iX DEVELOPER
Winter 2020/21
14,90 €
Österreich 16,40 €
Schweiz 27.90 CHF
Luxemburg 17,10 €
www.ix.de

Machine Learning

Bessere Modelle, produktiver Einsatz

Grundlagen

Unsupervised und Reinforcement Learning
Modelle verstehen und optimieren
Long Short-Term Memory

Tools

TensorFlow und PyTorch
Wichtige Libraries für Python

Infrastruktur

ML von der Stange aus der Cloud
Vom Modell zum Produktivbetrieb
Programmiersprache Julia

Praxis

Texte mit NLP gekonnt analysieren
Qualifizierte Vorhersagen für Zeitreihen
Tutorial Bildanalyse

Security und Recht

Angriffsvektoren auf neuronale Netze
Urheberrecht und Datenschutz

B1 Consulting Managed Service & Support

individuell – umfassend – kundenorientiert

Neue oder bestehende Systemlandschaften stellen hohe Anforderungen an Ihr IT-Personal. Mit einem individuellen Support- und Betriebsvertrag von B1 Systems ergänzen Sie Ihr Team um die Erfahrung und das Wissen unserer über 130 festangestellten Linux- und Open-Source-Experten.

Unsere Kernthemen:

Linux Server & Desktop · Private Cloud (OpenStack & Ceph) · Public Cloud (AWS, Azure, OTC & GCP) · Container (Docker, Kubernetes, Red Hat OpenShift, Rancher & SUSE CaaSP) · Monitoring (Icinga, Nagios & ELK) · Patch Management · Automatisierung (Ansible, Salt, Puppet & Chef) · Videokonferenzen

Unser in Deutschland ansässiges Support- und Betriebsteam ist immer für Sie da – mit qualifizierten Reaktionszeiten ab 10 Minuten und Supportzeiten von 8x5 bis 24x7!

Zwei Tage Linux-/Open-Source-Consulting zum Preis von einem!
Mail an **info@b1-systems.de** und Aktionscode **IX2020** angeben*!

*Aktionscode einmal pro Unternehmen einlösbar

B1 SYSTEMS

B1 Systems GmbH - Ihr Linux-Partner

Linux/Open Source Consulting, Training, Managed Service & Support

ROCKOLDING · KÖLN · BERLIN · DRESDEN

www.b1-systems.de · info@b1-systems.de

Dampf-Machine-Learning

Als wir 2018 unser erstes Sonderheft zum Thema Machine Learning erstellt haben, wirkte das KI-Universum wie eine große Spielwiese. Forscherinnen und Forscher experimentierten mit diversen Ansätzen und meldeten laufend neue Erfolge. Und zwar tatsächlich häufig im spielerischen Umfeld: Computer bewiesen ihr Können im Schach, in Go, bei Jeopardy oder in Starcraft II.

Zwei Jahre später nutzen mehr und mehr Unternehmen Machine Learning im produktiven Einsatz. Das Motto Steampunk mit seiner Kombination aus futuristischen und klassischen Elementen und dem kreativen Ambiente passt meines Erachtens gut zu denjenigen, die sich von der weiterhin fehlenden Standardisierung und der unvollständigen Werkzeugkiste nicht abhalten lassen, ihre eigenen "Maschinen" zu bauen.

Und tatsächlich hat das ML-Ökosystem in den letzten zwei Jahren deutliche Fortschritte beim Übergang vom Experimentieren zum Einsatz aufzuweisen. Die großen Frameworks TensorFlow und PyTorch nähern sich einander an und bieten einen leichteren Einstieg als 2018. Für die Kompatibilität zwischen den Tools und Plattformen hat sich das Austauschformat ONNX inzwischen fest etabliert, und für Versionierung und Integration existieren Open-Source-Tools.

Das Bild der Dampfmaschine weckt bei mir Erinnerungen an die Figur des Bömmel in der Feuerzangenbowle, der in der Filmfassung – leicht abgewandelt von Heinrich Spoerls Literaturvorlage – den Schülern die rheinisch gefärbte Erklärung gibt: „Wat is'n Dampfmaschin? Da stelle mer uns mal janz dumm und sagen: Ne Dampfmaschin dat issene jroße, runde, schwarze Raum. Und der jroße, runde, schwarze Raum, der hat zwei Löcher. Dat eine Loch, da kömmt der Dampf erein, und dat andere Loch, dat krieje mer später“.

