Addestramento

Addestramento

Premessa

Procediamo all'addestramento della rete con i seguenti assunti:

1. Conosciamo l'obbiettivo da raggiungere.
2. Possediamo i dati: sintetici o meno.
3. I dati sono stati convertiti in tensori.
4. I dati sono stati opportunamente pre-processati e normalizzati.
5. È stata definita la rete, i layer che la costituiscono e come eseguirli.

Il prossimo passo è la creazione di ciò che in gergo è chiamato *loop di addestramento*.

Addestramento

Premessa

La rete creata possiede uno o più layer, ognuno con i propri compiti, e li combina al fine di ottenere uno o più output. Oltre questo vi sono poi i molteplici parametri addestrabili che definiscono, nell'insieme, la conoscenza appresa dalla rete.

Una rete così definita, con dei parametri *inizializzati* ma non addestrati **è una rete funzionante**.

Come funziona?

In generale, al pari del lancio di una moneta. Può andare bene…può andare male…Non è di certo affidabile, solida e ripetibile.

Addestramento

Premessa

Per questo esiste la fase di addestramento: una serie di ripetizioni di analisi dei dati in ingresso per giungere ad un solo obbiettivo.

Modificare, uno per uno, **i parametri addestrabili** della rete al fine di ottenere l'output migliore per ognuno degli input forniti.

Da notare **due** particolari:

1. Nulla impedisce di modificare manualmente ogni parametro a mano. Questo, a patto che si sappia come modificarli e lo si faccia, in alcuni casi, per *milioni* o addirittura *miliardi* di parametri.
2. Fine ultimo è avere per ogni input l'output migliore. ***Migliore** non significa **esatto***. Ci saranno casi in cui si ottiene un output *esatto*, altri in cui il risultato è *quasi esatto* ed altri ancora in cui ci saranno *errori*…

Addestramento

Loss e optimizer

Durante il loop di addestramento entrano in gioco due entità:

1. la funzione di costo o *loss*.
2. Il metodo/sistema di ottimizzazione, l'*optimizer*.

In termini generali:

- La ***loss*** ha la funzione di indicare quanto errato è l'apprendimento del modello ossia quanto la conoscenza attuale lo porta a sbagliare le previsioni fatte sugli input rispetto gli output attesi.
- L'***optimizer*** sfrutta gli errori fatti, la quantità e la loro gravità e decide come andare a modificare ogni singolo parametro di addestramento per far sì che al prossimo tentativo gli esiti siano **migliori**.

Addestramento

Funzionamento generale

L'addestramento, in passi, può essere descritto così:

- Si considera un ***batch*** di elementi di addestramento: *può essere un solo elemento, più elementi, l'intero dataset…*
- I dati ed eventualmente le etichette sono dati in pasto alla rete e, da quest'ultima, si estraggono gli output…le previsioni.
- La funzione di costo identifica gli errori confrontando, eventualmente, previsioni e valori attesi.
- Gli errori commessi sono valutati al fine di identificarne la gravità, la provenienza e il peso che hanno avuto.
- L'ottimizzatore sfrutta queste informazioni per modificare i parametri della rete per il giro successivo.
- L'operazione procede fino a che tutti i campioni di addestramento non sono stati utilizzati, ossia è stata completata un'***epoca***…una o più volte.

Addestramento

Iperparametri

Gli **iperparametri** sono parametri decisi prima di avviare il loop di addestramento e che andranno a definire come questo avverrà. Sono anch'essi modificabili, non influenzeranno la conoscenza della rete ma il modo e il tempo in cui apprenderà.

Di iperparametri ve ne sono molteplici; i più comuni:

- *Numero di epoche*.
- *Batch size*.
- *Learning rate*.

Addestramento

Numero di epoche

Premesso che addestrare il modello significa fare in modo che la rete impari la correlazione che esiste tra i dati di input e l'output:

- L'addestramento è un processo iterativo che viene eseguito per un determinato numero di volte.

*Ogni iterazione viene detta **epoca**.*

- Ad ogni epoca l'intero dataset viene dato in pasto al modello, eventualmente a piccoli passi.

*Ogni passo che porterà al compimento di un epoca è chiamato **step**.*

Addestramento

Batch size

Ad ogni epoca, i dati di input vengono passati alla rete a gruppi. Il termine comunemente utilizzato per indicare questo gruppo di campioni è ***batch***. Il batch può contenere un numero variabile di elementi:

- Batch di un solo elemento.
- Batch di *n* elementi, con *n* minore della totalità del dataset.
- Batch contenente l'intero dataset.

