Federated Learning tramite HTTP: esempio di un caso medico

Arlind Pecmarkaj[1] and Leonardo Bigelli[2]

[1]a.pecmarkaj@campus.uniurb.it
[2]l.bigelli2@campus.uniurb.it

14 giugno 2024

Riassunto

Questo progetto descrive l'implementazione di un approccio di Federated Learning basato sul protocollo HTTP per l'addestramento e la creazione di un modello per la classificazione di pazienti diabetici, pre-diabetici o sani.

1 Introduzione e obiettivi

Il Machine Learning è una tecnologia che è sempre più usata nel campo medico. Ciò è dovuto ai potenziali miglioramenti che può offrire. Basti pensare a diagnosi che vengono migliorate grazie all'analisi di grandi dati sanitari e di molteplici marker da cui si possono trarre pattern non visibili da un umano. Il problema principale che si riscontra in questo campo con il Machine Learning tradizionale sono i dati stessi: lavorando con dati sensibili, lo spostamento di essi verso server esterni per la creazione di modelli di ML non è facile per via delle limitazioni legali e i problemi che riguardano la privacy. Un approccio con l'uso di Federated Learning permette di mantenere i dati in locale nelle strutture e comunque creare un modello che fa uso di tutti i dati possibili. In questo progetto si è cercato di creare un modello di Federated Learning per la diagnosi del diabete usando come protocollo di comunicazione HTTP.

2 Federated Learning e l'uso di HTTP

Il Federated Learning è una branca del Machine Learning in cui entità di calcolo multiple addestrano un modello in collaborazione. La particolarità è che i dati di addestramento rimangono decentralizzati. I nodi interessati creano una federazione, da qui il nome della tecnologia, in cui vengono scambiati soltanto i parametri appresi dai modelli e non i dati usati per l'addestramento. Nel nostro caso abbiamo deciso di simulare 4 nodi (simulati ognuno con un notebook Colab) che vengono coordinati da un server. Il processo di apprendimento è diviso in round: all'inizio di ogni round i client ricevono il modello globale dal server su cui faranno l'addestramento con i propri dati locali per poi restituirlo indietro. Alla fine del round i modelli vengono aggregati tramite un algoritmo. Poiché i nodi contribuiscono equamente al modello e lavorano sulla stessa quantità di dati si è scelto di usare FedAvg, la media dei parametri addestrati. La comunicazione dei nodi verso il server avviene tramite protocollo HTTP. In particolare

- I nodi addestrano un modello di rete neurale tramite Python e fanno una chiamata HTTP verso il server con i parametri addestrati. Per far ciò il server mette a disposizione delle API per trasferire i dati. I nodi rimangono in attesa fino alla risposta.
- Il server rimane in attesa fino a che non riceve i dati da tutti i nodi. A quel punto fa la media dei parametri ottenuti e li restituisce indietro.
- I nodi appena ricevono i parametri aggiornati, li usano per addestrare nuovi modelli e il ciclo si ripete.

Abbiamo deciso di simulare 4 nodi, denominati a, b, c e d. Nello specifico le API messe a disposizione dal server sono raggiungibili alla porta 3001, nel nostro caso contattando l'indirizzo IP 93.65.162.11 e sono

- Gli endpoint *nodo_a*, *nodo_b*, *nodo_c*, *nodo_d* sono usati dai rispettivi nodi per trasferire i parametri addestrati al server.

- L' endpoint *pesi* è usato per ricevere la media dei parametri dal server. É chiamabile solo quando tutti i nodi hanno inviato i propri parametri.

3 Addestramento

3.1 Dataset

Il dataset utilizzato per questo esperimento è "CDC Diabetes Health Indicators", reperibile dalla piattaforma dell'UCI, Al suo interno sono presenti dei dati sanitari di alcuni utenti, per prevedere se un soggetto in particolare sia soggetto o meno al problematica del diabete. Il dataset è composto da più di 25 mila instance, il che lo rende l'ideale per essere suddiviso in parti uguali.

3.2 Rete Neurale

Come modello utilizzato per il progetto è stato scelto di usare una rete neurale molto semplice. Questo è dovuto al fatto che lo scopo è la simulazione di un sistema federato e non l'implementazione di un modello di Deep Learning. Più precisamente, si è utilizzata una rete multi-layer perceptron, con solamente due layers densi, per simulare l'intera rete. Segue una descrizione della rete impiegata su ogni nodo:

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 21, 3)	6
dropout (Dropout)	(None, 21, 3)	0
flatten (Flatten)	(None, 63)	0
dense_1 (Dense)	(None, 1)	64

Total params: 70 (280.00 Byte)
Trainable params: 70 (280.00 Byte)
Non-trainable params: 0 (0.00 Byte)

Figura 1: Modello della rete neurale.

