

Machine Learning Mit Python, Scikitlearn Und Tensorflow PDF (Begrenzte Kopie)

Sebastian Raschka



Kostenlose Testversion mit Bookey



Zum Herunterladen sc

Machine Learning Mit Python, Scikitlearn Und Tensorflow Zusammenfassung

Entfalte die Möglichkeiten des maschinellen Lernens mit Python.

Geschrieben von Berliner Checkpoint Kapitel Bücherclub

Kostenlose Testversion mit Bookey



Zum Herunterladen sca

Über das Buch

In einer von Daten dominierten Welt entfaltet das Machine Learning seine transformativen Fähigkeiten durch bahnbrechende Technologien, die unseren Alltag revolutionieren. Diese Technologien finden sich beispielsweise in der Sprachenerkennung von Smartphones oder in den personalisierten Empfehlungen von Streaming-Diensten.

Das Buch, das als umfassender Leitfaden fungiert, zielt darauf ab, die Grundprinzipien des Machine Learning mithilfe von Python zu vermitteln. Python hat sich als das bevorzugte Werkzeug zur Entwicklung effektiver Machine-Learning-Systeme etabliert, da es eine Vielzahl von Bibliotheken und Werkzeugen bietet, die den Lernprozess erleichtern.

Mit praktischen, schrittweisen Anleitungen werden die Leser in die Lage versetzt, Rohdaten in umsetzbare Erkenntnisse umzuwandeln. Der Prozess beginnt mit der Anwendung von effizienten Lernalgorithmen, deren Ergebnisse sorgfältig analysiert und bewertet werden. Während des Buches werden verschiedene Problemtypen im Machine Learning behandelt, darunter Klassifikation, Regression und Clustering – grundlegende Konzepte, die für die Analyse und Vorhersage von Daten unerlässlich sind.

Ein herausragendes Beispiel für das erlernte Wissen ist das Erstellen eines Sentiment-Analysetools, das emotionale Inhalte in Texten erkennen kann.

Kostenlose Testversion mit Bookey



Zum Herunterladen scannen

Die Leser erfahren, wie man dieses Modell in einer Webanwendung implementiert, was den praktischen Nutzen und die Anwendbarkeit von Machine Learning in realen Szenarien demonstriert.

Zusätzlich vertiefen die Leser ihr Verständnis durch Best Practices und reale Anwendungen, die ihnen zeigen, wie man das Potenzial von Daten optimal ausschöpfen kann. Das Buch ist somit nicht nur ein Lehrwerk, sondern auch ein wertvolles Nachschlagewerk für alle, die die Welt des Machine Learning erfolgreich navigieren möchten.

Kostenlose Testversion mit Bookey



Zum Herunterladen scannen

Über den Autor

In den folgenden Kapiteln wird Sebastian Raschka vorgestellt, ein leidenschaftlicher Data Scientist und Enthusiast im Bereich Machine Learning. Seine Faszination für das Aufdecken komplexer Muster in großen Datensätzen spiegelt sich in seiner aktuellen Dissertation an der Michigan State University wider, wo er an innovativer Software für virtuelles Screening in der computerunterstützten Wirkstoffentdeckung arbeitet. Dies beinhaltet die Untersuchung neuartiger Methoden für das Protein-Ligand-Docking, ein Prozess, der von zentraler Bedeutung für die Medikamentenentwicklung ist, da er das Verhalten von Proteinen und deren Wechselwirkungen mit möglichen Arzneistoffen analysiert.

Sebastians Herangehensweise an Data Science ist geprägt von einer starken Überzeugung für Teamarbeit und Open-Source-Zusammenarbeit. Er betont die Bedeutung des Wissensaustauschs und erkennt, dass konstruktives Feedback nicht nur das individuelle Wachstum fördert, sondern auch zur gemeinsamen Weiterentwicklung von Projekten beiträgt. Diese Philosophie steht im Einklang mit der sich schnell entwickelnden Welt der Data Science, wo Austausch und gemeinsames Lernen zu bahnbrechenden Ergebnissen führen können.

In seiner Freizeit zeigt sich Sebastian nicht nur als sportbegeisterter Mensch, der gerne Fußball und Tennis spielt, sondern auch als aktiver Mitgestalter



von Diskussionen über prädiktive Modellierung. Er zeigt Interesse daran, sich mit Gleichgesinnten auszutauschen und kreative Ideen zu entwickeln, was seine Offenheit für neue Ansätze und Perspektiven unterstreicht. Insgesamt bietet diese Einführung in Sebastians Lebenswelt einen umfassenden Überblick über seine Passionen, Überzeugungen und den Beitrag, den er zur Wissenschaft leistet.

Kostenlose Testversion mit Bookey



Zum Herunterladen scannen



Probieren Sie die Bookey App aus, um Zusammenfassungen von über 1000 der weltbesten Bücher zu lesen

1000+ Titel, 80+ Themen freischalten

Jede Woche werden neue Titel hinzugefügt



Einblicke in die weltbesten Bücher



Kostenlose Testversion mit Bookey



Inhaltsverzeichnis der Zusammenfassung

Kapitel 1: Fähigkeit, aus Daten zu lernen

Kapitel 2: Einfache ML-Algorithmen zur Klassifikation

Kapitel 3: Tour durch Klassifikatoren mit scikit-learn

Kapitel 4: Datenvorverarbeitung

Kapitel 5: Datenkompression durch Dimensionsreduktion

Kapitel 6: Best Practices für Modellbewertung &
Hyperparameteroptimierung

Kapitel 7: Verschiedene Modelle für Ensemble-Lernen kombinieren

Kapitel 8: Anwendung von ML auf die Sentiment-Analyse

Kapitel 9: Einbindung des ML Modells in die Webanwendung

Kapitel 10: Vorhersage kontinuierlicher Zielvariablen mit
Regressionsanalyse

Kapitel 11: Unbeschriftete Daten – Clusteranalyse

Kapitel 12: Mehrschichtiges ANN von Grund auf

Kapitel 13: Parallelisierung des Trainings von neuronalen Netzen mit
TensorFlow

Kapitel 14: Funktionsweise von TensorFlow

Kostenlose Testversion mit Bookey



Zum Herunterladen sc

Kapitel 15: Klassifizierung von Bildern mit tiefen konvolutionalen NNs

Kapitel 16: Modellierung sequenzieller Daten mit rekursiven NNs

Kostenlose Testversion mit Bookey



Zum Herunterladen sca

Kapitel 1 Zusammenfassung: Fähigkeit, aus Daten zu lernen

Kapitelübersicht: Den Computern die Fähigkeit geben, aus Daten zu lernen

In diesem Kapitel wird das spannenden Feld des Machine Learning (ML) vorgestellt, das es Computern ermöglicht, aus großen Datenmengen zu lernen und eigenständig Wissen abzuleiten. Machine Learning ist ein zentraler Bestandteil der Künstlichen Intelligenz (KI) und spielt eine wesentliche Rolle in Technologien, die wir täglich nutzen, wie Spamfilter, Spracherkennung und selbstfahrende Autos.

Hauptkonzepte von Machine Learning

Das Kapitel behandelt grundlegende Konzepte des Machine Learning, einschließlich den drei Hauptarten des Lernens: überwacht, unüberwacht und bestärkend. Diese Grundlagen sind entscheidend für die Entwicklung und Implementierung effektiver ML-Systeme.

1. Überwachtes Lernen: Hierbei wird ein Modell anhand gekennzeichneteter Daten trainiert. Das Ziel ist es, Muster zu erkennen und Vorhersagen für neue, unbekannte Daten zu treffen. Es umfasst zwei Hauptanwendungen:



- **Klassifikation:** Vorhersage von kategorischen Ergebnissen (z.B. Unterscheidung zwischen Spam und Nicht-Spam).
- **Regression:** Vorhersage kontinuierlicher Werte (z.B. die Prognose von SAT-Punkten basierend auf Lernzeit).

2. **Bestärkendes Lernen:** Bei diesem Ansatz trainiert ein Agent, indem er durch Interaktionen mit einer Umgebung lernt, wobei die Auswahl von Aktionen durch Belohnungen gesteuert wird. Beispiele umfassen die Strategien in Computerspielen.

3. **Unüberwachtes Lernen:** Dieses Lernen beschäftigt sich mit Daten ohne vorgegebene Labels und versucht, verborgene Strukturen zu entdecken. Wichtige Techniken sind:

- **Clusterbildung:** Organisation von Daten in Gruppen mit ähnlichen Eigenschaften.
- **Dimensionsreduktion:** Reduzierung der Komplexität hochdimensionaler Daten, um deren Analyse und Visualisierung zu erleichtern.

Terminologie und Notation

Das Kapitel erklärt die Nützlichkeit der linearen Algebra in ML, indem es zeigt, wie Daten in Matrizen und Vektoren dargestellt werden. Wichtige Begriffe wie Proben und Merkmale werden systematisch definiert, um das



Verständnis und die praktische Anwendung zu erleichtern.

Fahrplan zum Aufbau von Machine Learning-Systemen

Der Aufbau eines Machine Learning-Systems erfolgt in mehreren entscheidenden Schritten:

1. **Datenvorverarbeitung:** Rohdaten müssen gereinigt und in ein nutzbares Format umgewandelt werden.
2. **Modelltraining:** Verschiedene Algorithmen werden ausgewählt und optimiert, wobei Leistungskennzahlen wie Genauigkeit und Kreuzvalidierung berücksichtigt werden.
3. **Modellbewertung:** Beurteilung der Modellverallgemeinerung mithilfe von Testdatensätzen, um sicherzustellen, dass das Modell auch mit unbekannten Daten gut abschneidet.

Python als Werkzeug für Machine Learning

Python hat sich als die bevorzugte Programmiersprache für Data Science etabliert. Sie wird unterstützt von leistungsstarken Bibliotheken wie NumPy, SciPy sowie den ML-spezifischen Tools Scikit-learn und TensorFlow. Die Installation dieser Tools erfolgt typischerweise über die Befehlszeile (mit pip) oder durch die Verwendung der Anaconda-Distribution, die eine umfassende Umgebung für wissenschaftliches Rechnen bereitstellt.

