Progetti Machine Learning A.A. 2023/24

Version 1.0

1 Tipologie di Progetto

Sono disponibili due tipologie di progetto (**A** e **B**), con modalità di svolgimento e valutazione leggermente differenti.

La scelta tra le due tipologie di progetto deve essere comunicata da ogni studente prima di sostenere la prova orale.

1.1 Progetto A

- Svolto individualmente
- Contributo al voto finale: 20%
- Consegna: notebook Jupyter opportunamente commentato, non oltre il giorno della prova orale
- Da discutere durante la prova orale

1.2 Progetto B

- Svolto individualmente o in gruppi di 2-3 studenti
- Contributo al voto finale: 40%
- Consegna: codice sorgente e breve relazione, entro il **15/09/2024**
- Da discutere non oltre la sessione di esami di settembre 2024, in data da concordare con i docenti al momento della consegna

Cosa deve contenere la relazione? Si richiede una descrizione **sintetica** di: metodologia adottata e scelte progettuali affrontate; implementazione della soluzione (senza riportare il codice); valutazione delle prestazioni della soluzione.

Si suggerisce di strutturare la relazione con lo stile di un articolo scientifico, eventualmente utilizzando i template LaTeX/Word foniti da ACM o IEEE:

- `https://www.acm.org/publications/proceedings-template`
- `https://www.ieee.org/conferences/publishing/templates.html`

1.3 Modalità di Consegna

Per entrambe le tipologie di progetto, la consegna avviene tramite l'invio di una e-mail indirizzata ai docenti. Per evitare problemi con i filtri antivirus dei server di posta, anziché allegare il codice alla mail, si raccomanda di caricare il codice su uno spazio di storage (es., Google Drive, Dropbox) oppure in un repository pubblico (es. GitHub) e inserire nella mail solo un link al codice.

2 Tracce – Progetto A

Si noti che le tracce non contengono (volutamente) dettagli relativi alla metodologia di risoluzione richiesta. In assenza di specifiche puntuali, è lasciata allo studente la libertà scegliere la strategia di risoluzione da adottare.

2.1 Traccia A1 - Riconoscimento di immagini MNIST-like

L'obiettivo del progetto è addestrare e valutare uno o più modelli di classificazione. Lo studente può scegliere di utilizzare uno dei seguenti due data set, entrambi contenenti immagini nello stesso formato del noto data set MNIST (immagini in scala di grigio, 28x28):

- **Extended MNIST (EMNIST)**: versione estesa del dataset MNIST, contenente lettere alfabetiche invece delle cifre. Viene fornito un file CSV `emnist-letters.csv` con circa 80,000 immagini relative a lettere maiuscole o minuscole. Nella classificazione non occorre distinguere tra maiuscole e minuscole, quindi le classi da riconoscere sono 26. Ogni riga del file contiene l'etichetta della classe corrispondente all'immagine (un intero tra 1 e 26), seguita da 784 interi compresi tra 0 e 255 che codificano l'intensità di grigio di ogni pixel.

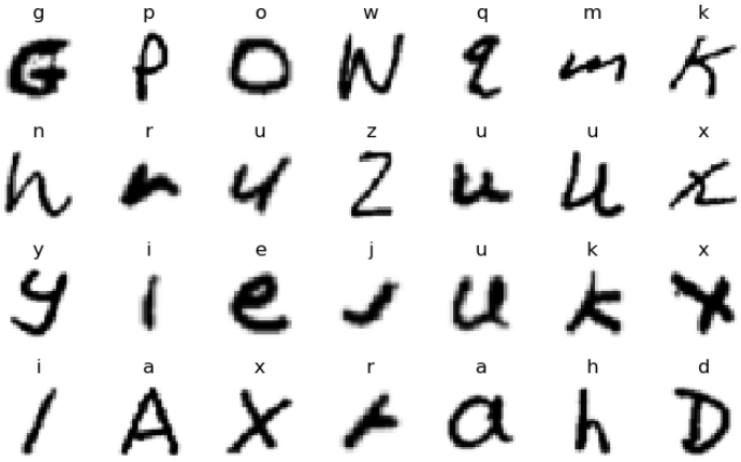

Figure 1: Alcuni esempi di immagini EMNIST.

