Prefazione

Gli algoritmi hanno sempre avuto un ruolo importante nella scienza e nella pratica dell'informatica. Questo libro si concentra sull'utilizzo degli algoritmi per risolvere i problemi tipici del mondo reale. Per trarre il massimo da questi algoritmi, è indispensabile comprendere più profondamente la loro logica e le loro basi matematiche. Inizieremo con un'introduzione agli algoritmi ed esploreremo varie tecniche di progettazione. Procedendo con la lettura, affronteremo la programmazione lineare, la valutazione (*ranking*) delle pagine e i grafi, e affronteremo gli algoritmi di machine learning, sempre sulla scorta delle loro basi matematiche e logiche. Questo libro contiene anche vari casi di studio per argomenti pratici come le previsioni del tempo, il clustering di tweet e i motori di raccomandazione per film, problemi mediante i quali impareremo ad applicare questi algoritmi in modo ottimale. Al termine di questo libro, avrete acquisito tutta la sicurezza necessaria per applicare gli algoritmi alla risoluzione di problemi computazionali tipici del mondo reale.

A chi è rivolto questo libro

Questo è un libro rivolto ai programmatori. Che voi siate programmatori esperti e desideriate acquisire una comprensione più profonda delle basi matematiche degli algoritmi o che abbiate solo una conoscenza limitata della programmazione o della *data science* e desideriate saperne di più su come sfruttare questi algoritmi, ben collaudati, per migliorare il modo in cui progettate e scrivete il codice, sicuramente lo studio di questo libro vi sarà di grande utilità pratica. L'esperienza di programmazione con Python è *essenziale*, mentre la conoscenza della *data science* è utile ma non indispensabile.

Argomenti trattati

Il Capitolo 1 riassume i fondamenti degli algoritmi. Si apre con una sezione sui concetti di base necessari per comprendere il funzionamento dei diversi algoritmi. Riepiloga come abbiamo iniziato a utilizzare algoritmi per formulare matematicamente determinate classi di problemi. Menziona anche i limiti dei diversi algoritmi. Quindi spiega i vari modi per specificare la logica di un algoritmo. Poiché per scrivere gli algoritmi in questo libro viene utilizzato Python, il capitolo spiega come impostare l'ambiente per svolgere gli

esempi. Quindi, discute i vari modi in cui le prestazioni di un algoritmo possono essere quantificate e confrontate con altri. Infine, il capitolo tratta i vari metodi di convalida di una determinata implementazione di un algoritmo.
Il Capitolo 2 si concentra sulle strutture di dati necessarie per contenere i dati temporanei di un algoritmo. Gli algoritmi possono essere a elevata intensità di dati, a elevata intensità di calcolo o entrambe le cose. Ma per tutti i diversi tipi, la scelta delle giuste strutture di dati è essenziale per la loro ottimale implementazione. Molti algoritmi hanno una logica ricorsiva o iterativa e richiedono strutture di dati specializzate di natura fondamentalmente iterativa. Poiché in questo libro utilizziamo il linguaggio Python, il capitolo si concentra sulle strutture di dati di Python.
Il Capitolo 3 presenta gli algoritmi di base impiegati per operazioni di ordinamento e ricerca. Questi algoritmi possono in seguito diventare la base per altri più complessi. Il capitolo inizia presentando diversi tipi di algoritmi di ordinamento, confrontando le prestazioni di vari approcci. Quindi, ne esamina vari per operazioni di ricerca e li mette a confronto, quantificandone le prestazioni e la complessità. Infine, il capitolo presenta le applicazioni pratiche di questi algoritmi.
Il Capitolo 4 presenta i concetti fondamentali di progettazione di vari algoritmi. Spiega anche diversi tipi di algoritmi e discute i loro punti di forza e punti deboli. Comprendere questi concetti è importante quando si tratta di progettare e ottimizzare algoritmi complessi. Il capitolo inizia discutendo diversi tipi di progetti algoritmici. Quindi, presenta la soluzione per il famoso problema del commesso viaggiatore. Poi introduce la programmazione lineare e i suoi limiti. Infine, presenta un esempio pratico che mostra come la programmazione lineare possa essere utilizzata per la pianificazione della capacità.
Il Capitolo 5 si concentra sugli algoritmi per problemi con i grafi. Sono molti i problemi computazionali che possono essere rappresentati al meglio in termini di grafi. Questo capitolo presenta i metodi per rappresentare un grafo e per eseguire ricerche su di esso. Eseguire ricerche in un grafo significa seguire sistematicamente gli archi del grafo in modo da visitarne i nodi. Un algoritmo di ricerca su grafi può scoprire molto sulla struttura di un grafo. Molti algoritmi iniziano cercando nel grafo fornito in input per ottenere alcune informazioni strutturali. Sono vari gli algoritmi che si occupano di ricerche sui grafi. Le tecniche di ricerca sono al centro del campo degli algoritmi dei grafi. Il capitolo tratta le due rappresentazioni computazionali più comuni dei grafi: come liste di adiacenza e come matrici di adiacenza. Successivamente, presenta un semplice algoritmo di ricerca in ampiezza (*breadth-first*). Poi presenta la ricerca in profondità (*depth-first*)e fornisce alcuni risultati standard sull'ordine in cui una ricerca in profondità visita i nodi.
Il Capitolo 6 introduce gli algoritmi di machine learning senza supervisione. Sono detti "senza supervisione" perché il modello o l'algoritmo cerca di apprendere dai dati le sue strutture, i suoi schemi e le sue relazioni intrinseche senza alcuna supervisione. In primo luogo, vengono discussi i metodi di clustering, metodi di machine learning che cercano di trovare modelli di somiglianza e relazioni fra i campioni di dati nel nostro dataset e quindi raggruppano questi campioni in gruppi, in modo tale che ogni gruppo o cluster di campioni di dati manifesti una qualche somiglianza, in base agli attributi o alle sue caratteristiche intrinseche. Poi il capitolo discute gli algoritmi di riduzione della dimensionalità, utilizzati quando si ha un numero eccessivo di feature. Successivamente, esamina alcuni algoritmi che si occupano del rilevamento delle anomalie. Infine, il capitolo presenta l'estrazione delle regole associative, un metodo di data mining utilizzato per esaminare e analizzare grandi dataset di transazioni per identificare modelli e regole