Kostenlose Testversion mit Bookey



Zum Herunterladen scannen

Zusammenfassung des Kapitels

Insgesamt bietet dieses Kapitel ein grundlegendes Verständnis für die Thematik des Machine Learning und dessen vielfältige Arten von Lernansätzen. Es stellt die essenziellen Schritte zum Aufbau effektiver Systeme vor und legt so den Grundstein für vertiefte Themen wie Klassifikation, Regression und Clusterbildung in den folgenden Kapiteln. Durch die Integration von Python-Tools erhält der Leser praktisches Wissen, das für die Anwendung von Machine Learning in Echtzeitszenarien unverzichtbar ist.

Kostenlose Testversion mit Bookey



Zum Herunterladen sca

Kapitel 2 Zusammenfassung: Einfache ML-Algorithmen zur Klassifikation

In diesem Kapitel führen wir in die grundlegenden Machine Learning-Algorithmen zur Klassifikation ein, insbesondere den Perzeptron und die adaptiven linearen Neuronen (Adaline). Ziel ist es, ein tiefes Verständnis dieser Konzepte zu gewinnen und deren Implementierung in Python zu erlernen. Wir verwenden dabei den Iris-Datensatz, um die praktischen Anwendungen der Algorithmen zu veranschaulichen und uns auf fortgeschrittene Klassifizierer in der kommenden Lektion mit der scikit-learn Bibliothek vorzubereiten.

Grundlagen des Machine Learning

Zunächst erläutern wir die Intuition hinter Machine Learning-Algorithmen, gefolgt von einer Einführung in das Datenmanagement mit den Bibliotheken pandas, NumPy und Matplotlib. Diese Werkzeuge sind unverzichtbar für die Handhabung und Visualisierung von Daten in Machine Learning-Projekten.

Historischer Kontext der künstlichen Neuronen

Die Geschichte der künstlichen Neuronen beginnt 1943 mit Warren McCulloch und Walter Pitts, die das MCP-Neuron entwickelten, um das Verhalten biologischer Neuronen zu simulieren. Dieser Fortschritt inspirierte 1957 Frank Rosenblatt zur Entwicklung der Lernregel für das Perzeptron, die es Maschinen ermöglicht, aus Erfahrungen zu lernen und Gewichte für



Klassifizierungsentscheidungen zu optimieren.

Die Funktionsweise des Perzeptrons

Wir definieren eine Entscheidungsfunktion für die binäre Klassifikation, die eine lineare Kombination aus Eingabewerten und Gewichtungsvektoren nutzt. Der Perzeptron-Algorithmus sagt mithilfe einer Einheitssprungfunktion Klassennamen voraus. Rosenblatts Lernregel umfasst die Initialisierung von Gewichten, die Vorhersage von Klassennamen und die Anpassung der Gewichte basierend auf Vorhersagefehlern – Fehlklassifikationen führen dazu, dass die Gewichte näher an die Zielklasse angepasst werden, was die Genauigkeit verbessert.

Implementierung und Training des Perzeptrons

Zur praktischen Anwendung erstellen wir eine objektorientierte API in Python, um Perzeptron-Objekte zu initialisieren, die aus Daten lernen und Vorhersagen treffen. Die Demo erfolgt mit dem Iris-Datensatz, aus dem wir zwei Klassen zur Klassifikation extrahieren. Wir visualisieren die Daten und die lineare Entscheidungsgrenze, um die Leistungsfähigkeit des Perzeptrons zu demonstrieren.

Einführung in Adaline

Die adaptive Version des Perzeptrons, Adaline, verbessert die Gewichtsanzpassung, indem sie eine lineare Aktivierungsfunktion verwendet. Hierdurch wird die Minimierung der kontinuierlichen Kostenfunktionen,



insbesondere der Summe der quadrierten Fehler (SSE), zur Gewichtsoptimierung ermöglicht. Diese Methodik stellt den Übergang zu fortgeschritteneren Klassifizierern dar.

Optimierung der Kostenfunktion

Wir besprechen, wie Gradient Descent zur Minimierung der Kostenfunktionen genutzt wird, um die Gewichte zu aktualisieren und die Fehler über die Trainingsmuster zu minimieren. Eine wichtige Rolle spielt dabei das Feature-Scaling, insbesondere die Standardisierung, welche die Konvergenzzeit drastisch reduziert. Wir demonstrieren, wie standardisierte Merkmale die Effizienz während des Adaline-Trainings verbessern können.

Stochastischer Gradient Descent

Ein tiefergehendes Verständnis wird durch die Einführung von stochastischem Gradient Descent erreicht, das eine inkrementelle Gewichts Anpassung pro Beispiel ermöglicht und somit eine schnellere Konvergenz bei großen Datensätzen sicherstellt. Mini-Batch-Lernen vereint die Vorteile beider Ansätze.

Umsetzung und Visualisierung von Adaline SGD

In der letzten Phase implementieren wir die AdalineSGD-Klasse, die den stochastischen Gradient Descent umsetzt. Mit zusätzlichen Funktionen zur Datenmischung und partiellen Anpassung visualisieren wir die Trainingsergebnisse sowie die durchschnittlichen Kosten über die Epochen.



Fazit

Zusammenfassend haben wir die grundlegenden linearen Klassifizierer in Python implementiert und deren Trainingsprozesse nachvollzogen. Dieses erkannte Wissen wird eine wertvolle Grundlage bieten, um in den folgenden Kapiteln fortgeschrittenere Klassifizierer wie die logistische Regression und Support Vector Machines zu erforschen, wodurch wir unser Machine Learning-Repertoire erheblich erweitern können.



Kapitel 3 Zusammenfassung: Tour durch Klassifikatoren mit scikit-learn

Eine Tour durch Machine Learning-Klassifikatoren mit scikit-learn

In diesem Kapitel wird ein Überblick über populäre Machine Learning-Klassifikatoren gegeben, die sowohl in akademischen als auch in industriellen Anwendungen weit verbreitet sind. Die Leserschaft wird mit der scikit-learn-Bibliothek vertraut gemacht, die eine benutzerfreundliche Schnittstelle zur Implementierung der gängigsten Algorithmen bietet.

Wichtige Lerninhalte

Das Kapitel bietet Einblicke in verschiedene Klassifikationsalgorithmen, darunter die logistische Regression, Unterstützung von Vektormaschinen (SVMs) und Entscheidungsbäume. Es werden praktische Beispiele vorgestellt, die die Stärken und Schwächen dieser Algorithmen beleuchten, insbesondere im Hinblick auf ihre Entscheidungsgrenzen.

Auswahl eines Klassifikationsalgorithmus

Die Auswahl des richtigen Klassifikators ist entscheidend, da das No Free Lunch-Theorem besagt, dass kein einzelner Algorithmus für alle Probleme optimal ist. Der Prozess umfasst mehrere Schritte: die Auswahl relevanter Merkmale, die Erhebung und Aufbereitung von Trainingsdaten, die



Bestimmung geeigneter Leistungskennzahlen, die Wahl des Klassifikators sowie die anschließende Bewertung und Feinabstimmung des Modells. Diese grundlegenden Schritte bilden das Fundament für das Verständnis komplexerer Themen wie Merkmalsauswahl und Hyperparameter-Tuning, die in späteren Kapiteln behandelt werden.

Erste Schritte mit scikit-learn: Training eines Perzeptrons

Das Kapitel zeigt, wie man ein Perzeptron-Modell mithilfe von scikit-learn mit dem bekannten Iris-Datensatz trainiert. Die Schritte umfassen das Einlesen und Vorverarbeiten der Daten, das Aufteilen in Trainings- und Testdaten, die Standardisierung der Merkmale sowie das Trainieren des Modells. Schließlich wird die Modellleistung mittels Genauigkeitsmetriken bewertet.

Logistische Regression: Klassenwahrscheinlichkeiten und Kostenfunktionen

Es folgt eine Einführung in die logistische Regression, ein Modell, das besonders für binäre Klassifikationsaufgaben geeignet ist. Hauptmerkmale sind die Nutzung der Sigmoidfunktion zur Schätzung von Klassenwahrscheinlichkeiten und die Anwendung der Log-Likelihood-Funktion zur Optimierung von Modellen, was einen Übergang von klassischen quadrierten Fehlern zu einer Wahrscheinlichkeits-basierten Kostenfunktion darstellt. Die Implementierung in scikit-learn ist für sowohl binäre als auch multiklassige



Probleme ausgelegt.

Unterstützende Vektormaschinen (SVM)

SVMs erweitern das Konzept des Perzeptrons, indem sie sich auf die Maximierung des Abstands zwischen Klassen konzentrieren. Dies ist besonders nützlich für nicht-linear trennbare Fälle, da SVMs Slack-Variablen nutzen können. Die Technik wird anhand des Iris-Datensatzes demonstriert.

K-Nächste Nachbarn (KNN)

KNN, ein nichtparametrischer Algorithmus, bewertet die Klassifizierung basierend auf den Mehrheitsstimmen von Nachbarn. Besondere Aufmerksamkeit gilt den Auswirkungen der gewählten Distanzmetriken und der Auswahl des Parameters k . Zudem wird der Fluch der dimensionalen Komplexität angesprochen, der die Leistungsfähigkeit des Algorithmus beeinflussen kann.

Ensemble-Methoden: Random Forests

Random Forests kombinieren die Entscheidungen mehrerer Entscheidungsbäume, um die Robustheit gegenüber Überanpassung zu erhöhen und die Vorhersagegenauigkeit zu verbessern. Der Prozess umfasst das Ziehen zufälliger Bootstrap-Stichproben, den Aufbau von Entscheidungsbäumen auf der Grundlage einer Teilmenge von Merkmalen und die Aggregation der Vorhersagen mittels Mehrheitsabstimmung.