Generalmente, il batch deve contenere una raccolta varia di campioni in modo tale da ottenere **variabilità** in fase di apprendimento:

- Al calare dei campioni si riduce, probabilmente, la variabilità dei dati. Le sfumature di realtà, le differenze e le affinità non si notano.
- Al crescere dei campioni cresce la difficoltà di cogliere affinità e differenze in quanto i dati possono essere troppi da «digerire».

Addestramento

Learning rate

L’aggiornamento dei parametri della rete, quindi l’addestramento, dipende in particolar modo dal learning rate o tasso di apprendimento.

Questo iperparametro, insieme ad altri:

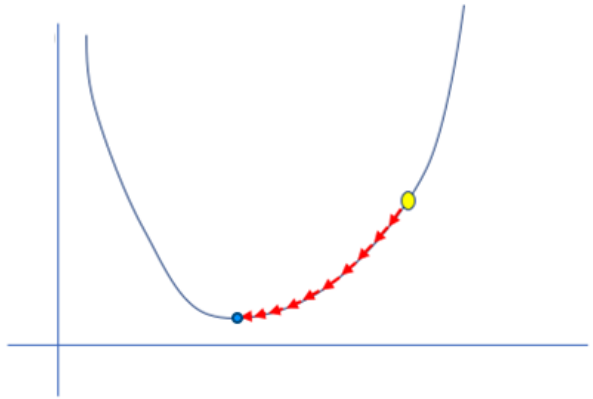
- Viene usato dell’ottimizzatore.
- Funge da regolarizzatore per decide di quanto i parametri verranno modificati al passo successivo.

È un valore decimale che va definito all’inizio dell’addestramento e, la scelta corretta di questo valore può determinare la riuscita o la non riuscita della fasi di apprendimento dell’algoritmo di machine learning.

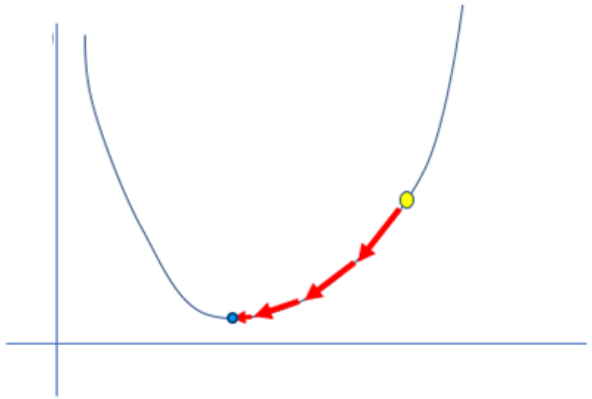
Addestramento

Learning rate

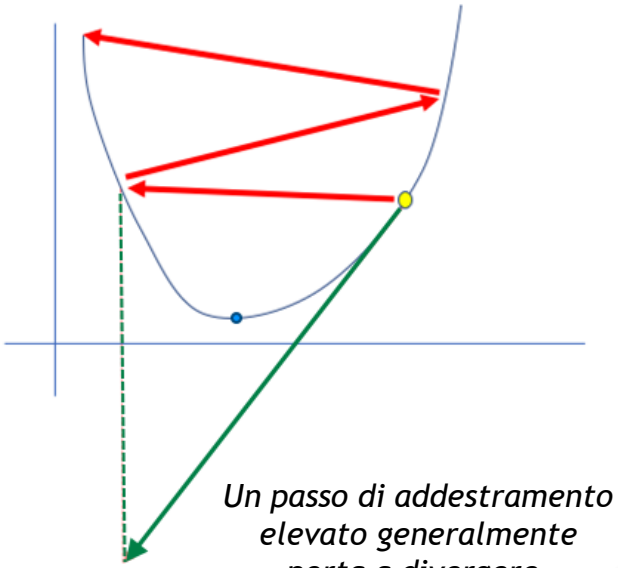
Un valore troppo piccolo, troppo grande…può rallentare l'apprendimento o impedirne addirittura la convergenza.



Procedere a piccoli passi dilata il tempo di convergenza e può far raggiungere dei minimi locali diversi dall'ottimo.



Procedere con un passo «adeguato» è l'obbiettivo da perseguire per ottenere un addestramento rapido ed efficiente.



