[course site]

#DLUPC

Day 4 Lecture 2

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
  - 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

Outline

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

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.

Nomenclature

loss function

=

cost function

=

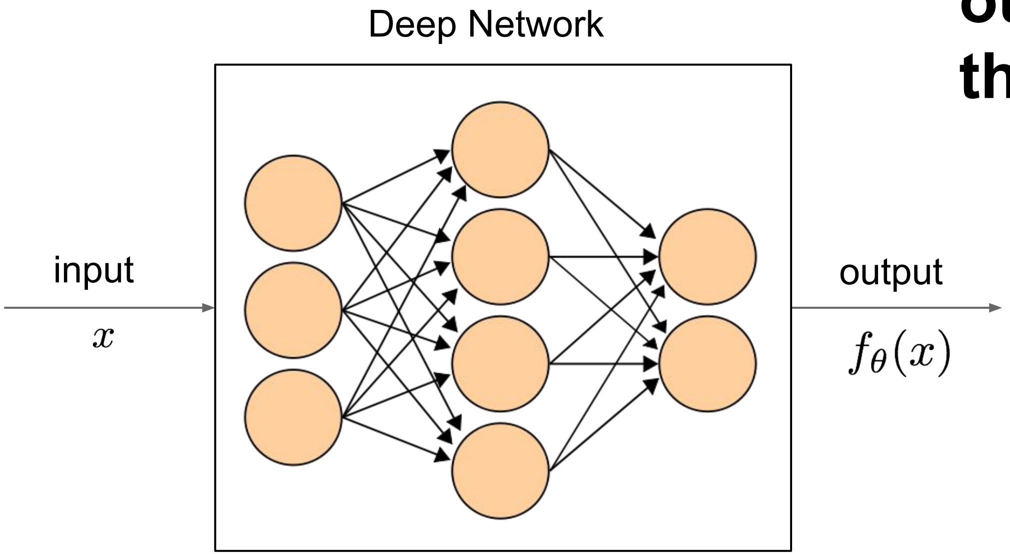
objective function

=

error function

Loss function (1)

How good does our network with the training data?



labels (ground truth)

input

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

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.
- 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:

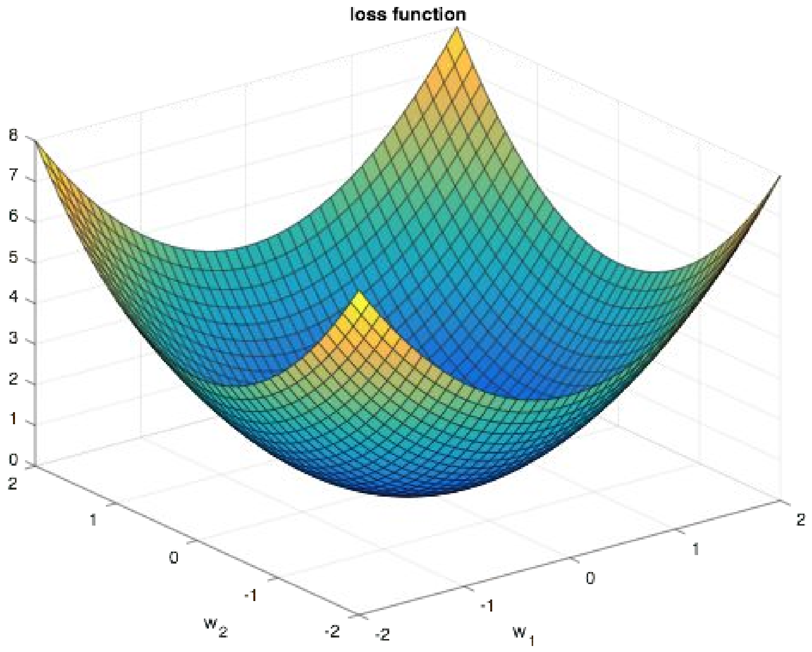
$$\frac{\partial \mathcal{L}}{\partial w} \quad \frac{\partial \mathcal{L}}{\partial b}$$

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

Common types of loss functions (1)