Entscheidungsbäume

Entscheidungsbäume zeichnen sich durch ihre Transparenz im Entscheidungsprozess aus, während gleichzeitig das Risiko der Überanpassung minimiert werden muss. Dies geschieht durch die Begrenzung der Baumtiefe und die Verwendung geeigneter Unreinheitsmaße wie Gini-Unreinheit oder Entropie.

Fazit

Das Kapitel schließt mit der Erkenntnis, dass ein tiefes Verständnis der verwendeten Algorithmen und deren Funktionsweise von zentraler Bedeutung ist. Die Qualität der Daten spielt ebenfalls eine entscheidende Rolle im Lernprozess. Im nächsten Kapitel werden Datenvorverarbeitungstechniken behandelt, die entscheidend sind, um robuste Machine Learning-Modelle zu entwickeln.



Kapitel 4: Datenvorverarbeitung

In diesem Kapitel über die Erstellung guter Trainingssätze und deren Datenvorverarbeitung wird die entscheidende Rolle der Datenqualität für den Erfolg von Machine-Learning-Algorithmen hervorgehoben. Um sicherzustellen, dass die Modelle effektiv funktionieren, werden verschiedene Techniken zur Datenvorverarbeitung erläutert, die sich auf den Umgang mit fehlenden Werten, die Formatierung kategorischer Daten und die Merkmalsauswahl konzentrieren.

Umgang mit fehlenden Daten ist ein grundlegender Aspekt, da diese in realen Datensätzen häufig vorkommen, sei es durch Erhebungsfehler oder unvollständige Antworten. Fehlende Werte, oft als NaN oder NULL gekennzeichnet, können die Analyse behindern. Es werden Methoden zur Identifizierung dieser Werte sowie Techniken zur Imputation vorgestellt, beispielsweise durch Mittelwertimputation unter Verwendung der ``Imputer``-Klasse von scikit-learn.

Kategorische Daten werden in nominale und ordinale Merkmale unterteilt und erfordern unterschiedliche Verarbeitungsansätze. Ordinale Daten können in numerische Werte umgewandelt werden, während nominale Daten durch Techniken wie ``LabelEncoder`` und One-Hot-Encoding kodiert werden, um Missverständnisse in der Analyse zu vermeiden.



Ein weiterer zentraler Punkt ist die **Partitionierung des Datensatzes** in Trainings- und Testsets. Durch die Verwendung der Funktion ``train_test_split`` aus scikit-learn kann die Verteilung der Klassenverhältnisse aufrechterhalten werden, was für eine verlässliche Bewertung des Modells von Bedeutung ist.

Die **Merkmals-Skalierung** stellt sicher, dass alle Merkmale ähnliche Wertebereiche haben, was die Leistung vieler Algorithmen optimiert. Techniken wie die Normalisierung (Min-Max-Skalierung) und die Standardisierung (Anpassung an einen Mittelwert von 0 und eine Standardabweichung von 1) sind hier entscheidend.

Zur Vermeidung von Überanpassung und zur Reduzierung der Modellkomplexität kommen verschiedene **Methoden zur Auswahl bedeutungsvoller Merkmale** zum Einsatz. Dazu zählen L1- und L2-Regularisierung, die komplexe Modelle bestrafen, sowie eine sequenzielle Merkmalsauswahl, wie die sequenzielle Rückwärtsauswahl (SBS), die unwichtige Merkmale eliminiert.

Entscheidungsbäume, insbesondere durch den Einsatz von Zufallswäldern, bieten eine robuste Möglichkeit zur **Bewertung der Merkmalsbedeutung**. Diese Methode hilft dabei, die relevantesten Merkmale im Datensatz zu identifizieren und zu priorisieren, was die Effizienz und Genauigkeit des Modells steigert.



Abschließend fasst das Kapitel zusammen, dass effektive Datenvorverarbeitung entscheidend für das Training leistungsfähiger Machine-Learning-Modelle ist. In den nächsten Kapiteln wird das Thema der Merkmals-Extraktion behandelt, um Daten auf niederdimensionale Teilräume zu komprimieren, was das Modell weiter verfeinert und verbessert.





Installieren Sie die Bookey App, um den Volltext und Audio freizuschalten

Kostenlose Testversion mit Bookey





Warum Bookey eine unverzichtbare App für Buchliebhaber ist

-  **30min Inhalt**
Je tiefer und klarer unsere Interpretation ist, desto besser verstehen Sie jeden Titel.
-  **3min Idee-Clips**
Steigere deinen Fortschritt.
-  **Quiz**
Überprüfen Sie, ob Sie das soeben Gelernte beherrschen.
-  **Und mehr**
Mehrere Schriftarten, Laufende Pfade, Sammlungen...

Kostenlose Testversion mit Bookey



Kapitel 5 Zusammenfassung: Datenkompression durch Dimensionsreduktion

In diesem Kapitel wird die Bedeutung der Dimensionsreduktion in der Datenanalyse und maschinellem Lernen hervorgehoben. Diese Techniken sind entscheidend für die effiziente Speicherung, Verarbeitung und Analyse von großen Datensätzen. Es werden drei zentrale Methoden vorgestellt, die jeweils unterschiedliche Ansätze zur Reduktion der Datenmenge und zur Verbesserung der Analyse bieten.

Zu Beginn erfolgt eine Übersicht über die drei Haupttechniken der Dimensionsreduktion:

1. **Hauptkomponentenanalyse (PCA):** Eine unüberwachte Methode, die genutzt wird, um die wichtigsten Merkmale eines Datensatzes basierend auf den Korrelationen der Variablen zu extrahieren.
2. **Lineare Diskriminanzanalyse (LDA):** Eine überwachte Technik, die speziell auf die Maximierung der Klassenabtrennung ausgerichtet ist und häufig in Klassifikationsaufgaben verwendet wird.
3. **Kern-Hauptkomponentenanalyse (KPCA):** Ein nicht-linearer Ansatz, der sich eignet, wenn einfache Klassifikationen nicht ausreichen, um komplexe Muster zu erfassen.



Unüberwachte Dimensionsreduktion mittels PCA

Die PCA arbeitet, indem sie die Beziehungen zwischen Merkmalen untersucht und versucht, die Richtungen mit der höchsten Varianz zu finden. Der Prozess umfasst die Standardisierung der Daten, das Erstellen einer Kovarianzmatrix, die Zerlegung in Eigenvektoren und Eigenwerte, die Auswahl von 'k' relevanten Eigenvektoren und die Projektion der Daten in einen neuen Merkmalsraum. Ein praktisches Beispiel der PCA wird mit einem Weindatensatz veranschaulicht, wobei die transformierten Daten durch Streudiagramme in zwei Dimensionen visualisiert werden, um die Effektivität der Methode in der Dimensionsreduktion zu demonstrieren.

Überwachte Datenkompression mittels LDA

Die LDA fokussiert sich auf die Klassifikation, indem sie Mittelwertvektoren für jede Klasse berechnet und Streuungsmatrizen sowohl innerhalb als auch zwischen den Klassen konstruiert. Durch die Diagonalisierung dieser Matrizen entstehen lineare Diskriminanten, die die Merkmalsextraktion optimieren. Dieser Ansatz ist besonders nützlich in Anwendungsszenarien, in denen die Trennung der Klassen für die Klassifikationsleistung von Bedeutung ist.

Kern-PCA: Umgang mit Nichtlinearität



Die KPCA wird verwendet, wenn die Daten in ihren ursprünglichen Dimensionen nicht linear trennbar sind. Sie wendet eine Kernel-Funktion an, um die Daten in einen höherdimensionalen Raum zu projizieren, in dem eine lineare Trennung möglich ist. Die Implementierung der KPCA erfordert die Berechnung von Kernmatrizen, deren Zentrierung und die Extraktion der entsprechenden Eigenvektoren. Anhand synthetischer Datensätze, wie Mond- und konzentrischen Kreisen, wird die Wirksamkeit der KPCA illustriert, die die Trennbarkeit linearer Klassen in transformierten Daten aufzeigt.

Fazit

Das Kapitel fasst die drei grundlegenden Techniken der Dimensionsreduktion—PCA, LDA und KPCA—zusammen und gibt Praktikern wertvolle Kenntnisse, um Daten effektiv aufzubereiten. Darüber hinaus werden die Leser befähigt, Modelle zu evaluieren und Hyperparameter zu optimieren, um den Herausforderungen im Bereich des maschinellen Lernens gerecht zu werden. Diese Methoden sind essenziell für jeden, der komplexe Datenanalysen durchführt oder maschinelles Lernen anwendet.



Kapitel 6 Zusammenfassung: Best Practices für Modellbewertung & Hyperparameteroptimierung

Zusammenfassung der besten Praktiken für die Modellbewertung und Hyperparameteroptimierung

In diesem Kapitel werden effektive Strategien zur Entwicklung robuster Machine-Learning-Modelle behandelt, die insbesondere auf die Bewertung und Optimierung von Hyperparametern abzielen. Die zentralen Aspekte umfassen die Gewährleistung unverzerrter Leistungsschätzungen, die Diagnose typischer Probleme mit Algorithmen, das Feintuning von Modellen sowie die Anwendung unterschiedlicher Leistungskennzahlen zur Bewertung der Modellqualität.

Workflow-Optimierung mit Pipelines

Ein wichtiger Schritt in diesem Prozess ist die Verwendung der Pipeline-Klasse aus der Bibliothek scikit-learn. Mit Pipelines lassen sich Vorverarbeitungsschritte — wie Datenbereinigung und -normalisierung — sowie Modellanpassungen präzise und effizient verknüpfen. Dies gewährleistet die konsistente Anwendung von auf Trainingsdaten basierenden Parametern auf neue, unbekannte Daten, was die Effizienz des gesamten Modells erhöht.