Viene riportato di seguito un frammento di codice Python che può essere usato per caricare i dati, separare le immagini dalle etichette, ed effettuare il reshaping delle immagini da 784x1 a 28x28:

```
import pandas as pd
df = pd.read_csv("emnist-letters.csv")
X = df.iloc[:,1:].to_numpy().reshape(-1, 28, 28, order="F")
y = df.iloc[:,0].to_numpy()-1
```

- **MedMNIST-Pneumonia**: il dataset *MedMNIST* contiene una collezione di immagini di natura medica (e.g., ottenute tramite radiografie o TAC) riportate nel formato delle immagini MNIST. Per il progetto si considera uno specifico insieme di immagini, relative a radiografie del torace, classificate in base alla presenza (classe 1) o assenza (classe 0) di polmonite. Vengono forniti i due file `pneumonia_images.npy` e `pneumonia_labels.npy`, ottenuti esportando array di NumPy. Possono essere caricati in NumPy come segue:

```
import numpy as np
X = np.load("pneumonia_images.npy")
y = np.load("pneumonia_labels.npy")
```

Figure 2: Alcuni esempi di immagini MedMNIST.

2.2 Traccia A2 - Predizione del prezzo di automobili

L'obiettivo del progetto è addestrare e valutare uno o più modelli di regressione per predire il prezzo di vendita di automobili usate.

Il progetto si basa su un dataset contenente circa 40,000 esempi di automobili usate vendute nel Regno Unito, fornito tramite il file CSV `car_prices.csv`. Il dataset contiene le seguenti colonne:

Colonna	Descrizione
`manufacturer`	Casa produttrice
`model`	Modello
`year`	Anno di produzione
`transmission`	Cambio automatico/manuale
`mileage`	Miglia percorse
`fuelType`	Alimentazione (e.g., benzina, diesel)
`mpg`	Consumo stimato (miglia per gallone)
`engineSize`	Cilindrata
`price`	Prezzo di vendita (target della predizione)

2.3 Traccia A3 - Mountain Car

L'obiettivo del progetto è implementare e valutare una o più soluzioni basate su reinforcement learning per il problema *Mountain Car* della suite OpenAI Gymnasium: `https://gymnasium.farama.org/environments/classic_control/mountain_car/`

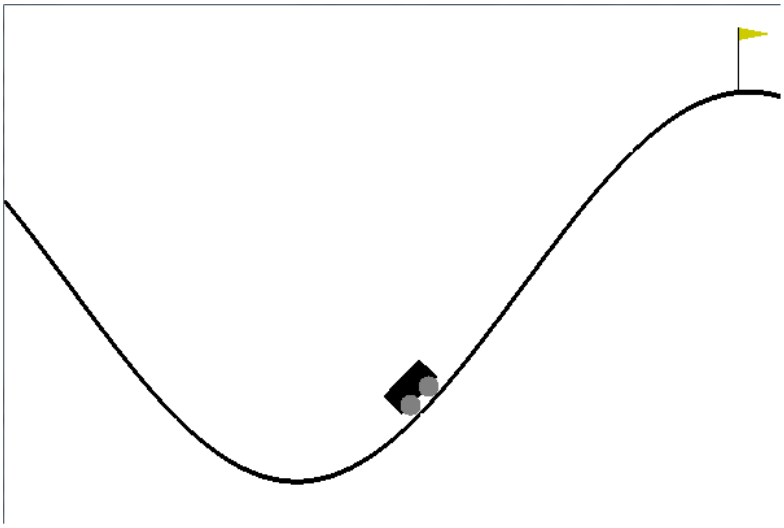

Figure 3: Illustrazione del problema Mountain Car.

3 Tracce – Progetto B

Si noti che le tracce non contengono (volutamente) molti dettagli relativi al problema da risolvere o alla metodologia di risoluzione richiesta. In assenza di specifiche puntuali, è lasciata allo studente la libertà di interpretare alcuni aspetti del problema e/o decidere la strategia di risoluzione da adottare.

Un numero limitato di gruppi può svolgere ciascuna traccia. **L'assegnazione della traccia deve quindi essere richiesta inviando una e-mail ai docenti.** Le richieste saranno accolte secondo una logica *first-come-first-served*.

3.1 Traccia B0

Gli studenti che volessero affrontare un particolare problema nell'ambito del progetto possono proporlo ai docenti. Per proporre un argomento, occorre inviare una e-mail ai docenti del corso, indicando (i) nome e matricola degli studenti coinvolti, (ii) una descrizione del progetto che si intende svolgere. I docenti valuteranno ed eventualmente accetteranno la proposta.

Se l'argomento proposto è correlato con esercizi/progetti svolti nell'ambito di altri corsi, tale circostanza deve essere espressamente segnalata nella proposta del progetto.