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

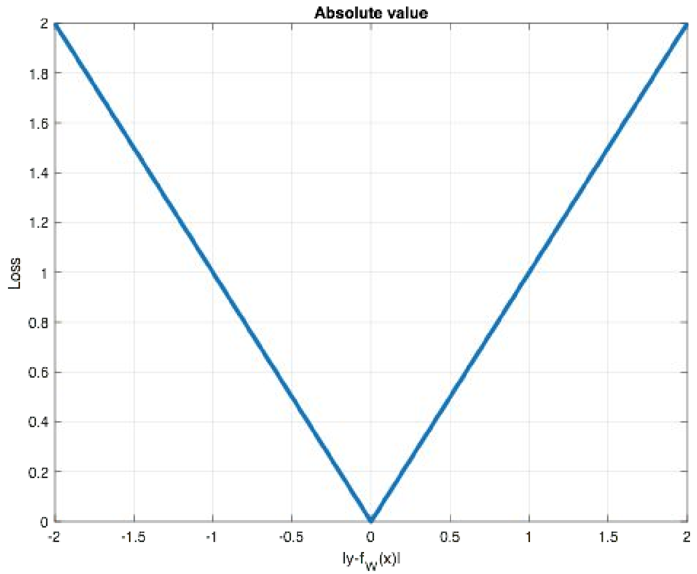
Common types of loss functions (2)

- Loss functions depen on the type of task:
  - Classification: the network predicts **categorical** variables (fixed number of classes)
    - Example: classify email as spam, predict student grades from essays.
    - hinge loss, Cross-entropy loss

Absolute value, L1-norm

- Very intuitive loss function
  - produces sparser solutions
    - good in high dimensional spaces
    - prediction speed
  - less sensitive to outliers

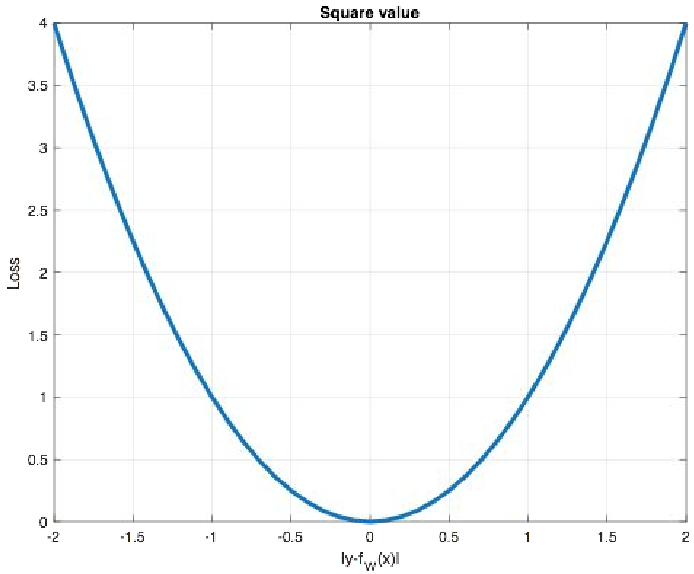
$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} |y_i - f_\theta(x_i)|$$



Square error, Euclidean loss, L2-norm

- Very common loss function
  - More precise and better than L1-norm
  - Penalizes large errors more strongly
  - Sensitive to outliers

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

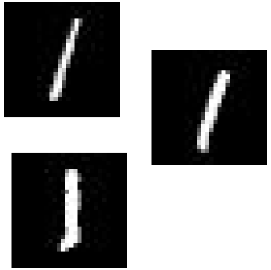


Outline

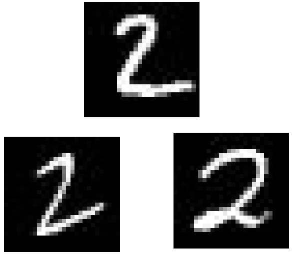
- Introduction
  - Definition, properties, training process
- **Common types of loss functions**
  - Regression
  - **Classification**
- Regularization
- Example

Classification (1)

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



class “1”

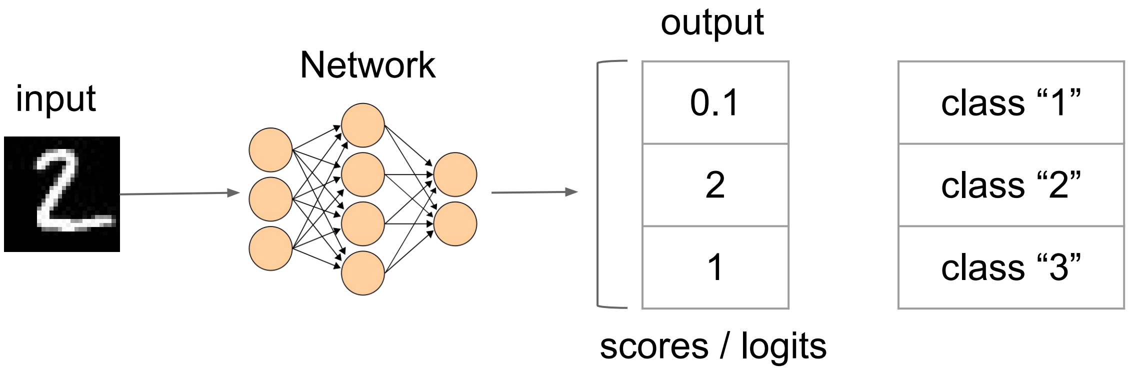


class “2”

class “3”