di interesse. Questi modelli rappresentano relazioni e associazioni interessanti fra i vari elementi nelle transazioni.
Il Capitolo 7 descrive i tradizionali algoritmi di machine learning con supervisione in relazione a un insieme di problemi di machine learning in cui è presente un dataset etichettato, ovvero dove vi sono attributi di input ed etichette o classi di output. Questi input e output vengono utilizzati per definire un sistema generalizzato, che può successivamente essere utilizzato per prevedere i risultati per punti dei dati non considerati in precedenza. Innanzitutto, viene introdotto il concetto di classificazione nel contesto del machine learning. Quindi, viene presentato il più semplice degli algoritmi di machine learning, la regressione lineare. Segue uno degli algoritmi più importanti, l'albero decisionale. Vengono discussi i limiti e i punti di forza degli algoritmi ad albero decisionale, seguiti da due importanti algoritmi, SVM e XGBoost.
Il Capitolo 8 introduce innanzitutto i concetti e i componenti principali di una tipica rete neurale, che sta diventando la tecnica più importante di machine learning. Quindi, presenta i vari tipi di reti neurali e spiega i tipi di funzioni di attivazione utilizzati per realizzarle. Poi tratta l'algoritmo di retropropagazione, l'algoritmo più utilizzato per far convergere il problema in una rete neurale. Successivamente, spiega la tecnica di trasferimento dell'apprendimento, che può essere utilizzata per semplificare notevolmente e automatizzare parzialmente l'addestramento dei modelli. Infine, presenta un esempio reale: come utilizzare il deep learning per rilevare oggetti nei dati multimediali.
Il Capitolo 9 presenta gli algoritmi per l'elaborazione del linguaggio naturale (NLP). Questo capitolo procede dagli aspetti teorici a quelli più pratici in modo progressivo. In primo luogo, presenta i fondamenti, seguiti dalle basi matematiche. Quindi, discute una delle reti neurali più utilizzate per progettare e implementare un paio di importanti casi d'uso per i dati testuali. Tratta anche i limiti dell'elaborazione del linguaggio naturale. Infine, presenta un caso di studio in cui viene addestrato un modello per rilevare l'autore di un articolo in base allo stile di scrittura.
Il Capitolo 10 si concentra sui motori di raccomandazione, che sono un modo per modellare le informazioni disponibili in relazione alle preferenze dell'utente e quindi utilizzarle per fornire raccomandazioni informate. La base del motore di raccomandazione è sempre costituita dalle interazioni fra utenti e prodotti. Questo capitolo inizia presentando l'idea alla base dei motori di raccomandazione, quindi, discute dei vari tipi di motori di raccomandazione e infine, illustra come vengono utilizzati per suggerire articoli e prodotti agli utenti.
Il Capitolo 11 si concentra sui problemi relativi agli algoritmi basati sui dati. Il capitolo inizia con una breve panoramica delle questioni relative ai dati e poi presenta i criteri per la loro classificazione. Successivamente, spiega come applicare determinati algoritmi alle applicazioni che operano su dati in streaming e quindi presenta l'argomento della crittografia. Infine, mostra un esempio pratico di estrazione di modelli dai dati di Twitter.
Il Capitolo 12 introduce gli algoritmi relativi alla crittografia. Il capitolo inizia presentando le basi storiche. Quindi, tratta gli algoritmi di crittografia simmetrica. Partiremo dagli algoritmi di hashing MD5 e SHA ed esamineremo i limiti e i punti deboli associati all'implementazione di algoritmi simmetrici. Successivamente, esamineremo gli algoritmi di crittografia asimmetrica e come vengono utilizzati per creare certificati digitali. Infine, vedremo un esempio pratico che riassume tutte queste tecniche.
Il Capitolo 13 spiega come gli algoritmi per dati su larga scala gestiscono i dati che non possono essere contenuti nella memoria di un singolo nodo e coinvolgono l'elabora-