Laden des Breast Cancer Wisconsin Datensatzes



Der Breast Cancer Wisconsin Datensatz dient als praktisches Beispiel und umfasst 569 Tumormuster mit 30 Merkmalen, die mit der Aggressivität von Tumoren korrelieren. Um die Daten zu strukturieren, werden diese mit Hilfe von Pandas geladen und in Trainings- (80 %) und Testdaten (20 %) unter Zuhilfenahme der `train_test_split`-Funktion von scikit-learn aufgeteilt.

Kombination von Transformatoren und Schätzern in einer Pipeline

Die Integration von Transformatoren, wie dem StandardScaler zur Merkmalsnormierung oder Principal Component Analysis (PCA) zur Dimensionsreduktion, mit Vorhersagemodellen wie der logistischen Regression vereinfacht die Modellanpassung und -vorhersage erheblich. Dies liefert ein umfassenderes Bild der Leistungsfähigkeit des Modells.

Verwendung von k-facher Kreuzvalidierung zur Bewertung der Modellleistung

Kreuzvalidierung spielt eine entscheidende Rolle bei der Einschätzung der Generalisierungsfähigkeit von Modellen. Insbesondere die Holdout-Methode, die in Trainings-, Validierungs- und Testdatensätze unterteilt, hilft, Overfitting zu vermeiden und realistische Leistungsschätzungen zu erzielen.

K-fache Kreuzvalidierung erklärt

Bei der k-fachen Kreuzvalidierung wird der Datensatz in k Teilmengen aufgeteilt. Das Modell wird auf k-1 Falten trainiert und in der restlichen



Falte validiert, wodurch dieser Vorgang k Mal wiederholt wird, um eine präzise durchschnittliche Leistungsbewertung zu ermöglichen.

Fehlerbehebung von Algorithmen mit Lern- und Validierungskurven
Lernkurven sind nützliche Werkzeuge zur Diagnose von Overfitting und Underfitting, indem sie die Modellgenauigkeit in Bezug auf die Trainingsdaten visualisieren. Validierungskurven hingegen helfen dabei, die Auswirkungen von Hyperparametern auf die Modellleistung zu verstehen.

Feintuning von Machine-Learning-Modellen durch Grid Search
Um die Hyperparameter eines Modells zu optimieren, wird die Methode der Grid Search eingesetzt. Diese Methode ermöglicht eine umfassende Erkundung verschiedener Kombinationen von Hyperparametern, sodass die besten Parameter und deren Ergebnisse nach dem Training einfach abgerufen werden können.

Algorithmusauswahl mit verschachtelter Kreuzvalidierung
Die verschachtelte Kreuzvalidierung kombiniert Modellselektion und -validierung auf strukturierte Weise. Hierbei wird die äußere k -fache Kreuzvalidierung eingesetzt, um die Leistung zu testen, während die innere dafür dient, das Modell zu trainieren. Dies führt zu einer robusteren und ergebnisgenaueren Leistungsbewertung.

Berücksichtigung verschiedener Leistungsbewertungsmethoden



Neben der Genauigkeit sind auch andere Metriken wichtig, um die Modellqualität zu bewerten. Dazu zählen Präzision, Recall, der F1-Score und die ROC-AUC, die eine vielschichtige Sicht auf die Fähigkeiten des Modells bieten.

Umgang mit Klassenungleichgewicht

Ein häufiges Problem in der maschinellen Lernpraxis ist das Klassenungleichgewicht. Um sicherzustellen, dass Modelle in sämtlichen Klassen gut abschneiden, kommen verschiedene Techniken zum Einsatz. Dazu gehören die Anpassung der Klassen gewichtungen, Resampling-Methoden (Upsampling und Downsampling), sowie die Generierung synthetischer Muster mithilfe von SMOTE (Synthetic Minority Over-sampling Technique).

Zusammenfassung

Das Kapitel bietet den Lesern wertvolle Techniken zur Bewertung und Verbesserung von Machine-Learning-Modellen. Es vermittelt grundlegendes Wissen über die Nutzung von Pipelines, Kreuzvalidierung und verschiedene Leistungskennzahlen und legt damit den Grundstein für die Erkundung von Ensemble-Methoden im nächsten Kapitel.



Kapitel 7 Zusammenfassung: Verschiedene Modelle für Ensemble-Lernen kombinieren

Kapitel 7: Verschiedene Modelle für Ensemble-Lernen kombinieren

Überblick

In diesem Kapitel verschieben wir den Fokus von der Feinabstimmung individueller Klassifikatoren hin zu Ensemble-Lernmethoden. Diese Methoden kombinieren mehrere Klassifikatoren, um die prädiktive Leistung zu steigern. Wir werden die Grundlagen der Mehrheitsabstimmung, Bagging und Boosting kennenlernen.

Ensemble-Lernen

Das zentrale Ziel von Ensemble-Methoden ist die Entwicklung eines Meta-Klassifikators, der in der Lage ist, allgemeinere Vorhersagen zu treffen als individuelle Klassifikatoren. Zu den gängigen Techniken gehören:

- **Mehrheitsabstimmung:** Hierbei wird die Klasse gewählt, die von der Mehrheit der Klassifikatoren prognostiziert wird.
- **Pluralitätsabstimmung:** Eine Erweiterung der Mehrheitsabstimmung, die für die Mehrklassenklassifikation Anwendung findet.



- **Bagging:** Diese Technik verwendet unterschiedliche zufällige Ziehungen von Trainingsdatensätzen, um Überanpassung zu vermeiden.
- **Boosting:** Diese Strategie verbindet schwache Lernmodelle, die aus vergangenen Fehlern lernen, um robuste Modelle zu erstellen.

Ansatz der Mehrheitsabstimmung

Das Konzept der Mehrheitsabstimmung ist simpel und effektiv. Hierbei werden mehrere Klassifikatoren genutzt, um Klassenlabels vorherzusagen, wobei die finale Entscheidung auf der Klasse basiert, die die meisten Stimmen erhält. Anhand eines binären Klassifikationsbeispiels wird demonstriert, wie Ensemble-Vorhersagen zulassen, Fehlerquoten im Vergleich zu Einzelklassifikatoren zu senken.

Implementierung des Mehrheitsabstimmungs-Klassifikators

In dieser Sektion wird ein einfacher Ensemble-Klassifikator implementiert, der die Mehrheitsabstimmung anwendet. In Python umgesetzt, kombiniert der Algorithmus Vorhersagen durch gewichtete Mehrheitsabstimmung, wobei auch Klassenstimmanteile zur Verbesserung der Genauigkeit einbezogen werden können.

Bagging



Bagging, kurz für Bootstrap-Aggregation, erlaubt die Erstellung mehrerer Klassifikatoren aus verschiedenen Bootstrap-Proben der Trainingsdaten. Diese Methode ist besonders nützlich für instabile Modelle und trägt dazu bei, Überanpassung zu reduzieren. Ein praktisches Beispiel im Kontext eines Wein-Datensatzes verdeutlicht, wie Bagging die Leistung von Klassifikatoren verbessern kann.

Boosting mit AdaBoost

Boosting fokussiert sich darauf, schwache Klassifikatoren gezielt auf schwer klassifizierbare Proben zu trainieren. AdaBoost ist eine der bekanntesten Boosting-Methoden. Sie modifiziert die Gewichte der Proben im Training auf Grundlage der vorherigen Klassifikationsergebnisse, was zur Stärkung des Gesamtklassifikators beiträgt. Ein Beispiel mit AdaBoost auf dem Wein-Datensatz zeigt signifikante Leistungsverbesserungen im Vergleich zu Einzelentscheidungsbäumen.

Fazit

Ensemble-Techniken wie Mehrheitsabstimmung, Bagging und Boosting tragen dazu bei, die Modellleistung zu erhöhen, indem sie die Vorzüge mehrerer Klassifikatoren kombinieren und deren Schwächen minimieren. Allerdings bringen diese Methoden eine erhöhte rechnerische Komplexität mit sich und erfordern eine sorgfältige Überlegung hinsichtlich ihrer



praktischen Einsatzmöglichkeiten.

Nächste Schritte

Im kommenden Kapitel werden wir uns mit der Sentiment-Analyse beschäftigen, einer zunehmend relevanten Anwendung im Bereich des Machine Learnings, insbesondere im Kontext von Internet und sozialen Medien.

Kostenlose Testversion mit Bookey



Zum Herunterladen sca

Kapitel 8: Anwendung von ML auf die Sentiment-Analyse

In diesem Kapitel wird die Anwendung von Machine Learning auf die Sentiment-Analyse behandelt, ein bedeutendes Teilgebiet der natürlichen Sprachverarbeitung (NLP). Der Schwerpunkt liegt auf der Analyse eines Datensatzes von 50.000 Filmkritiken der Internet Movie Database (IMDb), der uns hilft, zwischen positiven und negativen Bewertungen zu unterscheiden. Positive Kritiken haben über sechs Sterne erhalten, während negative weniger als fünf Sterne erhalten.

Zunächst wird der Prozess der Datenbeschaffung und -vorbereitung erläutert. Der Datensatz wird heruntergeladen, extrahiert und von Rohdaten in ein strukturiertes Format, ein CSV-Dateiformat, umgewandelt. Dabei wird das ursprüngliche Format der Filmkritiken gesäubert und als DataFrame eingelesen. Wichtige Schritte in diesem Prozess sind das Initialisieren einer Fortschrittsanzeige und das Mischen der Daten vor der Speicherung.

Weiterhin wird das Bag-of-Words-Modell vorgestellt, ein Verfahren, das es ermöglicht, Wörter in numerische Merkmalsvektoren umzuwandeln, was notwendig ist, damit Machine-Learning-Algorithmen Texte verarbeiten können. Mithilfe des `CountVectorizer` aus der Python-Bibliothek scikit-learn wird ein Vokabular erstellt und die Häufigkeit von Wörtern gezählt, was zu spärlichen Merkmalsvektoren führt.



