Introduzione

Anche solamente dalla televisione o dai social media, probabilmente sapete già che il *machine learning*, l'apprendimento automatico, è diventata una delle tecnologie più interessanti del nostro tempo. Grandi aziende, come Google, Facebook, Apple, Amazon e IBM, hanno fortemente investito nella ricerca e nelle applicazioni del *machine learning*, e questo per un buon motivo. Anche se *machine learning* sembra ormai essere diventata un'espressione sulla bocca di tutti, di certo non ha ancora lo spazio che si merita. Questo interessante campo apre la strada a nuove possibilità ed è ormai diventato indispensabile nella nostra vita quotidiana. Pensate all'assistente vocale sullo smartphone o ai consigli di prodotti ai clienti o ai sistemi di prevenzione delle frodi con carta di credito o al filtraggio dello spam dalle caselle di posta elettronica o al rilevamento e diagnosi dei disturbi clinici; e l'elenco potrebbe continuare.

Primi passi con il machine learning

Se volete occuparvi di *machine learning* o migliorare le vostre capacità in termini di *problem solver* o magari addirittura considerare una carriera nel campo della ricerca sul *machine learning*, allora questo libro fa per voi! Per chi è davvero alle prime armi, i concetti teorici sui quali si basa il *machine learning* potrebbero essere piuttosto ostici, ma i molti libri pratici che sono stati pubblicati negli ultimi anni vi aiuteranno a familiarizzare con il *machine learning*, implementando potenti algoritmi di apprendimento.

Teoria e pratica

Osservare esempi pratici di codice ed elaborare le applicazioni d'esempio rappresentano un ottimo modo per approfondire questo campo. Inoltre, la presenza di esempi concreti aiuta a illustrare i concetti generali, mettendo direttamente in azione il materiale appreso. Tuttavia, ricordate sempre che da un grande potere derivano grandi responsabilità!
Oltre a offrire un'esperienza pratica nell'impiego del *machine learning* con il linguaggio di programmazione Python e le sue librerie, questo libro introduce i concetti matematici sui quali si basano gli algoritmi di *machine learning*, che sono essenziali per poterlo impiegare con successo. Pertanto, questo libro è differente da un testo puramente pratico: tratta i dettagli necessari relativamente ai concetti di *machine learning* e offre una descrizione

intuitiva, ma informativa, di come funzionano di algoritmi di *machine learning*, di come impiegarli e, soprattutto, di come evitare le trappole più comuni.

Perché Python?

Prima di approfondire il campo del *machine learning*, rispondiamo a questa domanda così fondamentale: "Perché Python?". La risposta è semplice: è potente ed è molto accessibile. Python è diventato il linguaggio di programmazione più popolare per la *data science* perché ci consente di mettere da parte gli aspetti noiosi della programmazione e ci offre un ambiente in cui possiamo delineare rapidamente nuove idee e mettere direttamente in pratica nuovi concetti.

Esplorare il campo del machine learning

Se provate a digitare "*machine learning*" come termine di ricerca in Google Scholar, otterrete un numero davvero imponente di pubblicazioni: 3.250.000. Naturalmente, non ci è possibile trattare tutti i minimi dettagli di tutti gli algoritmi e di tutte le applicazioni che sono emerse negli ultimi 60 anni. Tuttavia, in questo libro, cercheremo di imbarcarci in un viaggio di grande interesse, che tratta tutti gli argomenti e i concetti essenziali, per consentirvi di partire "con il piede giusto". Se trovate che la vostra sete di conoscenza non è ancora soddisfatta, potrete sfruttare tutte le risorse citate in questo libro per ampliare le vostre conoscenze in questo campo.
Noi, gli autori, ci sentiamo di dire che lo studio del *machine learning* ci ha migliorati come scienziati, pensatori e problem solver. Per noi, lo scopo del libro è quello di condividere questa conoscenza con voi. La conoscenza deriva dall'apprendimento, e per ottenerlo è fondamentale l'entusiasmo ma anche la padronanza delle competenze, che può essere raggiunta solo con la pratica.
Il percorso sarà a tratti difficoltoso e alcuni argomenti potranno risultare più ostici di altri, ma speriamo che raccoglierete questa opportunità e vi concentrerete sulla ricompensa. Ricordate che siamo sempre insieme in questo viaggio e, nel corso di questo libro, aggiungeremo sempre nuove e potenti tecniche al vostro arsenale, che vi aiuteranno a risolvere, guidati dai dati, anche i problemi più difficili.