3.2 Traccia B1 - Federated Learning with Flower (max 3 gruppi)

Nella maggior parte dei casi, i dati su cui vengono addestrati i modelli di ML non sono direttamente generati/collezionati sulla macchina che esegue il processo di addestramento. Al contrario, è possibile che i dati provengano da una moltitudine di sorgenti e dispositivi differenti e distribuiti geograficamente (e.g., smartphone, sensori distribuiti in una città, PC portatili). Quello che accade tradizionalmente è che questi dati vengono raccolti e inviati ad un server (e.g., nel cloud), dove in seguito si procederà all'addestramento del modello.

Questo modello "classico" di ML non è ideale o attuabile in molti scenari, per diverse ragioni:

- Regolamenti sull'uso dei dati (e.g., GDPR in Europa), che potrebbero impedire di spostare dati sensibili in altre regioni geografiche;
- Preferenza degli utenti, che potrebbero non volere che i propri dati siano trasferiti altrove (e.g., informazioni sullo stato di salute);
- Volume eccessivo dei dati (e.g., sensori e video-camere in scenari Internet-of-Things potrebbero produrre una quantità di dati tale da non poter essere trasferita in maniera efficiente).

Il paradigma del *Federated Learning* (FL) fornisce una soluzione a queste problematiche, puntando a "spostare l'addestramento" anziché spostare i dati. Nell'ambito del FL molteplici *nodi* (e.g., organizzazioni diverse, dispositivi mobili diversi) partecipano all'addestramento di un modello, senza che nessuno dei nodi sia in possesso dell'intero data set. Nella configurazione di FL più semplice, tra i nodi esiste un *server* centrale e molteplici nodi *client*. Ciascun client è in possesso di una porzione dei dati da utilizzare. L'addestramento procede come segue:

1. Il server inizializza il modello da addestrare (e.g., architettura e pesi inizializzati di una rete neurale) e ne invia una copia a ciascun client.

2. Ciascun client, con i dati a sua disposizione, effettua alcune epoche di addestramento del proprio modello locale.

3. Ciascun client invia al server i parametri aggiornati del modello.

4. Il server aggrega le diverse versioni del modello prodotte dai client. Esistono diverse *strategie* per l'aggregazione dei modelli. La più semplice è *FedAvg*, in cui il server semplicemente aggiorna ciascun parametro come la media del valore calcolato da ciascun client (eventualmente pesata sul numero di esempi di training usati da ogni client).

5. Il server invia il modello aggiornato ai client. Si ripetono gli step sopra per un certo numero di *round*.

L'obiettivo del progetto è sviluppare degli script/notebook per studiare sperimentalmente il funzionamento del FL in una varietà di scenari. Si richiede di utilizzare il framework per FL *Flower* (paper[1]; sito web[2]), che si integra con le principali librerie per ML, tra cui TensorFlow. Tra le varie funzionalità, Flower consente di *simulare* scenari di FL, eseguendo un numero elevato di client sulla macchina locale.[3]

Nello specifico, si richiede di valutare le prestazioni di FL in termini di accuratezza (rispetto ad un modello addestrato in maniera tradizionale sull'intero data set) e tempo di convergenza (come numero totale di epoche e round), al variare delle seguenti configurazioni:

- task e data set di riferimento (almeno 3 a scelta; e.g., è possibile far riferimento ai data set proposti nei progetti A)
- numero di client
- strategia di aggregazione (usandone almeno 2 tra quelle disponibili in Flower)
- distribuzione dei dati tra i client (e.g., in un task di classificazione, ci si aspettano prestazioni diverse a seconda che i client osservino distribuzioni simili o differenti della classi nel proprio training set, fino al caso estremo in cui solo alcuni client osservano esempi di alcune classi)

Si richiede di produrre i risultati di tale valutazione sotto forma di tabelle e/o grafici. Inoltre, gli script sviluppati dovrebbero consentire di replicare gli esperimenti in maniera quanto più automatizzata possibile.

