Funzioni di loss

Funzioni di loss

Definizione

Si definisce funzione di costo o *loss* una funzione matematica che si prefigge l'obbiettivo di:

*Valutare la bontà di un modello nel predire output **uguali** o **vicini** a quelli attesi.*

In generale:

- Assegna un punteggio basandosi sul confronto fra predizioni e attese.
- È una funzione della quale si ricerca un massimo o un minimo.
- È correlata al problema che si affronta e si vuole risolvere.

Funzioni di loss

Le loss più comuni

La *loss* è correlata al problema che si tenta di risolvere. Per questo motivo è possibile distinguere tipologie ‘comuni’ di loss per i principali problemi:

Regressione

- *Mean squared error loss*
- *Mean absolute error loss*
- *Root mean squared error loss*
- *Binary Cross-Entropy loss*
- *Categorical Cross-Entropy loss*
- …

Classificazione binaria e multi-classe

Funzioni di loss

Loss di classificazione

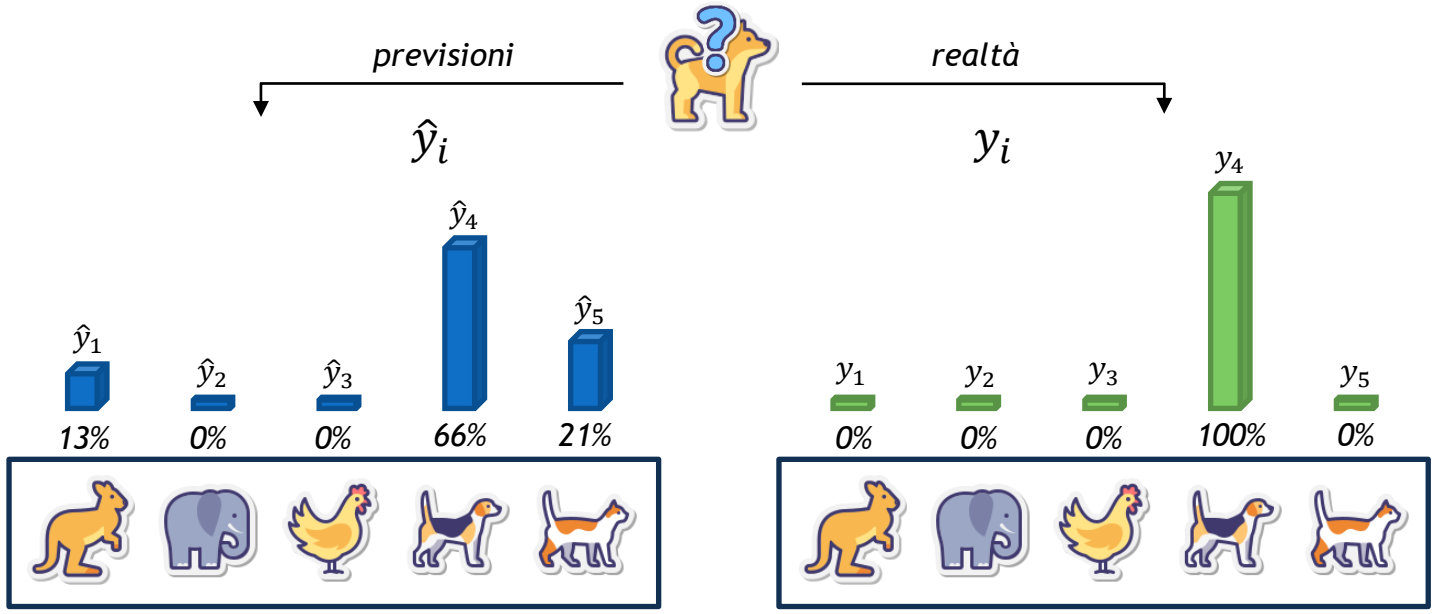
Principalmente due: *Cross-Entropy Loss* - *Binary Cross-Entropy Loss*.

- In termini generali: *misurano entrambe la differenza fra due distribuzioni di probabilità.*
- Nell'ambito del machine learning: *misurano la distribuzione di probabilità dell'accadere di molteplici eventi e la confrontano con la reale distribuzione di probabilità.*
- *Binary è la versione specializzata della Cross-Entropy Loss per problemi di classificazione binaria, o, più in generale, per valutare l'accadere di due eventi specifici.*

Funzioni di loss

Loss di classificazione

Principalmente due: *Cross-Entropy Loss* - *Binary Cross-Entropy Loss*.