Classification (2)

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

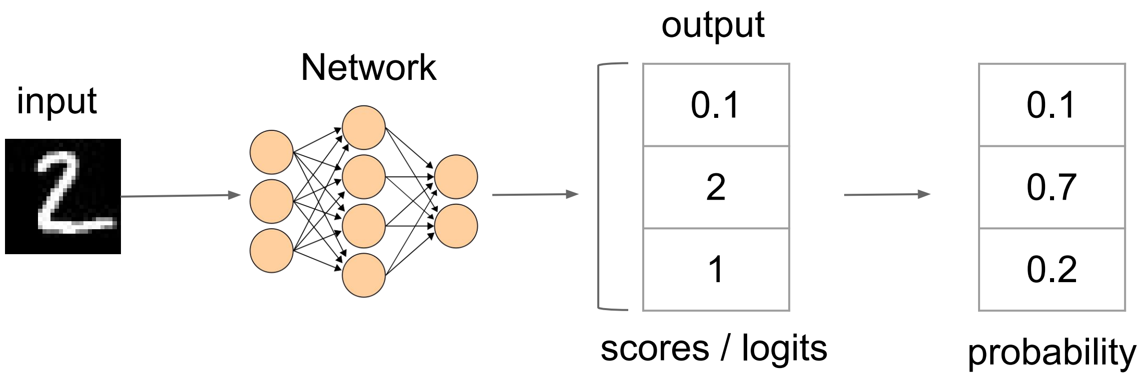


Classification (3)

- How can we create a loss function to improve the scores?
  - Somehow write the labels (ground truth of the data) into a vector → One-hot encoding
  - Non-probabilistic interpretation → **hinge loss**
  - Probabilistic interpretation: need to transform the scores into a probability function → Softmax

Softmax (1)

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

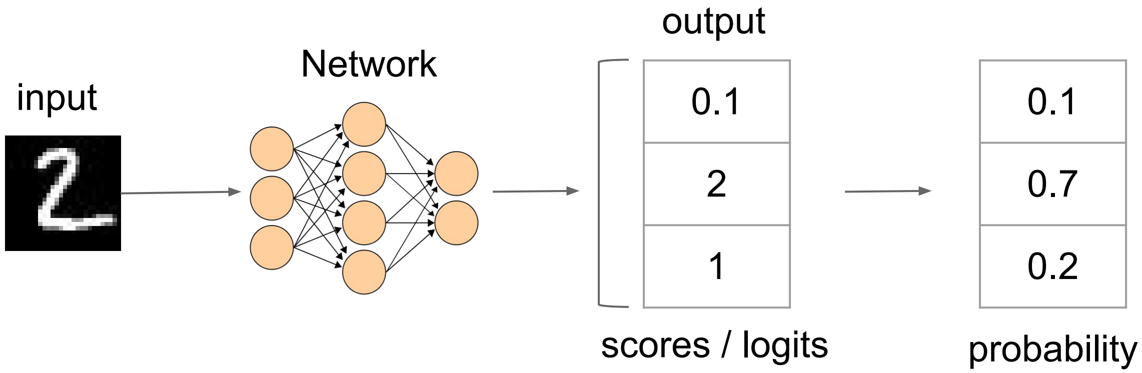


Softmax (2)

- Softmax function

scores (logits)