[1] `https://arxiv.org/pdf/2007.14390.pdf`
[2] `https://flower.dev/docs/framework/index.html`
[3] `https://flower.dev/docs/framework/how-to-run-simulations.html`

3.3 Traccia B2 - Age Detection (max 4 gruppi)

L'obiettivo del progetto è sviluppare un'applicazione in grado di stimare, in maniera quanto più accurata possibile, l'età di un soggetto ritratto in una fotografia. Il problema può essere logicamente scomposto nei seguenti sotto-problemi:

P0 data una fotografia di dimensione arbitraria che ritrae una persona (o, opzionalmente[4], più persone), produrre una nuova foto che contenga solo il volto, opportunamente centrato nell'immagine;

P1 data l'immagine di un volto, predire l'età. Ai fini del progetto, si lascia libertà di scelta sul tipo di output del modello:

- età del soggetto, come numero intero (e.g., 23);
- fascia d'età (e.g., 0-5, 5-10, 10-20, 20-30, ..., 90+)

Ai fini della vlutazione del progetto, si darà maggiore enfasi al problema **P1**. Si suggerisce quindi di risolvere prima tale problema.

Per risolvere **P0**, è consentito l'uso di modelli già addestrati e librerie esistenti (e.g., il modello MTCNN, di cui esistono varie implementazioni open-source, tra cui: `https://github.com/ipazc/mtcnn`).

Per risolvere **P1**, si richiede si sviluppare e addestrare un proprio modello (eventualmente sfruttando tecniche di transfer learning). Per l'addestramento, viene fornito il data set *UTKFace* (descritto qui[5]; disponibile per il download su Teams), contenente oltre 20,000 immagini di volti umani ritagliati, centrati ed etichettati con l'età anagrafica. L'etichetta è codificata nel nome di ciascun file, come numero intero seguito dal carattere '_'. Per estrarre tale informazione e costruire un DataSet di Tensorflow, si raccomanda di leggere la documentazione.[6]

Si richiede si sviluppare una interfaccia web o CLI per interagire con l'applicazione sviluppata. Per il problema P1, si richiede una opportuna valutazione delle prestazioni del modello, attraverso l'uso di un test set. Per il problema complessivo P0+P1, è sufficiente una dimostrazione su alcune immagini scelte dagli studenti.

Alcuni suggerimenti:

- Per svolgere questo progetto è consigliato l'uso di GPU per l'addestramento, eventualmente attraverso Google Colab.
- Considerato il numero non elevatissimo di esempi nel data set, valutare l'uso di tecniche di data augmentation.

[4] Non è obbligatorio valutare la soluzione in questo scenario.

[5] `https://susanqq.github.io/UTKFace/`

[6] `https://www.tensorflow.org/tutorials/load_data/images?hl=en`

3.4 Traccia B3 - AutoML (max 4 gruppi)

Automated ML (AutoML) indica un insieme di tecniche che hanno l'obiettivo di semplificare l'uso di ML e renderlo accessibile anche a persone con limitate competenze informatiche e limitate conoscenze dei modelli stessi. Per raggiungere tale obiettivo, AutoML cerca di semplificare la lunga e complessa sequenza di attività che caratterizzano solitamente un progetto di ML (data cleaning, feature engineering, model selection, hyperparameter tuning, evaluation), automatizzandole.

L'obiettivo di questo progetto è sviluppare un tool di AutoML che consenta, in maniera totalmente automatica, di risolvere problemi di regressione e classificazione su dati di tipo tabulare.

Il software sviluppato dovrebbe soddisfare i seguenti requisiti:

- Essere accessibile tramite interfaccia multi-piattaforma (web, desktop o CLI)
- Accettare in input data set in formato CSV (e, opzionalmente, altri); oltre a fornire i dati, l'utente deve indicare quale colonna del data set contiene le etichette (target della predizione) e selezionare il tipo di task da risolvere (i.e., classificazione/regressione)
- L'utente può selezionare diverse modalità di funzionamento:
  - *minAccuracy*: viene specificato un valore minimo di accuratezza (o valore massimo di MSE, per regressione) da raggiungere (su dati di validazione opportunamente estratti). Il software può terminare appena viene individuato un modello che soddisfa tale requisito.
  - *maxTime*: viene specificato un tempo massimo per la ricerca del modello migliore. Superato tale tempo, il software dovrebbe restituire il miglior modello individuato.
- L'utente, se desidera, può forzare la scelta di uno specifico modello (e.g., Random Forest, rete neurale) all'interno di una lista fornita dal software.
- Il software, quando termina l'esecuzione, riporta le metriche di prestazioni collezionate, una descrizione del modello scelto; inoltre, il modello viene esportato su file in un formato che consenta il successivo utilizzo all'interno di una libreria di ML (e.g., scikit-learn, Tensorflow).

Si richiede di valutare sperimentalmente l'efficacia della soluzione sviluppata, su almeno 3 diversi task (e.g., il task della traccia A2 e due task di classificazione a scelta).