Das Bild lässt sich auf die Sicht übertragen, die viele von neuronalen Netzen haben: Auf der einen Seite kommen die Daten rein, und die andere Seite verstehen wir später. Das genügt, solange die Maschine wie geschmiert läuft. Fehlentscheidungen können jedoch oft massive Auswirkungen haben, und anders als in manuell geschriebenem Code lässt sich nicht einfach nachvollziehen, woher das Modell seine Schlüsse zieht. Das kann Systeme nicht nur behindern, sondern kritische Folgen haben. Zudem führen immer wieder Biases in den Eingabedaten zu vorurteilsbehafteten Entscheidungen.

Für das Verständnis und Anpassen ist es wichtig, einen Blick in die Black Boxes der Modelle zu werfen, die den Bogen zurück zum schwarzen Raum der Dampfmaschine spannen. Jenseits davon existieren rechtliche Grundlagen wie DSGVO und Urheberrecht, die Unternehmen (glücklicherweise) davon abhalten, ML-Systeme mit allen verfügbaren Daten zu füttern.

Das vorliegende Machine-Learning-Heft setzt im Vergleich zum vorigen, das Sie kostenlos herunterladen können, einen deutlich stärkeren Fokus auf praktische Anwendungen. Die meisten Autoren haben außerdem zu ihren Artikeln Jupyter Notebooks erstellt, die Sie als Vorlage für eigene Projekte nutzen können. Damit bekommt die Dampfmaschine Räder, und wir kehren zurück zu Bömmel mit dem Satz, der im Buch seine Erklärung der Dampfmaschine erweitert: „Und wenn die jroße schwarze Raum Räder hat, dann es et en Lokomotiv“.

In diesem Sinne wünschen *heise developer* und *iX* Ihnen eine gute Fahrt für Ihre Projekte mit Dampfantrieb durch das Sonderheft.

RAINALD MENGE-SONNENTAG

iX Developer „Machine Learning“ (Winter 2018) zum Download: ix.de/zx1z

Grundlagen

Unsupervised Learning hilft besonders beim Verständnis unbekannter Daten, und mit Long-Short-Term-Memory-Netzen lässt sich unter anderem Kundenverhalten vorhersagen. Reinforcement Learning gewinnt im produktiven Einsatz zunehmend an Bedeutung. Für alle ML-Anwendungen ist das Verständnis der Black Box wichtig, um Entscheidungen nachzuvollziehen.

ab Seite 6

Tools

Python Libraries wie NumPy und Pandas helfen bei der Aufbereitung von Daten. Bei den ML-Frameworks ist TensorFlow weiterhin der Platzhirsch, aber PyTorch etabliert sich ebenfalls in einigen Bereichen. Für die Versionierung von ML-Daten existieren Open-Source-Tools wie DVC. Und auch im mobilen und IoT-Umfeld lässt sich ML sinnvoll nutzen.

ab Seite 34

Grundlagen

Unsupervised Learning: Methoden und Einsatz **6**
Experimente für Reinforcement Learning **16**
Long Short-Term Memory für Geschäftsanwendungen **24**
Erklärbarkeit und Fairness **30**

Tools

Ein Streifzug durch die PyData-IT-Landschaft **34**
TensorFlow 2.0 und Keras: Imperative Modellentwicklung **40**
PyTorch als Alternative zu TensorFlow **44**
Data Version Control im Team mit Open-Source-Werkzeugen **48**
Machine Learning für IoT und Mobile **54**

Infrastruktur

Vom Modell zum produktiven Einsatz **60**
Machine Learning as a Service **64**
Hardwarebeschleuniger für neuronale Netze **70**
Julia – eine differenzierbare Programmiersprachefür ML **76**

Praxis

Word Embeddings: Theorie und Beispielanwendung **84**
Textanalyse mit BERT **94**
Methoden für Zeitreihenvorhersagen **100**
Deep-Learning-Tutorial Bildanalyse:
- Teil 1: Bilder für das Modelltraining aufbereiten **108**
- Teil 2: Trainieren des Modells **114**
- Teil 3: Einsatz auf der Zielhardware **120**