4 Funzionamento

4.1 Separazione del dataset

Il dataset è stato suddiviso in quattro parti, una per ciascun nodo del sistema federato. Ciascun client ha a sua disposizione più di 6000 campioni su cui andare ad effettuare l'addestramento della rete. In un sistema reale il numero di record per ogni client non sarebbe sufficiente per ottenere un'affidabilità del sistema finale.

4.2 Funzionamento in locale

Ogni client addestra il proprio modello di rete neurale in locale con i propri dati che ha a sua disposizione. I dati di ciascun client sono diversi tra loro. Una volta finito il processo di addestramento, i parametri del modello vengono inviati, tramite un API, al server. Il client attende il risultato dell'elaborazione del server e, una volta ricevuto, aggiorna i parametri del suo modello e addestra nuovamente la rete. Il numero di iterazioni è arbitrario, buona norma sarebbe iterare fino al raggiungimento di una convergenza tra tutti i nodi della rete.

4.3 Funzionamento lato server

Il server ha il compito di simulare il nodo centrale del sistema di Federated Learning. Esso riceve i parametri dai vari client, per poi andare ad eseguire un algoritmo di aggregazione dei parametri. Quest'ultimo è stato implementato nuovamente da noi, in quanto non esistente per la nostra necessità. L'algoritmo sfruttato è il FedAvg, ovvero il calcolo della media dei vari nodi ricevuti da tutti e quattro i client. Nella realtà, il server dovrà utilizzare i parametri da lui prodotti per addestrare il suo modello globale. Ciò non è stato possibile per via della poca potenza computazionale della macchina su cui era avviato il server a nostra disposizione.

5 Risultati sperimentali

5.1 Esperimento di base

L'esperimento è stato condotto iterando il procedimento per quattro volte. La differenza tra l'accuratezza dei modelli in locale alla prima iterazione, con quella all'ultima non è molto marcata. Questo fatto potrebbe essere dovuto ai pochi dati iniziali che ciascun nodo ha a proprio disposizione. Seguono i risultati inerenti alla lost e all'accuratezza, ottenuti durante la prima iterazione:

```
[0.32994478940963745, 0.8643172383308411]
[0.3457787334918976, 0.8570640087127686]
[0.32686594128608704, 0.8657363653182983]
[0.34031620621681213, 0.859113872051239]
```

Figura 2: Risultati alla prima iterazione.

Seguono ora i dati inerenti all'ultima iterazione, ovvero a convergenza raggiunta:

Abbiamo notato che la convergenza era stata raggiunta subito alla terza iterazione, in quanto i risultati ottenuti sia alla terza sia alla quarta iterazione erano equivalenti.

```
[0.33523356914520264, 0.8616366982460022]
[0.34145206212997437, 0.8586407899856567]
[0.3251466155052185, 0.8676285147666931]
[0.33531248569488525, 0.8607694506645203]
```

Figura 3: Risultati alla prima iterazione.

5.2 Esperimento con client malevolo

A scopo dimostrativo, abbiamo forzato uno dei quattro nodi a inviare al server dei pesi molto elevati rispetti a tutti e tre gli altri nodi. In questo caso si è notato che, impiegando sempre un algoritmo di aggregazione che andasse ad effettuare la media, la convergenza veniva raggiunta praticamente subito, ma l'accuratezza dell'intero sistema era inferiore all'esperimento originale. Questo dimostra che l'approccio FedAvg non è abbastanza robusto per resistere ad attacchi di tipo *poisoning*, dove si mina il modello di uno dei nodi della rete.

6 Conclusioni

Lo scopo del progetto era simulare un sistema Federated Learning, utilizzando l'algoritmo di aggregazione FedAvg. Il sistema è stato implementato correttamente e si è notato subito che se il numero dei dati a disposizione dei client non è sufficientemente elevato, la convergenza dei modelli in locale avviene fin troppo rapidamente.
Abbiamo notato che, rendendo un nodo malevolo, l'algoritmo FedAvg non riesce a gestire ciò. In futuro si potrebbe provare ad utilizzare altri algoritmi di aggregazione per vedere quali tra un gruppo più o meno ristretto, posso gestire bene questa casistica.
Un altro possibile sviluppo futuro portrebbe essere l'implementazione dello stesso sistema ma su un dataset di dimensioni più onerose, per notare la differenza in termini di velocità di convergenza.