Ein zentraler Bestandteil der Textverarbeitung ist die Bewertung der Wortrelevanz. Hier kommt die Technik der Termfrequenz-inversen Dokumentenfrequenz (tf-idf) ins Spiel, die dafür sorgt, dass häufige, nicht-diskriminierende Wörter einen geringeren Einfluss auf die Vorhersagen des Modells haben. Die Implementierung des `TfidfTransformer` in scikit-learn wird gezeigt, um diese Umwandlung zu verdeutlichen.

Die Datenbereinigung ist ein weiterer wichtiger Schritt, um unerwünschte Zeichen zu entfernen und die Qualität der Texte zu verbessern. Dazu wird eine Methode zur Vorverarbeitung erläutert, die HTML-Markup und Interpunktion entfernt und gleichzeitig nützliche Emoticons beibehält.

Ein weiterer Abschnitt beschäftigt sich mit der Tokenisierung, die Texte in einzelne Wörter zerlegt. Hierbei wird das Stemming behandelt, unterstützt durch die NLTK-Bibliothek, die verschiedene Stemming-Algorithmen, einschließlich des Porter-Stemmers, vorstellt.

Im Anschluss wird dargelegt, wie ein logistisches Regressionsmodell zur Dokumentklassifizierung trainiert wird. Der Datensatz wird in Trainings- und Testsets aufgeteilt, und es wird `GridSearchCV` verwendet, um die Hyperparameter des Modells zu optimieren. Die Genauigkeit des Modells wird schließlich anhand des Testdatensatzes bewertet.



Für die Verarbeitung größerer Datensätze wird das Konzept des Out-of-Core-Lernens eingeführt. Dies ermöglicht eine inkrementelle Ausbildung der Klassifizierer mit kleineren Datenmengen, ohne diese vollständig im Speicher zu laden. Dies wird durch den Einsatz von ``HashingVectorizer`` und ``SGDClassifier`` demonstriert.

Installieren Sie die Bookey App, um den Volltext und Audio freizuschalten

Kostenlose Testversion mit Bookey





22k 5-Sterne-Bewertungen

Positives Feedback

Anselm Krause

der Buchzusammenfassung
ändnis, sondern machen den
sam und fesselnd. Bookey
lich neu erfunden.

Fantastisch!



Ich bin erstaunt über die Vielfalt an Büchern und Sprachen, die Bookey unterstützt. Es ist nicht nur eine App, es ist ein Tor zum globalen Wissen. Pluspunkte für das Sammeln von Punkten für wohltätige Zwecke!

Bärbel Müller

Fi



Di
Bu
Ve
Le

n Schröder

ufstelle
n. Die
prägnant,
rschön

Liebe es!



Bookey zu nutzen hat mir geholfen, eine Lesegewohnheit zu entwickeln, ohne meinen Zeitplan zu überlasten. Das Design der App und ihre Funktionen sind benutzerfreundlich und machen intellektuelles Wachstum für jeden zugänglich.

Dietmar Beckenbauer

Zeitsparer!



Bookey bietet mir
durchzugehen. Es
ob ich die ganze
Es ist einfach zu b

Tolle App!



Ich liebe Hörbücher, habe aber nicht immer Zeit, das ganze Buch zu hören! Bookey ermöglicht es mir, eine Zusammenfassung der Highlights des Buches zu bekommen, für das ich mich interessiere!!!

Elke Lang

Schöne App



Diese App ist ein Lebensretter für Buchliebhaber. Sie hilft bei vollen Terminkalendern. Die Zusammenfassungen sind genau auf den Punkt, und die Mind-Maps helfen, das Gelernte zu verstärken. Sehr zu empfehlen!

Kostenlose Testversion mit Bookey



Kapitel 9 Zusammenfassung: Einbindung des ML Modells in die Webanwendung

In diesem Kapitel wird die Integration von Machine Learning Modellen in Webanwendungen behandelt. Machine Learning kann die Nutzererfahrung in Online-Diensten erheblich verbessern, indem es Aufgaben wie Spam-Erkennung und personalisierte Empfehlungen automatisiert. Das Kapitel zielt darauf ab, ein Modell einzubetten, das in der Lage ist, Echtzeitdaten zu klassifizieren und kontinuierlich zu lernen.

Themenübersicht

- **Modell-Serialisierung:** Um Ressourcen zu sparen, ist es wichtig, trainierte Modelle für die zukünftige Nutzung zu speichern. Das Python-Modul ``pickle`` ermöglicht die Serialisierung von Modellen in ein Bytecode-Format, was eine einfache Speicherung und Wiederherstellung ohne erneutes Training erlaubt.

Erstellung eines Serializers:

- Zunächst muss sichergestellt werden, dass das trainierte Modell bereit ist.
- ``pickle`` wird verwendet, um das Modell und damit verbundene Objekte zu speichern. Diese werden in einem geeigneten Verzeichnis für die Webanwendung abgelegt.



```

```python
import pickle
import os

dest = os.path.join('movieclassifier', 'pkl_objects')
if not os.path.exists(dest):
 os.makedirs(dest)
pickle.dump(stop, open(os.path.join(dest, 'stopwords.pkl'), 'wb'),
protocol=4)
pickle.dump(clf, open(os.path.join(dest, 'classifier.pkl'), 'wb'), protocol=4)
```

```

- **Vectorizer-Nutzung** Ein `HashingVectorizer` wird implementiert, um Dokumente zur Vorbereitung für das Modell zu verarbeiten. Um die Organisation zu erleichtern, wird dieses in einer separaten Datei, `vectorizer.py`, gespeichert.

SQLite-Datenbankeinrichtung

SQLite stellt eine leichte Datenbanklösung dar, die ideal für kleinere Projekte ist. In diesem Abschnitt wird beschrieben, wie eine SQLite-Datenbank eingerichtet werden kann, um Nutzerfeedback zu speichern, das für die Verbesserung des Machine Learning Modells genutzt werden kann.



```
```python
import sqlite3
import os

if os.path.exists('reviews.sqlite'):
 os.remove('reviews.sqlite')

conn = sqlite3.connect('reviews.sqlite')
c = conn.cursor()
c.execute('CREATE TABLE review_db (review TEXT, sentiment
INTEGER, date TEXT)')

Beispielpbewertungen hinzufügen
example1 = 'Ich liebe diesen Film'
c.execute("INSERT INTO review_db (review, sentiment, date) VALUES (?,
?, DATETIME('now'))", (example1, 1))

example2 = 'Ich mochte diesen Film nicht'
c.execute("INSERT INTO review_db (review, sentiment, date) VALUES (?,
?, DATETIME('now'))", (example2, 0))

conn.commit()
conn.close()
```
```



Entwicklung mit Flask

Flask ist ein beliebtes Webframework für Python, das die Entwicklung von Webanwendungen erleichtert, indem es eine unkomplizierte Integration mit Python-Code ermöglicht.

Erstellung einer einfachen Flask-Anwendung:

- Zuerst wird ein Projektverzeichnis mit einer `app.py`-Datei und einem `templates/`-Verzeichnis erstellt.
- Flask wird initialisiert, und ein einfaches Formular zur Erfassung von Benutzereingaben, wie Namen, wird erstellt.

Inhalt der `app.py`:

```
```python
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
 return render_template('first_app.html')

if __name__ == '__main__':
```



```
app.run()
```

```
...
```

### **Inhalt von `first\_app.html`:**

```
```html
```

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
  <title>Erste App</title>
```