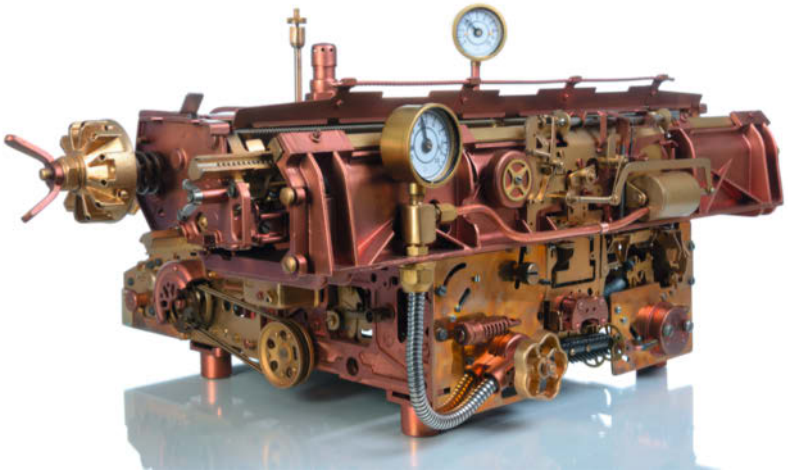
Machine Learning zur Schadcode-Erkennung **126**
Recommender-Systeme in der Praxis **129**

Security und Recht

Gefahren durch Angriffe auf KI **137**
Neuronale Netze: Angriffe und Verteidigung **140**
Datenschutz und Urheberrecht bei der KI-Entwicklung **143**

Sonstiges

Editorial **3**
Impressum **89**



Infrastruktur

Auf dem steinigen Weg vom Modell in den Produktivbetrieb helfen Open-Source-Tools wie ONNX und FastAPI. Die großen Cloud-Provider bieten derweil vorkonfigurierte ML-Dienste an. Für den Betrieb im eigenen Unternehmen existieren unterschiedliche Hardwarebeschleuniger. Die Programmiersprache Julia bietet einige Konzepte, mit der sie Python als ML-Sprache herausfordern möchte.

ab Seite 60

Praxis

Die Textanalyse ist ein spannendes Feld, das in den letzten Jahren dank Verfahren wie BERT deutlich an Fahrt aufgenommen hat. Word Embeddings sind ein Standardverfahren, um Zusammenhänge in Texten zu erkennen. Ein dreiteiliges Tutorial führt in die Bildanalyse ein, und das Heft beleuchtet Zeitreihenanalyse, Recommender-Systeme und ML im Kampf gegen Hacking.

ab Seite 84

Security und Recht

ML-Systeme sind zunehmend Ziel unterschiedlicher Angriffe, und bei neuronalen Netzen findet ein regelrechter Wettlauf zwischen Angriffs- und Verteidigungsmethoden statt. Derweil gilt es, beim Verwenden von Daten in ML-Anwendungen Datenschutz und Urheberrecht zu beachten.

ab Seite 137

Unsupervised Learning: Methoden und Einsatz

Ohne Beobachter

Harald Bosch, Arthur Varkentin

Machine Learning ist mehr als die Automatisierung menschlicher Arbeit. Gerade beim Verständnis unbekannter Daten helfen Methoden des unüberwachten Lernens, die kaum menschliche Eingaben benötigen.

Die Aufteilung der KI-Welt in überwachtes und unüberwachtes Lernen kann aus zwei Sichtweisen erfolgen. Die engere, algorithmenzentrierte Sicht umfasst Lernverfahren, bei denen es schlicht keine „einzig richtige" Lösung, das sogenannte Label, gibt. Im harten Kontrast zum überwachten Lernverfahren werden im unüberwachten Fall also keine Beispiele zum Lernen verwendet, die sowohl die Eingabe als auch die gewünschte Ausgabe umfassen. Stattdessen sind lediglich die Eingabedaten bekannt, und das Ziel besteht darin, eine Struktur zu erzeugen, die diese Daten optimal widerspiegelt.

