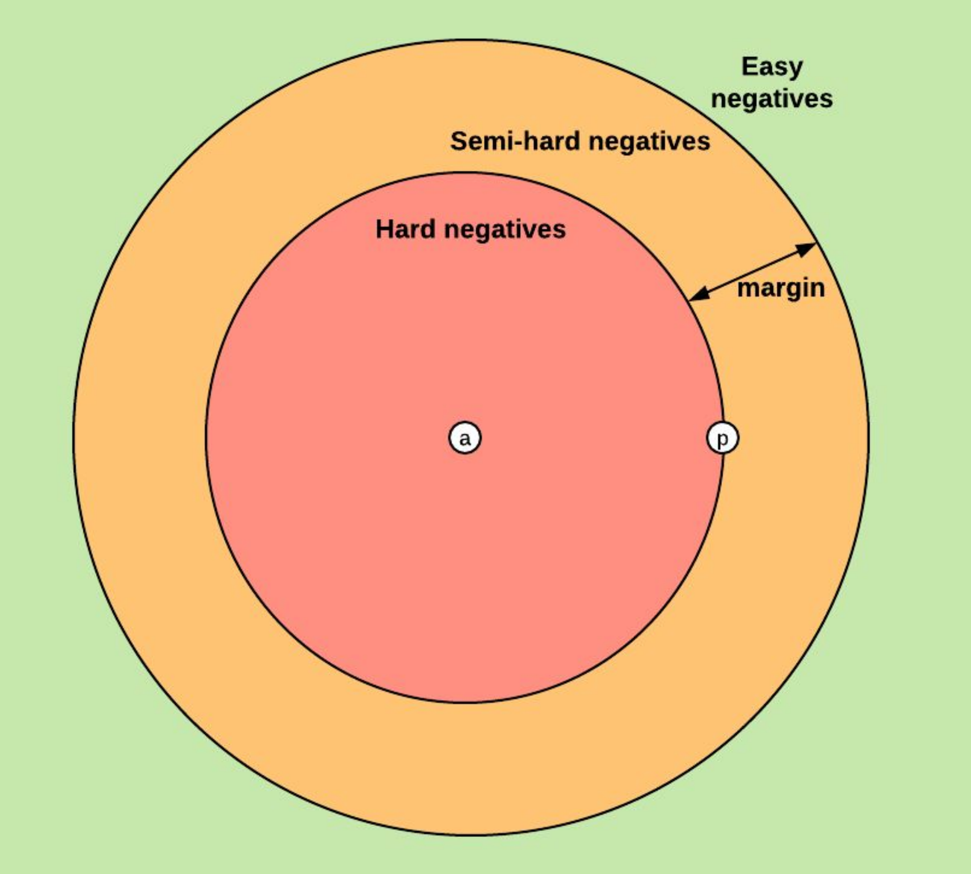


F. Schroff, Florian, D. Kalenichenko, J. Philbin. [Facenet: A unified embedding for face recognition and clustering](). CVPR 2015

Pair/triplet mining (1)

- Number of pairs/triplets grows quadratically/cubically → **strategy needs to be chosen**
- As training goes, more pairs/triplets are easy (loss equal to 0) which prevents network from training → **network needs hard examples**

Pair/triplet mining (2)



- For each anchor + positive pair, select a negative sample which is hard or semi-hard

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
- Metric learning
  - Contrastive loss, **triplet loss**

Outline

- Introduction
  - Definition, properties, training process
- Common types of loss functions
  - Regression
  - Classification
  - Metric learning
- **Example of cross-entropy loss**

Example

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

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 = 4.14

Ground Truth

1	0	0
0	1	0
0	0	1

Predictions

0	0	0
0	1	0
1	0	1

Confidences $S(l_k)$

0.3	0.1	0.1
0.3	**0.7**	0.2
0.4	0.2	**0.7**

Classification accuracy = 2/3 = 0.66
cross-entropy loss = 1.63

Why you should use cross-entropy instead of classification error

Example

a) Compute the accuracy of the predictions.

The class with the maximum confidence is predicted.

Confidences $S(l_k)$

0.1	0.3	0.3
0.2	**0.4**	0.3
0.7	0.3	**0.4**

Predictions

0	0	0
0	1	0
1	0	1

Ground Truth

1	0	0
0	1	0
0	0	1

[Why you should use cross-entropy instead of classification error]

Example

a) Compute the accuracy of the predictions.

Class predictions and ground truth can be compared with a confusion matrix.

Confidences $S(l_k)$

0.1	0.3	0.3
0.2	**0.4**	0.3
0.7	0.3	**0.4**

Predictions

0	0	0
0	1	0
1	0	1

Ground Truth

1	0	0
0	1	0
0	0	1

Accuracy =2 / 3 = 0.66

		Prediction		
		Class 1	Class 2	Class 3
GT	Class 1	0	0	1
	Class 2	0	1	0
	Class 3	0	0	1

Accuracy =2 / 3 = 0.66

Example

a) Compute the accuracy of the predictions
b) Compute the cross-entropy loss of the prediction

Confidences $S(l_k)$

0.1	0.3	0.3
0.2	0.4	0.3
0.7	0.3	0.4

Ground Truth y

1	0	0
0	1	0
0	0	1

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

[Why you should use cross-entropy instead of classification error]

Classification accuracy = 2/3
cross-entropy loss = 4.14

Example

a) Compute the accuracy of the predictions
b) Compute the cross-entropy loss of the prediction

Confidences $S(l_k)$

0.1	0.3	0.3
0.2	**0.4**	0.3
0.7	0.3	**0.4**

Ground Truth y

1	0	0
0	1	0
0	0	1

-log

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

2.3 0.92 0.92

[Why you should use cross-entropy instead of classification error]

Example

a) Compute the accuracy of the predictions
b) Compute the cross-entropy loss of the prediction

Confidences $S(l_k)$

0.1	0.3	0.3
0.2	**0.4**	0.3
0.7	0.3	**0.4**

Ground Truth y

1	0	0
0	1	0
0	0	1

-log

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

2.3 0.92 0.92

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_i$$

L=4.14

Why you should use cross-entropy instead of classification error

Assignment

a) Compute the accuracy of the predictions
b) Compute the cross-entropy loss of the prediction
c) Compare the result with the one obtained with the example and discuss how accuray and cross-entropy have captured the quality of the predictions.

Confidences $S(l_k)$

0.3	0.1	0.1
0.3	0.7	0.2
0.4	0.2	0.7

Ground Truth

1	0	0
0	1	0
0	0	1

[Why you should use cross-entropy instead of classification error]

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