A chi è rivolto questo libro

Se avete già studiato in dettaglio la teoria del *machine learning*, questo libro vi insegnerà a mettere in pratica le vostre conoscenze. Se invece già impiegate tecniche di *machine learning* e volete conoscerne meglio il funzionamento, questo libro è rivolto a voi.
Se poi, per voi, il campo del *machine learning* è un'assoluta novità, non preoccupatevi; troverete il tutto ancora più interessante! La promessa che vi facciamo è che il *machine learning* cambierà il vostro modo di pensare ai problemi che vi trovate a dover risolvere e vi mostrerà come affrontarli sbloccando la potenza già insita nei dati. Se volete imparare a usare Python per iniziare a rispondere alle domande critiche sui dati, questo è il libro giusto: che vogliate iniziare da zero o ampliare le vostre conoscenze nel campo della *data science*, scoprirete che questa è una risorsa essenziale e imperdibile.

Argomenti trattati

- Capitolo 1. Introduce le principali sotto-aree del *machine learning*, impiegate per affrontare le varie tipologie di problemi. Inoltre, tratta i passi essenziali per creare una tipica catena o *pipeline* per la costruzione di un modello di *machine learning*, che ci farà da guida i capitoli successivi.
- Capitolo 2. Torna alle origini del *machine learning* e introduce i classificatori binari perceptron e i neuroni lineari adattativi. Questo capitolo rappresenta un'introduzione "morbida" ai fondamentali della classificazione per pattern e si concentra sulle relazioni esistenti fra algoritmi di ottimizzazione e *machine learning*.
- Capitolo 3. Descrive gli algoritmi essenziali di *machine learning* dedicati alla classificazione e fornisce esempi pratici sull'uso di una delle più note e ricche librerie open source dedicate al *machine learning*: scikit-learn.
- Capitolo 4. Spiega come affrontare i problemi più comuni dei dataset grezzi, come i dati mancanti. Inoltre, tratta numerosi approcci utili per l'identificazione delle caratteristiche più informative nei dataset e la preparazione dei vari tipi di variabili in modo che divengano input corretti per gli algoritmi di *machine learning*.
- Capitolo 5. Descrive le tecniche essenziali per ridurre il numero delle caratteristiche di un dataset in set più piccoli, mantenendo nel contempo la maggior parte delle informazioni utili e discriminanti. Inoltre, tratta l'approccio standard alla riduzione della dimensionalità tramite l'analisi dei componenti principali e lo confronta con le tecniche di trasformazione con supervisione e non lineari.
- Capitolo 6. Tratta i pro e i contro nella stima delle prestazioni dei modelli predittivi. In più, tratta varie metriche per misurare le prestazioni dei nostri modelli e le tecniche di ottimizzazione degli algoritmi di *machine learning*.
- Capitolo 7. Introduce i vari concetti che consentono di combinare in modo efficace più algoritmi di apprendimento. Spiega come costruire *ensemble* di algoritmi con lo scopo di superare le debolezze dei singoli algoritmi di apprendimento, per ottenere predizioni più accurate e affidabili.
- Capitolo 8. Tratta i passi essenziali per trasformare dei dati testuali in rappresentazioni significative per gli algoritmi di *machine learning*, per consentire loro di predire le opinioni delle persone sulla base delle loro produzioni scritte.
- Capitolo 9. Prosegue lo sviluppo del modello predittivo del capitolo precedente ed esamina i passi essenziali per sviluppare applicazioni web che incorporano modelli di *machine learning*.
- Capitolo 10. Tratta le tecniche essenziali per la modellazione di relazioni lineari fra variabili target e di risposta, per eseguire predizioni su una scala continua. Dopo aver introdotto vari modelli lineari, affronta gli approcci a regressione polinomiale e ad albero.
- Capitolo 11. Passa a esaminare un'altra sotto-area del *machine learning*, l'apprendimento senza supervisione. Affronta algoritmi tratti da tre grandi famiglie di algoritmi a cluster, che vanno alla ricerca di gruppi di oggetti che condividono un certo livello di similarità.