Die weitere, problemorientierte Sicht umfasst unüberwachte Aufgabenstellungen für das Machine Learning, bei denen Labels eventuell existieren, aber nicht mühsam von Menschen erstellt werden. Beispielsweise ist die Vorhersage des nächsten Buchstabens in einem Text eigentlich ein Klassifikationsproblem (welcher Buchstabe ist der richtige?) und damit, im engeren Sinne, ein überwachtes Verfahren. Jedoch lassen sich die Trainingsdaten einfach und ohne menschliches Zutun erstellen, wenn Texte beliebig zerteilt werden und jeweils der erste Teil als Eingabe und der Anfang des zweiten Teils als gewünschte Ausgabe definiert ist. So können Lernverfahren, die eigentlich überwacht sind, anhand großer ungelabelter Datensammlungen Strukturen erzeugen.

Sinnvoller Einsatz

Nicht fundamental anders lernt das derzeit in den Medien häufig präsente OpenAI GPT-3 [1], das Internet auswendig aufzusagen. Damit es diese anspruchsvolle Aufgabe erfüllen kann, muss es sehr viel Wissen über menschliche Sprache und den Kontext eines Textes aufbauen. Dieses Wissen lässt sich in vielen weiteren Aufgaben gewinnbringend als Basis nutzen, um Antworten für Chatbots zu erstellen, Texte zusammenzufassen oder Bedienoberflächen anhand einer textuellen Beschreibung zu erstellen. Der GPT-Ansatz konnte damit Nutzen aus sehr viel mehr Daten ziehen als Verfahren, die sich nur auf die gelabelten Trainingsdaten der eigentlichen Problemstellung konzentriert haben. Die erste Version von GPT konnte damit auf Anhieb in neun von zwölf untersuchten Aufgabenstellungen die damals besten spezialisierten Umsetzungen schlagen [2].

Das ist nur ein Beispiel für das Potenzial von unüberwachtem Lernen. Allgemeiner gesprochen lassen sich die Strukturen, die aus den Verfahren entstehen, auf drei nicht ganz sauber trennbare Weisen einsetzen: Clustering, Kompression und Anomalie-Erkennung (s. Kasten Übliche Anwendungsfälle für unüberwachtes Lernen).

Clustering

Ein Hauptverwendungszweck für Unsupervised-Learning-Algorithmen ist das Clustering, bei dem zusammengehörige Datenpunkte in Clustern gruppiert werden können. Es dient primär dazu, Ähnlichkeitsstrukturen innerhalb der Daten zu identifizieren und zu segmentieren. Dabei ist häufig von Features die Rede: messbare Merkmale der Daten wie Verkaufswert eines Hauses, seine Lage oder die Verkehrsanbindung. Jedes Merkmal definiert eine eigene Dimension beziehungsweise repräsentiert eine Achse. Oftmals lassen sich Features als Spalten einer Tabelle denken, sie können jedoch durchaus abstrakt sein. In der Regel weisen die Merkmale Abhängigkeiten unter-

iX-TRACT

- Unsupervised Learning verzichtet auf die Eingabe von Daten beziehungsweise Beispielen im Vorfeld.
- Zahlreiche Aufgabenstellungen wie Anomalieerkennung bieten sich für den Einsatz an.
- Verschiedene Verfahren wie k-Means, HAC und DBSCAN taugen für unterschiedliche Einsatzgebiete.

einander auf. So richtet sich beispielsweise der Verkaufswert eines Hauses nicht unerheblich nach der Verkehrsanbindung und der sonstigen Infrastruktur. Beim Clustering lassen sich genau solche Zusammenhänge erkennen und die Datenpunkte entsprechend gruppieren.

Zahlreiche Algorithmen lassen sich für Clustering nutzen, und sie können unterschiedlichen Paradigmen folgen. Beispielsweise können die gefundenen Cluster flach oder hierarchisch gruppiert sein. Ersterer Ansatz partitioniert den Feature-Raum, also das Koordinatensystem, in dem sich die Datenpunkte aufhalten. Bei einer hierarchischen Gruppierung können Cluster selbst Teil eines größeren Clusters sein. Daneben gibt es dichtebasierte Verfahren, die über eine Dichtemetrik die Zugehörigkeit zu einem Cluster definieren.