```
</head>
```

```
<body>
```

```
  <div>Hallo, dies ist meine erste Flask-Web-App!</div>
```

```
</body>
```

```
</html>
```

```
...
```

Diese einfache Anwendung verdeutlicht die Funktionalität von Flask-Routen und die dynamische Erstellung von HTML-Vorlagen basierend auf Benutzereingaben. Die im Kapitel vermittelten Grundlagen befähigen Leser dazu, komplexere Webanwendungen zu entwickeln, die Machine Learning Modelle integrieren und so die Anwendungsmöglichkeiten verbessern.



Kapitel 10 Zusammenfassung: Vorhersage kontinuierlicher Zielvariablen mit Regressionsanalyse

Kapitelzusammenfassung: Vorhersage kontinuierlicher Zielvariablen mit Regressionsanalyse

Dieses Kapitel befasst sich mit der Regressionsanalyse, einem Teilgebiet des überwachten Lernens, das sich auf die Vorhersage kontinuierlicher Zielvariablen konzentriert. Im Gegensatz zur Klassifikation, die kategoriale Ergebnisse behandelt, dient die Regressionsanalyse nicht nur der Vorhersage, sondern auch dem Verständnis von Beziehungen zwischen Variablen sowie der Bewertung von Trends. Ein praktisches Beispiel ist die Vorhersage zukünftiger Verkaufszahlen.

Zentrale Konzepte der Regressionsanalyse:

- 1. Datenexploration:** Eine gründliche visuelle und statistische Erkundung von Datensätzen ist entscheidend, um Zusammenhänge zu erkennen.
- 2. Lineare Regression:** Es gibt einfache und multiple Formen, wobei die einfache lineare Regression die Beziehung zwischen einer erklärenden Variablen und einer Zielvariablen mittels der Gleichung $y = w_0 + w_1 x$ beschreibt, während die multiple Regression mehrere Variablen



berücksichtigt: $y = w_0 + \sum_{i=1}^m w_i x_i$.

Der Wohnungsdatensatz: Ein klassisches Beispiel in der Regressionsanalyse, das verschiedene Merkmale zu Wohnpreisen vereint. Hier wird empfohlen, visuelle Exploration, wie Streudiagrammmatrizen, und statistische Techniken zur Untersuchung der Datenbeziehungen einzusetzen.

Implementierung der linearen Regression:

Ordinary Least Squares (OLS): Diese Methode minimiert die Summe der quadrierten Fehler, um die optimale Linie durch die Daten zu finden. Sie bildet die Grundlage für die Schätzung der Regressionsparameter und verwendet Optimierungstechniken wie die Gradientensuche.

Praktische Anwendung mit Scikit-Learn: Das Kapitel zeigt die Benutzerfreundlichkeit von Scikit-Learn, einer populären Python-Bibliothek, zur Implementierung von Regressionsmodellen auf. Dies umfasst sowohl Standardmethoden als auch robuste Techniken wie RANSAC, die gegenüber Ausreißern resistent sind.

Bewertung der Modelleistung: Wichtige Metriken wie der mittlere quadratische Fehler (MSE) und R-Quadrat (R^2) werden eingeführt, um die Güte der Modelle zu beurteilen. Es wird die Notwendigkeit betont, separate Trainings- und Testdatensätze zu verwenden, um die



Generalisierbarkeit der Modelle zu sichern.

Regulierte Regressionsmethoden: Um Überanpassung zu vermeiden, werden Regularisierungstechniken wie Ridge und LASSO vorgestellt. Diese Methoden fügen Kostenfunktionen Strafen hinzu, um einfachere Modelle zu fördern, ohne die Vorhersagekraft zu beeinträchtigen.

Nichtlineare Beziehungen und polynomiale Regression: Wenn die Annahme einer linearen Beziehung nicht zutrifft, wird die polynomiale Regression eingeführt, bei der polynomiale Terme in die Regressionsgleichung integriert werden.

Random Forest Regression: Als alternative nicht-parametrische Methode wird die Random Forest Regression vorgestellt, die in der Lage ist, komplexe Beziehungen zu modellieren, ohne dass umfangreiche Merkmalsumwandlungen erforderlich sind.

Zusammenfassung: Das Kapitel legt die Grundlagen und Methoden der Regressionsanalyse dar, beinhaltet verschiedene Techniken zur Modellierung kontinuierlicher Datenbeziehungen und bereitet den Leser auf weiterführende Themen im Bereich des unüberwachten Lernens und der Clusteranalyse in den folgenden Kapiteln vor.



Kapitel 11 Zusammenfassung: Unbeschriftete Daten – Clusteranalyse

In diesem Kapitel erfolgt der Übergang vom überwachten Lernen zu einer spannenden Methode des unbeaufsichtigten Lernens: der Clusteranalyse. Clusteranalyse ist ein Verfahren, das darauf abzielt, verborgene Strukturen in Datensätzen zu entdecken, ohne vorher definierte Labels zu verwenden. Dabei werden natürliche Gruppierungen erkannt, in denen ähnliche Objekte zusammengefasst werden können.

Ein zentrales Konzept ist der **K-Means-Algorithmus**, eine der am häufigsten eingesetzten Methoden zur Gruppierung ähnlicher Objekte. Der Algorithmus arbeitet mit einer festgelegten Anzahl von Clustern (k), die durch die Nähe zu bestimmten Schwerpunkten bestimmt werden. Durch den iterativen Prozess — bei dem anfängliche Schwerpunkte ausgewählt, Proben den nächstgelegenen Schwerpunkten zugeordnet, diese aktualisiert und das Ganze wiederholt wird — wird schließlich eine Konvergenz erreicht. Die Implementierung dieses Algorithmus wird mit der Scikit-learn-Bibliothek erläutert, die umfassende Werkzeuge zum K-means-Clustering bereitstellt. Wichtige Techniken, wie die **Ellenbogenmethode** zur Optimierung der Clusterzahl und **Silhouettenanalysen** zur Bewertung der Clusterqualität, werden detailliert vorgestellt.

Im Abschnitt zur K-means-Gruppierung wird betont, wie einfach und



rechenintensiv K-means implementiert werden kann. Allerdings ist die Auswahl der Clusteranzahl, k , kritisch und kann die Leistung beeinträchtigen, wenn sie ungenau ist. Die Ellenbogenmethode bietet eine visuelle Darstellung der optimalen Clusteranzahl, indem sie die Beziehung zwischen k und der Summe der innerhalb der Cluster liegenden Quadrate untersucht.

Zusätzlich kommt die **K-Means++-Initialisierung zur Sprache**, die strategischere Platzierungen der Anfangsschwerpunkte ermöglicht, um konsistentere und bessere Cluster-Ergebnisse zu gewährleisten.

Ein wichtiger unterschiedlicher Ansatz ist der zwischen **hartem und weichem Clustering**. Während hartes Clustering jedem Datensatz eindeutig ein Cluster zuordnet, erlaubt weiches Clustering (z.B. fuzzy C-means) probabilistische Zuordnungen, die differenziertere Darstellungen der Mitgliedschaften ermöglichen.

Das Kapitel führt auch die beiden grundlegenden Methoden des hierarchischen Clustering ein, die Daten in geschachtelten Strukturen basierend auf Abständen organisieren. Dendrogramme veranschaulichen diese Strukturen und ermöglichen eine intuitive Einsicht in die Beziehungen zwischen den Clustern.

Ein weiteres leistungsstarkes Verfahren ist der **DBSCAN-Algorithmus**, der



Cluster auf der Grundlage der lokalen Punktdichte identifiziert. Dieser Algorithmus hat den Vorteil, dass er willkürlich geformte Cluster erkennen kann und weniger anfällig für Rauschen ist, ohne dass die Anzahl der Cluster im Voraus festgelegt werden muss.

Zusammenfassend behandelt das Kapitel drei gängige Clustering-Algorithmen: K-Means, agglomeratives hierarchisches Clustering und DBSCAN. Diese Verfahren zur Clusteranalyse ermöglichen es, ohne vordefinierte Klassifikationen verborgene Muster in Daten zu entdecken. Bewertungsmethoden wie die Ellenbogen- und Silhouettenmethoden helfen dabei, die Clusterstrukturen quantitativ zu analysieren. Im kommenden Kapitel richten wir unseren Fokus auf das überwachte Lernen mit mehrschichtigen künstlichen neuronalen Netzen, wo wir die nächsten Schritte zur tiefen Datenanalyse erkunden werden.



Kapitel 12: Mehrschichtiges ANN von Grund auf

Zusammenfassung von Kapitel 12: Implementierung eines mehrschichtigen künstlichen neuronalen Netzwerks von Grund auf

In Kapitel 12 wird eine ausführliche Einführung in mehrschichtige künstliche neuronale Netzwerke (ANNs) gegeben, auch bekannt als Deep Learning. Dieses Kapitel ist entscheidend, da es grundlegende Konzepte und Techniken behandelt, die den Leser auf die Verwendung fortschrittlicher Python-Bibliotheken in späteren Kapiteln vorbereiten.

Zunächst wird die historische Entwicklung neuronaler Netzwerke beleuchtet, die in den 1940er Jahren begann. Das Kapitel beschreibt wichtige Fortschritte, wie das Perzeptron-Modell und die Wiederbelebung des Interesses in den 1980er Jahren durch den Backpropagation-Algorithmus. Diese Technologien haben einen wesentlichen Einfluss auf moderne Anwendungen von Deep Learning, insbesondere in den Bereichen Bild- und Spracherkennung.