zione da parte di più CPU. Questo capitolo inizia discutendo quali tipi di algoritmi sono più adatti per essere eseguiti in parallelo. Quindi, descrive le questioni relative alla parallelizzazione degli algoritmi. Presenta inoltre l'architettura CUDA e mostra come utilizzare una o più GPU per accelerare gli algoritmi e quali modifiche devono essere apportate all'algoritmo per utilizzare efficacemente la potenza della GPU. Infine, il capitolo introduce il cluster computing e spiega come Apache Spark è in grado di creare dataset resilienti distribuiti (RDD) per costruire un'implementazione parallela estremamente veloce degli algoritmi standard.
Il Capitolo 14 inizia con l'importante argomento della spiegabilità, che sta diventando sempre più importante a mano a mano che si va verso un'automatizzazione dei processi decisionali. Quindi, il capitolo presenta gli aspetti etici dell'utilizzo di un algoritmo e le possibilità di operare sulla base di pregiudizi, *bias*. Successivamente, tratta in dettaglio le tecniche per la gestione dei problemi NP-difficili. Infine, riassume i modi per implementare gli algoritmi e le sfide da affrontare.

Requisiti

Per sfruttare al meglio questo libro, impiegate Python versione 3.7.2 o successiva su un computer con almeno 4 GB di memoria RAM (consigliati 8 GB o più) e con sistema operativo Windows, Linux o Mac.
Anche se state utilizzando la versione digitale di questo libro, vi consigliamo di digitare il codice o di scaricarlo dal repository GitHub (URL disponibile nel prossimo paragrafo). In questo modo eviterete eventuali errori presenti nel codice o legati a operazioni di copia-incolla.

File degli esempi

Potete scaricare i file degli esempi su GitHub all'indirizzo `https://github.com/PacktPublishing/40-Algorithms-Every-Programmer-Should-Know`.
Una volta scaricato il file, assicuratevi di espandere la cartella utilizzando l'ultima versione di:

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

Convenzioni utilizzate

In questo libro vengono impiegate alcune convenzioni per il testo.
`CodiceNelTesto`: indica le parole in codice nel testo, i nomi delle tabelle del database, i nomi delle cartelle, i nomi dei file, le estensioni dei file, i nomi dei percorsi, gli URL, gli input dell'utente e gli handle di Twitter. Ecco un esempio: "Vediamo come aggiungere un nuovo elemento a uno stack utilizzando `push` o come rimuovere un elemento da uno stack utilizzando `pop`".

Un blocco di codice è formattato come segue:

```
define swap(x, y)
    buffer = x
    x = y
    y = buffer
```

Quando desideriamo attirare l'attenzione su un particolare di un blocco di codice, le righe o gli elementi pertinenti sono in grassetto:

```
define swap(x, y)
    buffer = x
    x = y
    y = buffer
```

Il *corsivo* indica i comandi nei menu, nuovi termini, parole importanti, titoli di libri o parole in altre lingue. Ecco un esempio: "Un modo per ridurre la complessità di un algoritmo è scendere a compromessi sulla sua accuratezza, producendo un tipo di algoritmo chiamato *algoritmo approssimato*".

> **NOTA**
>
> Gli avvertimenti, i suggerimenti di programmazione o le note importanti vengono segnalati in questo modo.

L'autore

Imran Ahmad è istruttore certificato Google e da diversi anni insegna per Google e Learning Tree. Fra gli argomenti da lui affrontati vi sono Python, il machine learning, gli algoritmi, i big data e il deep learning. Nel suo dottorato di ricerca, ha proposto un nuovo algoritmo basato sulla programmazione lineare chiamato ATSRA, che può essere utilizzato per assegnare in modo ottimale le risorse in un ambiente di cloud computing. Negli ultimi quattro anni ha collaborato a un progetto di alto profilo di machine learning presso il laboratorio di analisi avanzate del governo federale canadese. Il progetto consiste nello sviluppare algoritmi di machine learning in grado di automatizzare le procedure di esame delle domande di immigrazione. Imran sta attualmente lavorando allo sviluppo di algoritmi per utilizzare in modo ottimale le GPU per addestrare modelli complessi di machine learning.

Il revisore tecnico

Benjamin Baka è uno sviluppatore di software ed è appassionato di nuove tecnologie e di tecniche di programmazione. Ha dieci anni di esperienza in diverse tecnologie: C++, Java, Ruby, Python e Qt. Alcuni dei progetti cui sta lavorando si trovano sulla sua pagina GitHub. Attualmente sta operando su tecnologie entusiasmanti per mPedigree.