Bei den folgenden Beispielen stehen drei der am häufigsten verwendeten Algorithmen im Fokus, die jeweils einen Vertreter der Verfahrenstypen beschreiben. Sie beschränken sich zu Gunsten der Anschaulichkeit auf zweidimensionale Verteilungen, aber die Verfahren lassen sich ebenso auf vieldimensionale Daten anwenden, wie der weiter unten stehende Abschnitt „Reduzierte Dimensionen“ aufzeigt.

Codebeispiel

Dieser Artikel enthält keine Code-Snippets. Stattdessen finden sich unter ix.de/z7n2 sowohl der Quellcode zu allen Screenshots im Artikel als auch die Verweise zu den Language Models, den Auswahlmethoden und dem MNIST-Datensatz. Die vorgestellten Verfahren lassen sich durch Änderungen im Code den eigenen Bedürfnissen anpassen.

k-Means

Der k-Means-Algorithmus gehört zu den partitionierenden Verfahren und ist ein vergleichsweise naiver Ansatz, der jedoch häufig zum Erfolg führt und für eine Reihe von Problemstellungen sehr gut geeignet ist. Jeder Cluster wird durch den Mittelwert seiner Mitglieder (Zentroid) repräsentiert. Durch ein iteratives Ping-Pong zwischen der Mittelwertberechnung und der Neuzuordnung von Punkten zum nächstgelegenen Zentroid minimiert sich der Abstand jedes Punkts zu seinem Zentroid (s. Abb. 1):

1. Wähle k zufällige Punkte als Zentren der k Cluster. Das sind die initialen Zentroiden c_k!
2. Berechne für jeden Punkt p die Abstände zu den Zentroiden c_k und ordne den Punkt dem nächstgelegenen zu! Diese Zuordung wird für jeden Cluster als m_k gespeichert.
3. Überschreibe die Zentroide c_k durch die neuen Mittelpunkte der Cluster und wiederhole Schritte 2 und 3, bis die Mitgliedschaften m_k konvergieren, sich also nicht mehr ändern!
4. Gib die Zentroiden c_k und die finalen Clustermitgliedschaften der Punkte m_k zurück!

Besonders am Anfang des Prozesses stellt sich die Frage, wie viele Cluster es zu erzeugen gilt. Auch wenn die Aufgabenstellung eine Antwort liefern kann, ist es ratsam, die Anzahl `k` zu variieren und dabei die Kostenfunktion zu betrachten. Dabei kommt häufig die sogenannte Ellenbogenmethode zum Einsatz: Sinnvoll ist die Cluster-Anzahl `k`, bei der die Kostenfunktion den größten Knick macht. Die Kostenfunktion bei k-Means lässt sich durch die Summe der Fehlerquadrate bestimmen, konkret also die Summe der Abweichungen aller Punkte zu ihren Cluster-Zentren:

$$cost = \sum_{j=1}^{k} \sum_{p_i \in m_j} ||p_i - c_j||^2$$

k-Means hat zwei Vorteile:

- Es ist einfach zu implementieren und
- die Ausführung ist schnell.

Die Nachteile sind folgende:

- Ein Data Scientist muss den Startwert k bestimmen und sollte mehrere Werte testen.
- Nicht alle Verteilungen lassen sich gut segmentieren. Das gilt insbesondere für verschlungene Cluster oder Cluster mit unterschiedlichen Dichten.
- Bei hohen Dimensionen lassen sich die Clusterzugehörigkeiten aufgrund der großen Abstände nicht mehr einfach bestimmen.
- Das Verfahren erkennt keine Ausreißer.

Ein typischer Anwendungsfall für k-Means wären das Positionieren von Verteilzentren und die Definition von Zuständigkeitsbereichen, weil das Verfahren die Fahrtzeiten zu den Verteilzentren minimiert.