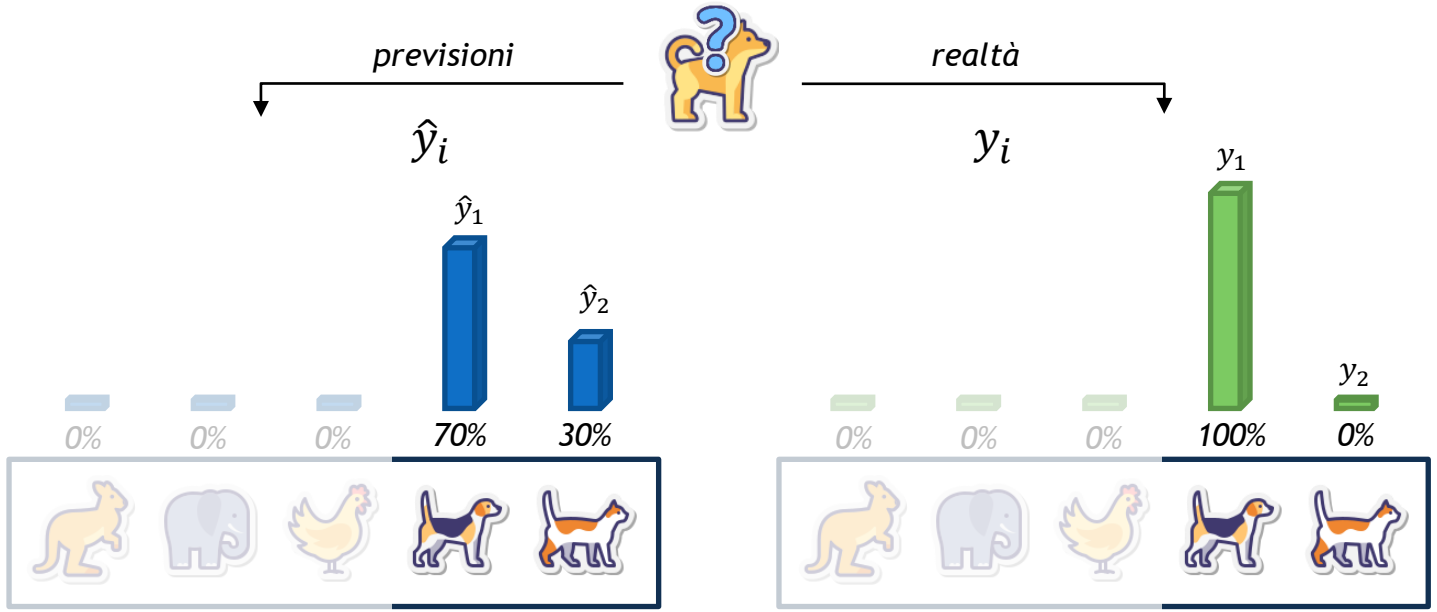


$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

Funzioni di loss

Loss di classificazione

Principalmente due: *Cross-Entropy Loss* - ***Binary Cross-Entropy Loss***.



$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

Funzioni di loss

Loss di regressione

Principalmente tre: ***MSE/RMSE*** - *MAE*.

MSE/RMSE:

- Misura quanto la predizione di un modello combacia con il valore atteso.
- Considera per ogni coppia *(reale, predetto)* la differenza al quadrato.
- I risultati, sempre positivi, saranno sommati e mediati.
- P*er ottenere una loss che mantenga l'unità di misura dei dati si può eventualmente considerarne la radice quadrata, RMSE.*

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

Funzioni di loss

Loss di regressione

Principalmente tre: *MSE/RMSE* - ***MAE***.

MAE:

- Misura quanto la predizione di un modello combacia con il valore atteso.
- Considera per ogni coppia *(reale, predetto)* il valore assoluto della differenza.
- I risultati, sempre positivi, saranno sommati e mediati.
- *Dà un'idea diretta di quanto i valori predetti discostano dal valore atteso enfatizzando meno le differenze di outliers.*

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Funzioni di loss

Loss in PyTorch

In PyTorch, il modulo *nn* mette a disposizione una lista di loss comuni, standard e direttamente utilizzabili nelle proprie architetture di rete:

torch.nn - Loss Functions

Da questa lista è possibile individuare, ad esempio, le loss precedentemente descritte:

- Mean Absolute Error Loss : *L1Loss*
- Mean Squared Error Loss : *MSELoss*
- Cross-Entropy Loss : *CrossEntropyLoss*
- Binary Cross-Entropy Loss : *BCELoss*

Funzioni di loss

Loss in PyTorch: esempio

Di seguito un esempio di utilizzo delle loss tramite *nn* e senza:

```python
import torch
import torch.nn as nn

predetti    = torch.tensor([2.5, 4.8, 3.2, 5.1])
attuali     = torch.tensor([2.0, 5.0, 3.5, 4.8])

# Applicazione della Mean Squared Error
mse_loss = nn.MSELoss()
mse = mse_loss(predetti, attuali)
print(f"MSE Loss: {mse.item():.4f}")

# Applicazione della Mean Absolute Error
mae_loss = nn.L1Loss()
mae = mae_loss(predetti, attuali)
print(f"MAE Loss: {mae.item():.4f}")
```

0.1175

0. 3250

```python
import torch

predetti    = torch.tensor([2.5, 4.8, 3.2, 5.1])
attuali     = torch.tensor([2.0, 5.0, 3.5, 4.8])

# Applicazione della Mean Squared Error
mse = torch.sum(torch.pow(predetti-attuali, 2)) / 4
print(f"MSE Loss: {mse.item():.4f}")

# Applicazione della Mean Absolute Error
mae = torch.sum(torch.abs(predetti-attuali)) / 4
print(f"MAE Loss: {mae.item():.4f}")
```

0.1175

0. 3250

Proviamo?