- Capitolo 12. Estende il concetto dell'ottimizzazione a gradienti che abbiamo introdotto nel Capitolo 2. In questo capitolo, costruiremo potenti, reti neurali (NN – Reti neurali) multilivello basate sul noto algoritmo di retropropagazione di Python.
- Capitolo 13. Sviluppa le conoscenze tratte dal capitolo precedente per fornire una guida pratica per addestrare in modo più efficiente le reti neurali. Questo capitolo è incentrato su TensorFlow 2.0, una libreria open source Python che ci consente di utilizzare al meglio le moderne schede grafiche (GPU – *Graphic Processing Unit*) per costruire reti neurali profonde a partire da semplici elementi, tramite la comoda API Keras.
- Capitolo 14. Raccoglie il testimone del capitolo precedente e introduce le funzionalità e i concetti più avanzati di TensorFlow 2.0. TensorFlow è una libreria straordinariamente vasta e sofisticata e questo capitolo esamina concetti come la compilazione del codice in un grafo statico, per accelerarne l'esecuzione e per definire i parametri del modello addestrabile. Inoltre, questo capitolo fornisce ulteriori esempi di addestramento delle reti neurali profonde impiegando l'API di TensorFlow, Keras, e gli Estimator preimpostati di TensorFlow.
- Capitolo 15. Introduce le reti neurali convoluzionali (*CNN – Convolutional Neural Network*). Una rete neurale convoluzionale rappresenta un tipo particolare di architettura a rete neurale profonda particolarmente adatta ai dataset di immagini. Grazie alle loro prestazioni superiori rispetto agli approcci tradizionali, le reti neurali convoluzionali sono oggi ampiamente utilizzate nella visione computerizzata per ottenere risultati allo stato dell'arte in vari compiti di riconoscimento delle immagini. Nel corso di questo capitolo, scoprirete come i layer convolutivi possono fungere da potenti estrattori di caratteristiche per la classificazione delle immagini.
- Capitolo 16. Introduce un'altra nota architettura di rete neurale per il *deep learning*, particolarmente adatta a lavorare con il testo e altri tipi di dati sequenziali e dati di serie temporali. Come esercizio "di riscaldamento", questo capitolo introduce le reti neurali ricorrenti, per predire il *sentiment* delle recensioni di film. Poi, il capitolo parla dell'apprendimento da parte delle reti ricorrenti, per assimilare informazioni dai libri, così da generare un testo interamente nuovo.
- Capitolo 17. Introduce il noto schema "avversario" per le reti neurali, che può essere impiegato per generare nuove immagini, dall'aspetto realistico. Il capitolo parte con una breve introduzione agli *autoencoder*, un tipo particolare di architettura di rete neurale che può essere impiegato per la compressione dei dati. Il capitolo insegna poi a combinare la parte decoder di un autoencoder con una seconda rete neurale in grado di distinguere fra immagini reali e di sintesi. Consentendo alle due reti neurali di competere l'una con l'altra con un approccio all'addestramento di tipo avversario, implementerete una rete avversaria generativa che genera nuove cifre scritte a mano. Infine, dopo aver introdotto i concetti di base delle reti avversarie generative, il capitolo introduce dei miglioramenti in grado di stabilizzare l'addestramento avversario, come la metrica per la distanza Wasserstein.
- Capitolo 18. Tratta una sottocategoria del *machine learning* comunemente impiegata per l'addestramento di robot e altri sistemi autonomi. Questo capitolo parte introducendo le basi del *reinforcement learning* (*RL*) per farvi familiarizzare con le

interazioni agente/ambiente, il processo delle ricompense dei sistemi di *reinforcement learning* e il concetto di apprendimento dall'esperienza. Il capitolo si occupa delle due categorie principali del *reinforcement learning*: con e senza modello. Dopo aver affrontato i principali approcci algoritmici, come l'apprendimento Monte Carlo e a distanza temporale, implementerete e addestrerete un agente in grado di navigare in un ambiente costituito da un mondo a griglia impiegando l'algoritmo *Q-learning*. Per concludere, questo capitolo introduce l'algoritmo *deep Q-learning*, che è una variante del *Q-learning* basato però su reti neurali profonde.