DBSCAN

DBSCAN [3] steht für Density-Based Spatial Clustering of Applications with Noise und ist im Gegensatz zu k-Means in der Lage, Ausreißer (Rauschen, engl. Noise) zu erkennen. Der Fokus liegt jedoch bei der Dichteverteilung der Datenpunkte, also ihrem jeweiligen Abstand zueinander. Damit eignet sich der Algorithmus für Cluster jeglicher Form. Ein weiterer wichtiger Vorteil gegenüber k-Means ist, dass es nicht nötig ist, die Anzahl der Cluster vorab zu schätzen, da so viel Cluster wie nötig entstehen. DBSCAN definiert drei Arten von Punkten: Kernpunkte sind dichte Punkte, die eine bestimmte Anzahl weiterer Punkte in ihrer Nachbarschaft haben. Sie definieren letztlich was zu einem Cluster gehört und was nicht. Randpunkte oder Dichte-erreichbare Punkte sind Datenpunkte, die selbst keine Kernpunkte sind, aber in der Nachbarschaft mindestens eines Kernpunktes liegen. Sie sind damit Teil eines Clusters, erweitern aber seine Ausdehnung nicht. Alle anderen Punkte heißen Rauschpunkte beziehungsweise Ausreißer. Sie sind nicht Teil eines Clusters. Somit benötigt der Algorithmus zwei Parameter: Den Nachbarschaftsradius ε , der den maximalen Abstand definiert, bei denen jeweils zwei Punkte als benachbart zählen, und `minPTS`, der angibt, ab wie vielen Nachbarn ein Punkt als Kernpunkt gilt.

Der Ablauf lässt sich folgendermaßen beschreiben:

1. Finde die Nachbarpunkte innerhalb des Radius ε um jeden Punkt!
2. Identifiziere alle Kernpunkte mit mindestens `minPTS` Nachbarn!
3. Füge alle Kernpunkte, die zueinander weniger als ein ε entfernt sind, demselben Cluster hinzu!
4. Prüfe für jeden Punkt, der kein Kernpunkt ist, ob es ein Randpunkt oder ein Ausreißer ist, und weise die Nachbarn einem Cluster zu.

Für die Wahl der richtigen Parameter gilt die Daumenregel, dass `minPTS` zwei mal so groß wie Dimensionalität des Datensatzes ist. Ein guter Startwert für ε lässt sich über einen sogenannten k-nearest-neighbor-Graphen mit `k = minPTS-1` bestimmen. Abbildung 4 zeigt, aufsteigend sortiert, den kleinsten Abstand der Nachbarn für jeden Punkt. Das richtige ε befindet sich an der Stelle mit der größten Änderung.

Übliche Anwendungsfälle für unüberwachtes Lernen

Unüberwachte Verfahren erkennen Strukturen (Gruppen, alternative Dimensionen) in Daten und machen sie explizit verfügbar.

- Semi-Supervised Learning, auch Pre-Training genannt: Weltwissen aufbauen. Durch unüberwachte Verfahren soll möglichst viel über die Struktur der Daten erlernt werden, um später überwachte Problemstellungen einfacher zu lösen. Ein Beispiel hierfür sind Embeddings wie Wortvektoren (s. „Aus dem Zusammenhang“ auf Seite 84).
- Datenanalyse / Exploration: Im Umfeld von Data Mining wollen Data Scientists Strukturen in den Daten finden, um einen besseren Zugang zu unbekannten Datensammlungen zu haben. Nach Ähnlichkeit gruppierte Datenpunkte erlauben sowohl eine bessere Übersicht als auch Drill-Down-Operationen bis hinunter auf die Rohdaten.
- Visualisierung: Komplexe Daten mit vielen Attributen lassen sich nicht ohne Weiteres sinnvoll darstellen. Zweidimensionale Strukturen (Streudiagramme), die möglichst viele Eigenschaften der Daten beibehalten, können unüberwacht erlernt werden.
- Kompression: Je mehr Strukturinformationen dem Kompressor zur Verfügung stehen, umso effektiver kann er Daten komprimieren. Beispielsweise ist es mit ausreichend Wissen über die charakteristischen Gesichtsmerkmale möglich, ein Gesicht mit Eigenschaften wie rundlich, mit Stirnfalten, randlose Brille effizient zu beschreiben und daraus wieder ein passendes Bild zu rekonstruieren.
- Anomalie-Erkennung: Wenn die normale Struktur der Daten bekannt ist, lässt sich bei neuen Datenpunkten beurteilen, ob sie in die Struktur passen. Alles, was nicht passt, ist wahrscheinlich neu und ungewohnt.