Un passo di addestramento elevato generalmente porta a divergere.

Addestramento

Forward e Back Propagation

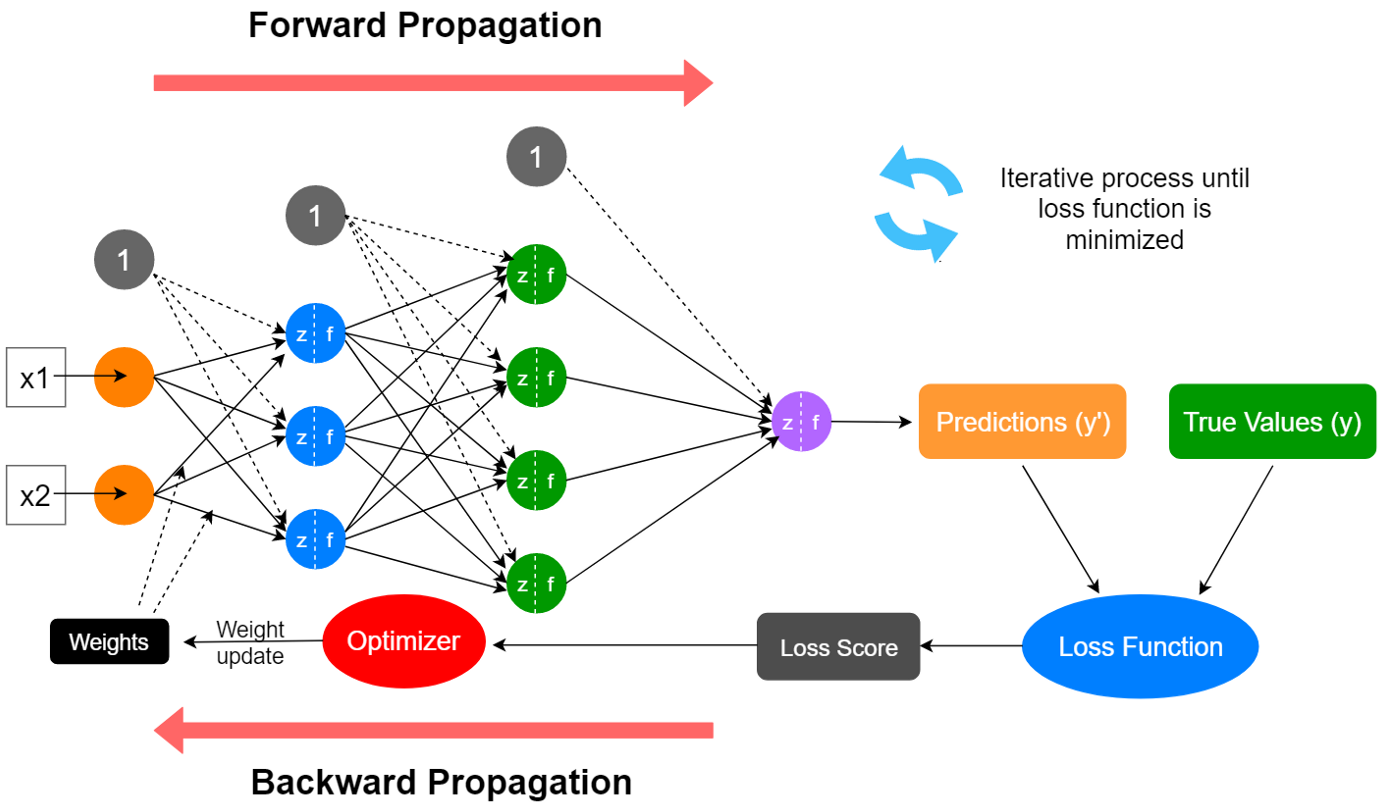
L'addestramento di una rete, nella sua accezione più semplice, può essere descritto in due fasi:

1. ***Forward propagation***: la rete riceve gli input, ogni neurone, parametro, layer, calcola il suo output ed infine si giunge alle predizioni finali. Nel fare questo si commetteranno una serie di errori dei quali la funzione di loss terrà traccia, compresa la loro «gravità».

2. ***Back propagation***: la rete viene ripercorsa in maniera inversa, si prendono in esame le predizioni e i valori attesi per ognuna di queste, si tiene traccia dell'errore commesso, del peso che ogni parametro ha avuto nel concorrere all'errore e come, quest'ultimo, andrebbe modificato. Si effettua l'aggiornamento dei parametri e ci si prepara al prossimo ciclo di addestramento.