Das Kapitel führt dann in die Struktur mehrschichtiger neuronaler Netzwerke ein, auch Multilayer Perceptrons (MLPs) genannt. MLPs bestehen aus Eingangsschichten, verborgenen Schichten und Ausgangsschichten, die durch gewichtete Verbindungen miteinander



verbunden sind. Diese Architektur ermöglicht es, komplexe Funktionen zu modellieren, was für Klassifikationsaufgaben unerlässlich ist.

Ein zentraler Aspekt der Netzwerkentwicklung ist das Training, welches umfassend mit dem Backpropagation-Algorithmus erklärt wird. Dieser Algorithmus berechnet die erforderlichen Gradienten für die Aktualisierung der Gewichte während des Trainingsprozesses. Zunächst wird die Vorwärtsbewegung beschrieben, durch die Ausgaben und Fehler ermittelt werden, gefolgt von der Rückwärtsbewegung, wo die notwendigen Gewichtsaktualisierungen vorgenommen werden.

Bevor die Komplexität mehrschichtiger Netzwerke in den Vordergrund rückt, wird auch das Konzept von einlagigen Netzwerken angesprochen, wobei der ADaptive LInear NEuron (Adaline)-Algorithmus vorgestellt wird. Der Vorwärtspropagationsprozess wird erläutert als Methode, um Vorhersagen zu generieren, indem Eingaben durch die Netzwerkschichten unter Verwendung spezifischer Aktivierungsfunktionen geleitet werden.

Das Kapitel bietet eine detaillierte Schritt-für-Schritt-Anleitung zur Implementierung eines mehrschichtigen Perzeptrons. Dies umfasst praktische Beispiele zur Kodierung eines MLP, das auf Datensätze wie MNIST zur Erkennung handgeschriebener Ziffern zugreift. Darin sind auch Hilfsfunktionen enthalten, die den Trainings- und Validierungsprozess unterstützen sowie Visualisierungen zur Überwachung der



Leistungskennzahlen, wie Kosten und Genauigkeit, beinhalten.

Ein mathematischer Überblick über den Backpropagation-Algorithmus zeigt auf, wie die Berechnung der Gradienten erfolgt, um eine effiziente Gewichtsanzpassung zu ermöglichen. Besonders die Bedeutung der

**Installieren Sie die Bookey App, um den
Volltext und Audio freizuschalten**

Kostenlose Testversion mit Bookey





Lesen, Teilen, Befähigen

Beenden Sie Ihre Leseherausforderung, spenden Sie Bücher an afrikanische Kinder.

Das Konzept



Diese Buchspendenaktion wird gemeinsam mit Books For Africa durchgeführt. Wir starten dieses Projekt, weil wir dieselbe Überzeugung wie BFA teilen: Für viele Kinder in Afrika ist das Geschenk von Büchern wirklich ein Geschenk der Hoffnung.

Die Regel



Sammeln Sie 100 Punkte Lösen Sie ein Buch ein Spenden Sie für Afrika

Ihr Lernen bringt nicht nur Wissen, sondern ermöglicht es Ihnen auch, Punkte für wohltätige Zwecke zu sammeln! Für je 100 gesammelte Punkte wird ein Buch nach Afrika gespendet.

Kostenlose Testversion mit Bookey



Kapitel 13 Zusammenfassung: Parallelisierung des Trainings von neuronalen Netzen mit TensorFlow

In diesem Kapitel wird TensorFlow, eine leistungsstarke Deep-Learning-Bibliothek von Google, vorgestellt, die die Implementierung und das Training neuronaler Netze erheblich verbessert, insbesondere im Vergleich zu herkömmlichen Ansätzen mit NumPy. Die Einführung von TensorFlow markiert einen bedeutenden Fortschritt in der Effizienz des Machine Learning, da es moderne Hardware, insbesondere GPUs, nutzt, um Berechnungen schneller durchzuführen. CPUs bieten zwar mehrere Kerne, aber der globale Interpreter-Lock (GIL) in Python limitiert die Ausführung auf einen Kern, was die Notwendigkeit von GPU-optimierten Implementierungen für große Modelle und Datensätze unterstreicht.

TensorFlow selbst ist ein Open-Source-Framework, das sowohl auf CPUs als auch GPUs läuft und mehrere Programmiersprachen unterstützt, wobei Python die umfangreichste API hat. Die Kernidee von TensorFlow basiert auf gerichteten Graphen, die den Datenfluss visualisieren, und hochrangigen APIs, die die Entwicklung komplexer Modelle vereinfachen.

Zu Beginn des Kapitels lernen wir, wie man mit der Low-Level-TensorFlow-API arbeitet, um einen Berechnungsgraphen zu erstellen, Platzhalter für Eingaben zu definieren und TensorFlow-Sessions für die Ausführung von Berechnungen zu nutzen. Hier werden grundlegende



Strukturen wie Tensoren und Array-Manipulationen behandelt, genauso wie die Definition einfacher Modelle.

Ein praktisches Beispiel führt uns über die Implementierung der Methode der kleinsten Quadrate (OLS-Regressionsanalyse). Der Fokus liegt auf dem Modelldesign, der Initialisierung von Variablen und dem Training mit der Methode des Gradientenabstiegs. Durch die Visualisierung der Trainingskosten erhalten wir Einblicke in die Leistungsfähigkeit des Modells.

Mit der Einführung der hochrangigen APIs, insbesondere der Layers API und Keras, wird das Training nachfolgend vereinfacht. Wir zeigen, wie ein mehrschichtiges Perzeptron zur Klassifizierung des MNIST-Datensatzes aufgebaut wird. Diese Tools versprechen nicht nur Effizienz, sondern auch Benutzerfreundlichkeit während des Trainingsprozesses.

Ein wesentlicher Punkt in der Entwicklung neuronaler Netze sind die Aktivierungsfunktionen, die für die Leistungsfähigkeit der Modelle entscheidend sind. Wir besprechen verschiedene Funktionen, darunter die logistische Funktion (sigmoid), die hyperbolische Tangensfunktion (tanh) und die Rectified Linear Unit (ReLU). Jede dieser Funktionen hat spezifische Vorteile und hilft, Probleme wie das Verschwinden des Gradienten zu bewältigen, was für das Training tiefer neuronaler Netze von zentraler Bedeutung ist.



Zusammenfassend bietet dieses Kapitel einen umfassenden Überblick über TensorFlow und seine Rolle bei der Steigerung der Trainingseffizienz von neuronalen Netzen. Wir untersuchen grundlegende Konzepte, die Modellentwicklung und die Bedeutung von Aktivierungsfunktionen. Im nächsten Kapitel werden wir tiefer in die Funktionsweise von TensorFlow eintauchen und uns mit Graphensitzungen, Modellmanagement und fortgeschritteneren Konzepten befassen.



Kapitel 14 Zusammenfassung: Funktionsweise von TensorFlow

Zusammenfassung von Kapitel 14 - Machine Learning mit Python, ScikitLearn und TensorFlow

In diesem Kapitel wird die Nutzung von TensorFlow eingehend erläutert, wobei auf den praktischen Erfahrungen aus Kapitel 13 aufgebaut wird. Im Fokus stehen die grundlegenden Eigenschaften von TensorFlow, die Struktur von Tensoren, Berechnungsgrafiken, Variablen, Operationen sowie die Visualisierung von Modellen mit TensorBoard.

Einführung in TensorFlow

TensorFlow ist eine weit verbreitete und skalierbare API für Machine-Learning-Aufgaben, die seit ihrem ersten Release 2017 aktiv weiterentwickelt wird. Mit einer starken Unterstützung durch Google und die Open-Source-Community ist TensorFlow bestens dokumentiert und bietet Funktionen, die den Einsatz auf einer oder mehreren GPUs ermöglichen, sowie Optionen für mobile Anwendungen.

Tensoren und deren Ränge

Im Herzen von TensorFlow befinden sich Tensoren, die mehrdimensionale Arrays sind. Ihre **Ränge** geben die Anzahl der Dimensionen an:



- **Rang 0:** Ein Skalar, also eine einzelne Zahl.
- **Rang 1:** Ein Vektor, analog zu einem Array.
- **Rang 2:** Eine Matrix, vergleichbar mit einem 2D-Array.

Tensoren können mit Funktionen wie ``tf.rank`` und ``get_shape()`` manipuliert werden, um Informationen über ihre Dimensionen zu extrahieren.

Berechnungsgrafiken in TensorFlow

Ein zentrales Konzept von TensorFlow ist die Verwendung von Berechnungsgrafiken zur Darstellung von Operationen und Tensoren. Der Aufbau eines solchen Grafen erfolgt in mehreren Schritten:

1. Anlegen eines leeren Grafen.
2. Hinzufügen von Tensoren und den entsprechenden Operationen in Form von Knoten.
3. Ausführung des Grafen innerhalb einer TensorFlow-Sitzung, die die Berechnungen organisiert.

Platzhalter und Variablen

Platzhalter dienen dazu, externe Daten in diesen Grafen zu integrieren, während Variablen Modellparameter speichern und diese während des Trainings aktualisieren. Es ist entscheidend, dass Variablen innerhalb einer



Sitzung initialisiert werden, um korrekt zu funktionieren.

Durchführung von Operationen

In TensorFlow können Transformations- und Operationsanfragen an Tensoren ähnlich wie in NumPy durchgeführt werden. Wichtige Operationen beinhalten das Umformen von Tensoren (`tf.reshape`), das Transponieren (`tf.transpose`) und das Aufteilen (`tf.split`).

Kontrollflussmechanismen