Bei der Auswahl eines größeren Radius fasst das Verfahren an dieser Stelle schlagartig mehr Punkte in einem Cluster zusammen, weil eigentlich gut trennbare Cluster verschmelzen. Umgekehrt wächst die Zahl der benötigten Cluster stark an, wenn der Radius zu klein ist und die Cluster feingranular werden. Nichtsdestotrotz ist es insbesondere für größere Datensätze hilfreich, die Parameter zu variieren und die Ergebnisse zu vergleichen.
Die Vorteile von DBSCAN sind:

- Das Verfahren ist bei beliebig geformten Gruppen anwendbar,
- es findet die Anzahl der Cluster selbstständig und
- kann Ausreißer erkennen

Das Verfahren hat dabei folgende Nachteile:

- Für dünne und hochdimensionale Datensätze ist es nicht gut geeignet,
- es reagiert sehr sensibel auf die Wahl von ε und
- unterschiedliche Dichten sind wegen des globalen Parameters ε problematisch.

DBSCAN entstand ursprünglich für räumliche Daten in der Geografie, bei denen es viele Gruppen von Entitäten gibt, die komplex geformt sind und in einem einheitlichen Koordinatensystem leben. Beispiele sind Bergzüge, Wälder und Siedlungen.

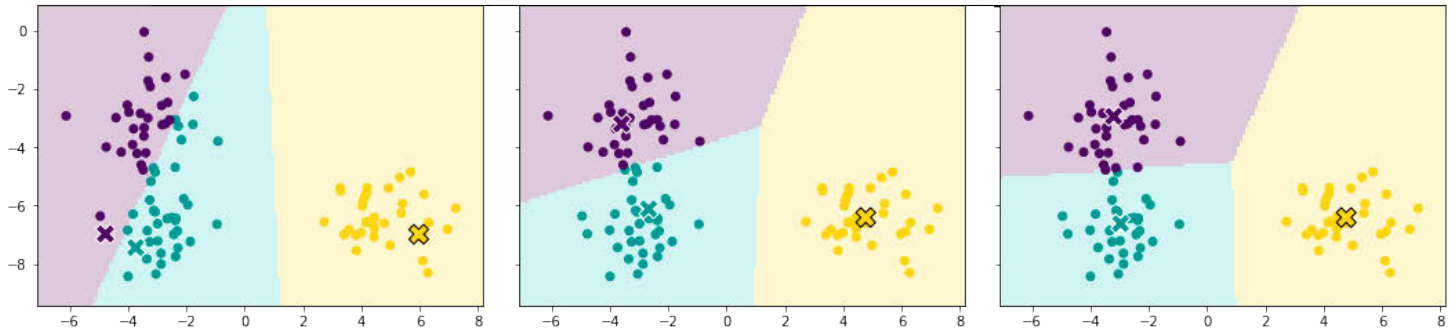
Hierarchisches Clustering

Hierarchisches Clustering ist eine weitere distanzbasierte Verfahrensart, bei der aber das Erstellen einer Cluster-Hierarchie im Vordergrund steht. Im Gegensatz zu den bisher gezeigten Verfahren können Datenpunkte dabei zu mehreren (hierarchisch verbundenen) Clustern gehören. Zum hierarchischen Clustering existieren zwei Grundtypen: das divisive und das agglomerative Clusterverfahren. Beim divisiven dient ein einziger großer Cluster als Ausgangsbasis, der alle Punkte beinhaltet. Anschließend erfolgt schrittweise die Aufteilung in Untercluster (Top-down).