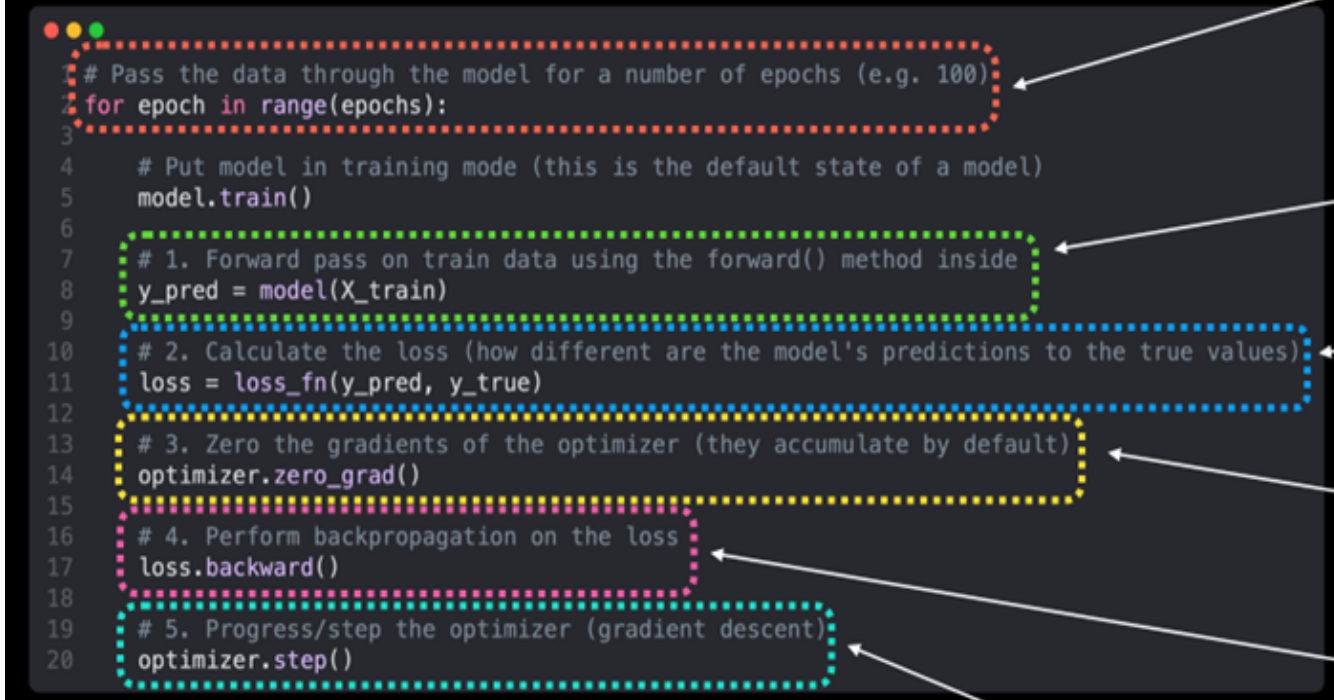
Addestramento

Forward e Back Propagation



Rif: (Medium)

PyTorch training loop



```python
# Pass the data through the model for a number of epochs (e.g. 100)
for epoch in range(epochs):

    # Put model in training mode (this is the default state of a model)
    model.train()

    # 1. Forward pass on train data using the forward() method inside
    y_pred = model(X_train)

    # 2. Calculate the loss (how different are the model's predictions to the true values)
    loss = loss_fn(y_pred, y_true)

    # 3. Zero the gradients of the optimizer (they accumulate by default)
    optimizer.zero_grad()

    # 4. Perform backpropagation on the loss
    loss.backward()

    # 5. Progress/step the optimizer (gradient descent)
    optimizer.step()
```

Note: all of this can be turned into a function

Pass the data through the model for a number of epochs (e.g. 100 for 100 passes of the data)

Pass the data through the model, this will perform the `forward()` method located within the model object

Calculate the loss value (how wrong the model's predictions are)

Zero the optimizer gradients (they accumulate every epoch, zero them to start fresh each forward pass)

Perform backpropagation on the loss function (compute the gradient of every parameter with `requires_grad=True`)

Step the optimizer to update the model's parameters with respect to the gradients calculated by `loss.backward()`

Proviamo?