TensorFlow unterstützt Kontrollflussoperationen wie `tf.cond`, die es ermöglichen, basierend auf Eingabebedingungen Entscheidungen innerhalb der Berechnungsgrafiken zu treffen. Dies erleichtert den dynamischen Aufbau von Grafiken.

Visualisierung mit TensorBoard

TensorBoard ist ein hilfreiches Modul in TensorFlow zur Visualisierung von Berechnungsgrafiken und zur Überwachung des Fortschritts im Training von Modellen. Es beinhaltet Schritte wie das Exportieren des Grafens mit `tf.summary.FileWriter` und das anschließende Starten von TensorBoard zur interaktiven Analyse.

Fazit und Ausblick

Dieses Kapitel vermittelt die grundlegenden Aspekte zur Erstellung und dem Training von Machine-Learning-Modellen mit TensorFlow. Im kommenden



Kapitel wird der Fokus auf die Implementierung von Convolutional Neural Networks (CNNs) gelegt, die sich für fortgeschrittene Aufgaben der Bildklassifizierung eignen.

Kostenlose Testversion mit Bookey



Zum Herunterladen sca

Kapitel 15 Zusammenfassung: Klassifizierung von Bildern mit tiefen konvolutionalen NNs

In diesem Kapitel widmen wir uns der Klassifizierung von Bildern mithilfe von tiefen konvolutionalen neuronalen Netzwerken (CNNs) und deren Implementierung in TensorFlow. Ziel ist es, die Funktionsweise von CNNs als fortschrittliche Merkmalsextraktionssysteme zu verstehen.

Zunächst betrachten wir die **Grundbausteine der CNNs**. Diese Netzwerke imitieren die menschliche visuelle Wahrnehmung, ein Konzept, das Yann LeCun in den 1990er Jahren entwickelte. Die Architektur von CNNs besteht aus konvolutionalen und Pooling-Schichten, die es ermöglichen, Bilder effizient zu verarbeiten. Sie nutzen spärliche Verbindungen und das Teilen von Parametern, um die Anzahl der erlernbaren Parameter zu reduzieren und die Merkmalsextraktion zu verbessern.

Ein zentrales Merkmal von CNNs ist die Fähigkeit, eine **Merkmalshierarchie** zu bilden, die von niedrigstufigen Attributen wie Kanten zu komplexeren Darstellungen wie ganzen Objekten reicht. Diese Hierarchie wird durch die verschiedenen Schichten des Netzwerks gebildet, was CNNs besonders effektiv bei bildbezogenen Aufgaben macht.

Die **konvolutionalen Operationen** sind ebenfalls entscheidend. Wir analysieren die diskrete Konvolution in ein- und zweidimensionalen



Kontexten und erläutern grundlegende mathematische Konzepte, darunter Konvolution und Kreuzkorrelation. Darüber hinaus beschäftigen wir uns mit Techniken wie Null-Padding und der Steuerung der Schrittweite, um die Ausgabedimensionen zu optimieren.

Die **Pooling-Schichten**, darunter Max-Pooling und Average-Pooling, dienen der Dimensionsreduktion und sind dafür verantwortlich, wichtige Merkmale zu erhalten. Sie fördern die lokale Invarianz, wodurch die Robustheit und Recheneffizienz des Netzwerks steigen.

Beim **Aufbau einer CNN-Architektur** beginnt der Prozess mit den Eingabebildern, gefolgt von konvolutionalen Schichten, Pooling-Schichten und endet in vollständig verbundenen Schichten, die auf einem Softmax-Ausgang basieren. Diese Struktur ist besonders geeignet für Aufgaben wie die Erkennung handgeschriebener Ziffern.

Um Überanpassung zu vermeiden, führen wir die **Regularisierung durch Dropout** ein. Diese Methode entfernt während des Trainings zufällig Einheiten und unterstützt eine allgemeinere Lernfähigkeit, was die Leistung bei unbekannten Daten verbessert.

In der **Implementierung des CNNs in TensorFlow** nutzen wir sowohl die Low-Level API als auch die TensorFlow Layers API. Zentrale Aspekte umfassen die Definition der konvolutionalen und vollständig verbundenen



Schichten, die Verwaltung von Platzhaltern für Eingabedaten und die Entwicklung von Trainingsmethoden sowie das Speichern und Laden des Modells.

Die **Modellergebnisse** zeigen, dass das Training eines CNNs mit dem MNIST-Datensatz zu einer hohen Genauigkeit führt. Dies verdeutlicht die Effektivität von CNNs im Bereich der Bildklassifizierung, da sie signifikante Leistungsverbesserungen im Vergleich zu traditionellen vollständig verbundenen Netzwerken bieten.

Zusammenfassend behandelt dieses Kapitel die Grundlagen und Eigenschaften von CNNs, die damit verbundenen konvolutionalen und Subsampling-Techniken sowie die vollständige Architektur zur Bildklassifizierung mit TensorFlow. In unserem nächsten Kapitel werden wir uns mit rekursiven neuronalen Netzwerken (RNNs) befassen, die speziell auf Sequenzdaten ausgerichtet sind und Anwendungen in Bereichen wie maschineller Übersetzung und Bildbeschreibung finden.



Kapitel 16: Modellierung sequenzieller Daten mit rekursiven NNs

Modellierung sequenzieller Daten mit rekursiven neuronalen Netzen

Überblick über die Themen

In diesem Kapitel werden rekursive neuronale Netze (RNNs) als zentrale Methode zur Modellierung von sequenziellen Daten, insbesondere Zeitreihendaten, vorgestellt. Die behandelten Themen umfassen die Einführung in sequenzielle Daten, die Struktur und Funktionsweise von RNNs, Long Short-Term Memory (LSTM) Einheiten, Techniken wie Truncated Backpropagation Through Time (T-BPTT), die Implementierung mehrschichtiger RNNs in TensorFlow sowie praxisnahe Anwendungsprojekte wie Sentimentanalyse und sprachliche Modellierung auf Zeichenebene. Außerdem wird auf Techniken zur Vermeidung explodierender Gradienten eingegangen, insbesondere Gradientenklippung.

Einführung in sequenzielle Daten

Sequenzielle Daten, wie Zeitreihen oder Textfolgen, zeichnen sich durch ihre geordnete Struktur aus, in der die Elemente voneinander abhängen. Im Gegensatz zu traditionellen Machine-Learning-Modellen, die oft von der Annahme der Unabhängigkeit der Eingaben ausgehen, sind RNNs speziell dafür entwickelt, die zeitlichen Beziehungen in diesen Daten zu erfassen.



Modellierung sequenzieller Daten

Es gibt drei wesentliche Modelle der Sequenzverarbeitung:

1. **Viele-zu-eins:** Eine Eingabesequenz wird in einen festen Ausgabewert umgewandelt, wie bei der Sentimentanalyse.
2. **Eins-zu-viele:** Ein einzelnes Eingangsformat führt zu einer Sequenzausgabe, beispielsweise bei der Bildbeschriftung.
3. **Viele-zu-viele:** Sowohl die Eingabe- als auch die Ausgabeformate sind Sequenzen, was bei der Video-Klassifizierung der Fall ist.

RNN Struktur und Datenfluss

Die Struktur von RNNs beinhaltet Rückkopplungsschleifen, die es ermöglichen, Informationen aus vorherigen Zeitschritten zu nutzen. Der Informationsfluss in einem RNN erfolgt durch eine Eingabeschicht, die die Daten des aktuellen Zeitschritts verarbeitet, und durch rekursive Verbindungen, die den verborgenen Zustand der vorherigen Schicht repräsentieren.

Berechnung von Aktivierungen in RNNs

Die Aktivierung der verborgenen Einheiten wird durch Gewichtsmatrizen bestimmt, die über die Zeit hinweg konstant bleiben. In einem einzelnen Verarbeitungsschritt wird der Nettoeingang berechnet und eine Aktivierungsfunktion angewendet.



Training von RNNs mit BPTT

Das Training von RNNs erfolgt durch Backpropagation Through Time (BPTT), bei dem Gradientenschätzungen über die Zeitspanne hinweg berechnet werden. Dabei treten Herausforderungen in Form von verschwindenden oder explodierenden Gradienten auf, die mit Techniken wie LSTM-Einheiten und T-BPTT angegangen werden.

Long Short-Term Memory (LSTM) Einheiten

LSTM-Netzwerke wurden entwickelt, um die Herausforderungen beim Training von RNNs zu überwinden. Sie verwalten Informationen durch einen Zellzustand und verfügen über drei spezifische Türmechanismen:

- **Vergessenstor:** Bestimmt, welche Informationen verworfen werden sollen.
- **Eingangsportal:** Aktualisiert den Zellzustand mit neuen Informationen.
- **Ausgangstor:** Steuert die Ausgabe auf Basis des verborgenen Zustands.

Implementierung von RNNs in TensorFlow

Das Kapitel liefert eine praktische Anleitung zur Implementierung von RNNs in TensorFlow, illustriert durch zwei konkrete Projekte:

1. **Sentimentanalyse:** Nutzung einer Viele-zu-eins RNN-Architektur zur Analyse von Filmkritiken auf IMDb.
2. **Zeichenniveau-Sprachmodellierung:** Generierung von Text durch



Zeichen, wobei Techniken zur Handhabung von Sequenzen, wie Gradientenklippung, essentiell werden.

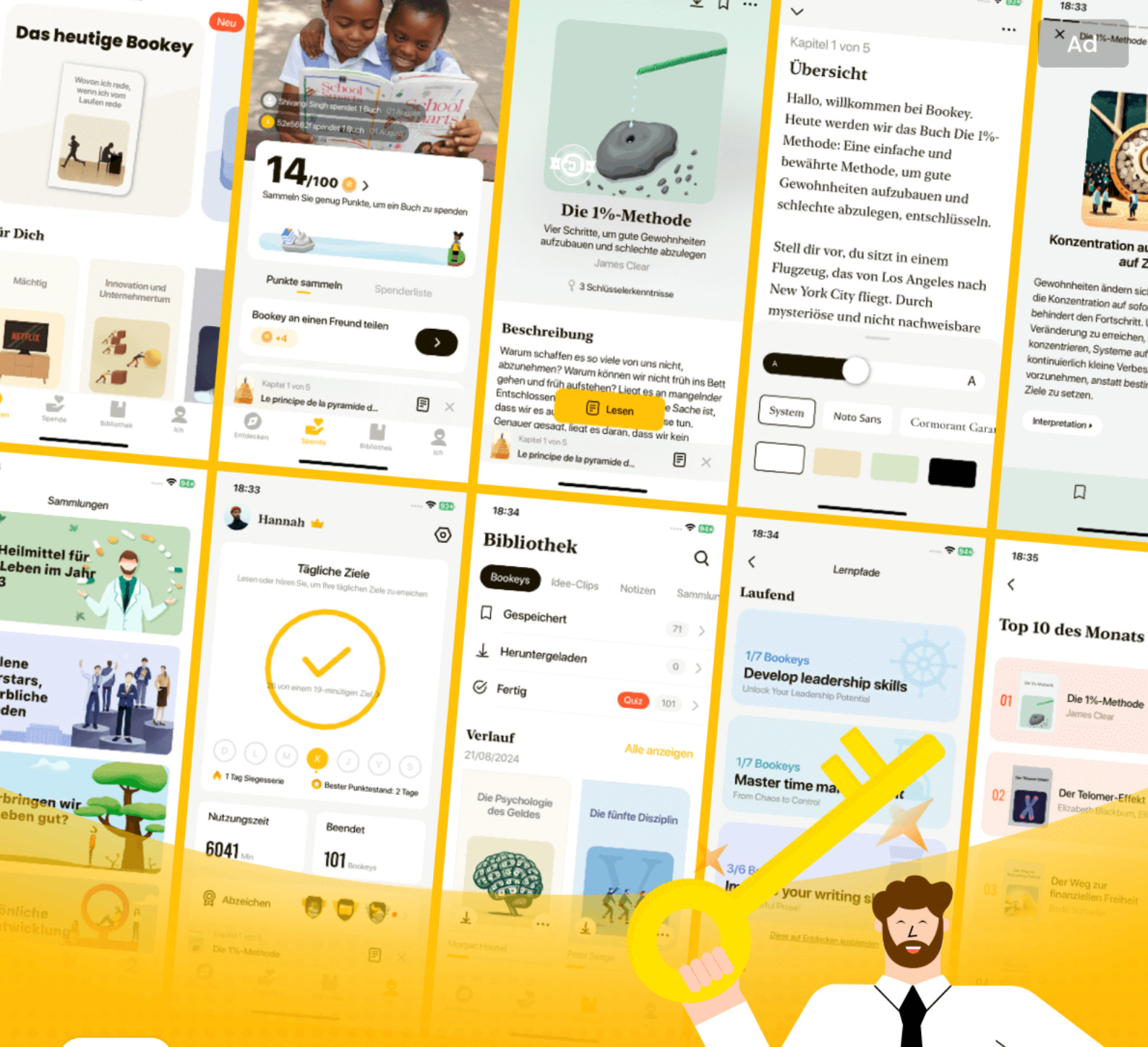
Training und Validierung

Für das Training von RNNs werden Techniken zur Datenvorverarbeitung

**Installieren Sie die Bookey App, um den
Volltext und Audio freizuschalten**

Kostenlose Testversion mit Bookey





Die besten Ideen der Welt entfesseln Ihr Potenzial

Kostenlose Testversion mit Bookey