Dotazione software necessaria

Per eseguire gli esempi di codice presentati in questo libro è necessaria un'installazione di Python 3.7.0 (o successivo) su MacOS, Linux o Microsoft Windows. Impiegheremo frequentemente le librerie di base di Python per il calcolo scientifico, fra cui SciPy, NumPy, scikit-learn, Matplotlib e pandas.
Nel primo capitolo troverete le istruzioni e alcuni utili suggerimenti per configurare l'ambiente Python e queste librerie di base. Aggiungeremo poi ulteriori librerie al nostro repertorio e troverete le istruzioni per la loro installazione nei rispettivi capitoli; per esempio, la libreria NLTK per l'elaborazione del linguaggio naturale nel Capitolo 8; il framework web Flask nel Capitolo 9; TensorFlow per l'addestramento efficiente delle reti neurali su GPU nei Capitoli da 13 a 18.

Download dei file di codice degli esempi

Potete scaricare i file di codice degli esempi da GitHub, all'indirizzo `https://github.com/rasbt/python-machine-learningbook-3rd-edition`. Una volta scaricato un file, espandete o estraete la cartella impiegando l'ultima versione di:

- WinRAR/7-Zip per Windows;
- Zipeg/iZip/UnRarX per Mac;
- 7-Zip/PeaZip per Linux.

Tutto il codice di questo libro è disponibile anche sotto forma di notebook Jupyter e trovate una breve introduzione nell'elenco del codice del *Capitolo 1*, in `https://github.com/rasbt/python-machinelearning-book-3rd-edition/tree/master/ch01#pythonjupyter-notebook`. Per altre informazioni generali sull'interfaccia grafica di Jupyter Notebook, potete consultare la documentazione ufficiale in `https://jupyter-notebook.readthedocs.io/en/stable`. Anche se vi consigliamo di impiegare Jupyter Notebook per l'esecuzione interattiva del codice, tutti gli esempi sono disponibili sotto forma di script Python (per esempio, `ch02/ch02.py`) e di notebook Jupyter (per esempio, `ch02/ch02.ipynb`). Inoltre, vi consigliamo di leggere il file `README.md` che accompagna ogni singolo capitolo, dove troverete ulteriori informazioni e aggiornamenti (per esempio, `https://github.com/rasbt/python-machine-learning-book-3rd-edition/blob/master/ch01/README.md`).

Convenzioni impiegate

In questo libro, troverete vari stili testuali che hanno lo scopo di distinguere i vari tipi di informazioni. Ecco alcuni esempi di questi stili e del loro significato.

- I piccoli frammenti di codice nel testo sono rappresentati nel seguente modo:

  "I pacchetti già installati possono essere aggiornati specificando il flag `--upgrade`".

- Un blocco di codice è formattato nel seguente modo:

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> y = df.iloc[0:100, 4].values
>>> y = np.where(y == 'Iris-setosa', -1, 1)
>>> X = df.iloc[0:100, [0, 2]].values
>>> plt.scatter(X[:50, 0], X[:50, 1],
...             color='red', marker='x', label='setosa')
>>> plt.scatter(X[50:100, 0], X[50:100, 1],
...             color='blue', marker='o', label='versicolor')
>>> plt.xlabel('sepal length')
>>> plt.ylabel('petal length')
>>> plt.legend(loc='upper left')
>>> plt.show()
```

- Ogni input una riga di comando o output è scritto nel seguente modo:

```
> dot -Tpng tree.dot -o tree.png
```

- I nuovi termini e le parole importanti o i termini in inglese sono scritti in corsivo.
- Le parole che compaiono sullo schermo, per esempio, nei menu o nelle finestre di dialogo, sono scritte in questo modo:

  "Fate clic sul pulsante *Next* per passare alla pagina successiva".

- Nelle formule, nei listati, identici agli originali disponibili al download, e nei frammenti di codice è stata naturalmente conservata la convenzione sintattica richiesta dal linguaggio quanto all'uso del punto decimale:

```
def__init__(self, env, discount_factor=0.95,
            epsilon_greedy=1.0, epsilon_min=0.01,
            epsilon_decay=0.995, learning_rate=1e-3,
            max_memory_size=2000):
```

  Nel testo, invece, per evitare ambiguità i numeri decimali e quelli superiori alle migliaia sono formattati in base alla consueta convenzione standard nazionale:

  "L'intero dataset contiene 50.000 esempi."

  e

  "Come potete vedere, il cambio di scala dei logit di un fattore $\alpha = 0{,}1$ genera probabilità pressoché uniformi [0,31, 0,31, 0,38]."