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

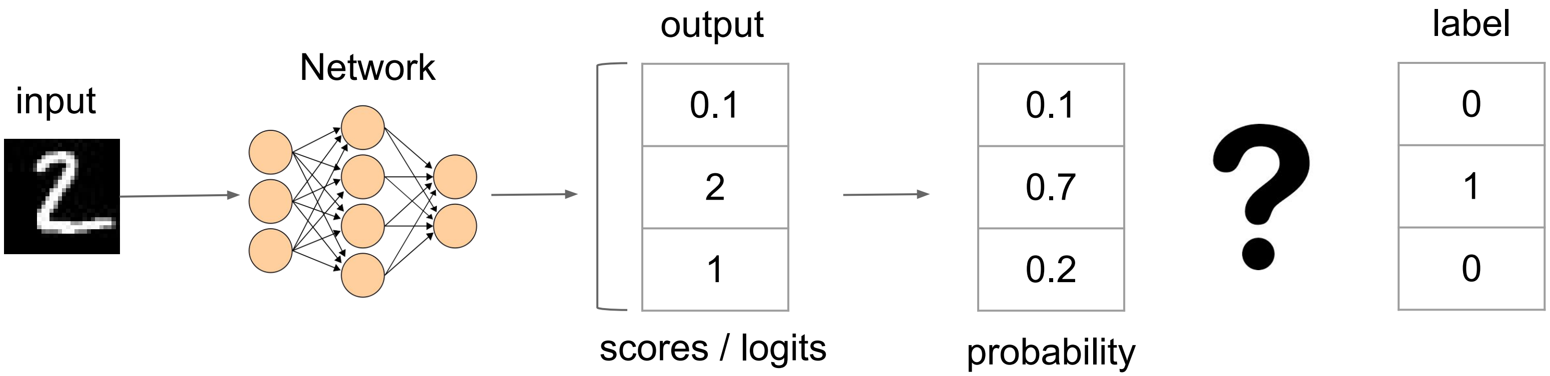


One-hot encoding

- 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

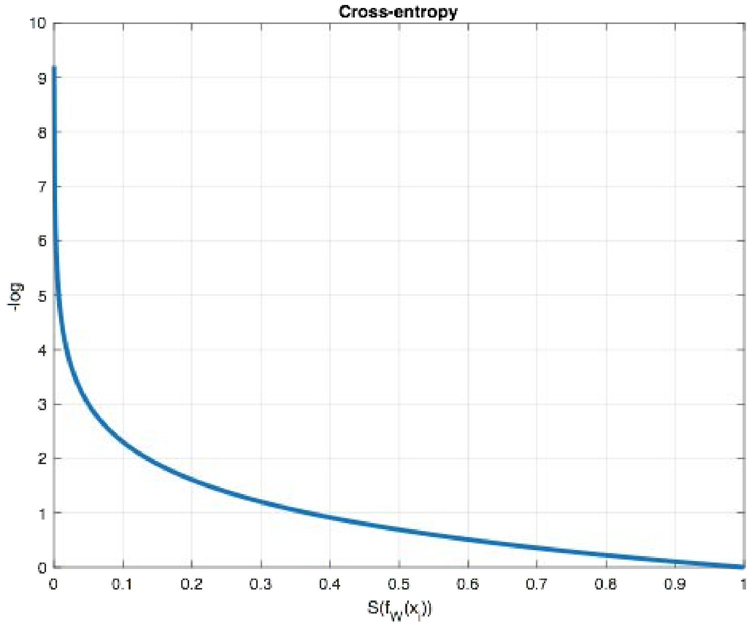
Cross-entropy loss (1)



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

Cross-entropy loss (2)

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



Cross-entropy loss (3)

- 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

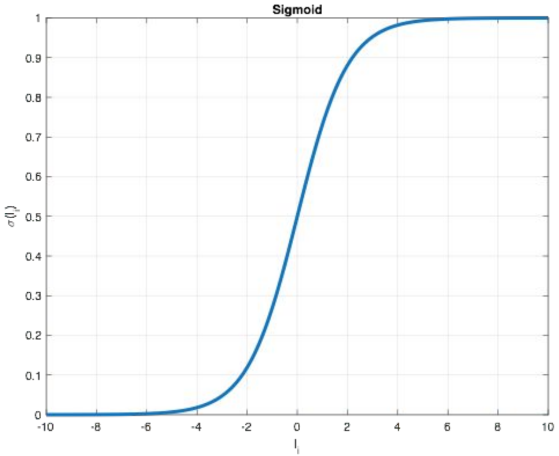
Cross-entropy loss (4)

- In general, cross-entropy loss works better than square error loss:
  - Square error loss usually gives too much emphasis to incorrect outputs.
  - In square error loss, as the output gets closer to either 0.0 or 1.0 the gradients get smaller, the change in weights gets smaller and training is slower.

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.
- Use sigmoid at each output independently instead of softmax

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



Multi-label classification (2)

- Cross-entropy loss for multi-label classification:

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

Outline

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

Regularization

- Control the capacity of the network to **prevent overfitting**
  - L2-regularization (weight decay):

regularization parameter

$$\mathcal{L}_{new} = \mathcal{L} + \frac{\lambda}{2} W^2$$

  - L1-regularization:

$$\mathcal{L}_{new} = \mathcal{L} + \frac{\lambda}{2} |W|$$

Outline

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

Example

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

			"1"	"2"	"3"
0.1	0.3	0.3	1	0	0
0.2	0.4	0.3	0	1	0
0.7	0.3	0.4	0	0	1

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

			"1"	"2"	"3"
0.3	0.1	0.1	1	0	0
0.4	0.7	0.2	0	1	0
0.3	0.2	0.7	0	0	1

Classification accuracy = 2/3
cross-entropy loss = 1.92

[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