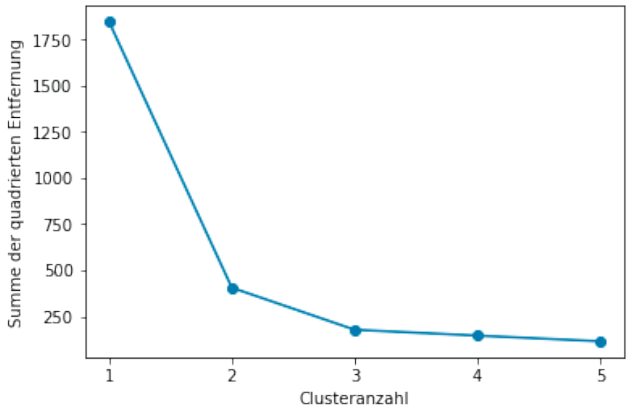
Beim wesentlich gebräuchlicheren agglomerativen Verfahren (Hierarchical Agglomarative Clustering, kurz: HAC) sind alle einzelnen Punkte selbst Cluster, die distanzbasiert bei jedem Schritt paarweise zu neuen Clustern zusammengefasst werden (Bottom-up). In beiden Fällen entsteht ein Binärbaum aus Clustern, das sogenannte Dendrogramm, das Gruppen innerhalb des Datensatzes repräsentiert. Voraussetzung für das HAC-Verfahren ist neben einem definierten Distanz- bzw. Ähnlichkeitsmaß für die Datenpunkte also auch auch ein Maß für die Ähnlichkeit zwischen zwei Clustern (Fusionsalgorithmus oder engl. Linkage Criteria). Eine zweidimensionale Verteilung, bei der sich der euklidsche Abstand als Distanzmaß anbietet, dient als Beispiel. In dem Fall entspricht ein kleiner Abstand einer großen Ähnlichkeit, wobei `0` die maximale Ähnlichkeit beziehungsweise Gleichheit von Datenpunkten darstellt. Das ist jedoch nur eine von vielen Möglichkeiten, die der Veranschaulichung dient.

Über `linkage criteria` lässt sich dem Clusteringverfahren ein angepasstes Verhalten mitgeben. Die gebräuchlisten sind

- Single Linkage: minimaler Abstand aller Punktpaarungen zwischen den beiden Ausgangs-Clustern. Ohnehin nahe beieinander liegende Cluster verschmelzen dabei. Der Cluster hangelt



k-Means über drei Iterationen. Die Zentroide wandern zum Schwerpunkt einer Gruppe und jeder von ihnen versucht seine eigene Gruppe zu definieren (Abb. 1).

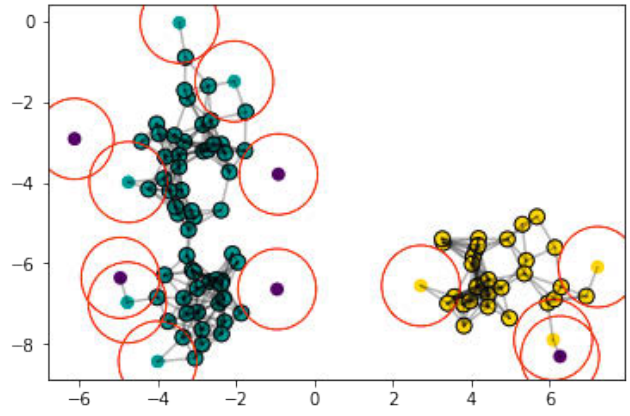


Ab drei Clustern sinken die Kosten nicht mehr nennenswert. Da in diesem Datensatz zwei Cluster sehr nahe beieinanderliegen, wären sowohl zwei als auch drei Cluster sinnvolle Werte (Abb. 2).

sich ähnlich zu DBSCAN an dichten Formen entlang, ist aber wesentlich anfälliger gegen Rauschen.

- Complete Linkage: maximaler Abstand aller Punktpaarungen zwischen den beiden Ausgangs-Clustern. Der Ansatz versucht, eine große Ausdehnung des Clusters lange zu vermeiden und erzeugt daher einen ausgeglicherenen Baum mit kompakten Strukturen.
- Average Linkage oder Centroid Linkage: gemittelter Abstand zwischen den Punkten beider Cluster, bei Average als Mittel der Abstände und bei Centroid als Abstände der Mittelpunkte. Der Ansatz ist ähnlich zu Complete Linkage, aber durch die Mittelung weniger anfällig gegen Ausreißer. Er erzeugt ähnliche Strukturen wie k-Means.
- Ward Linkage berechnet für jede mögliche Fusion, wie stark die Varianz im Cluster steigen würde, und wählt dann diejenige mit der geringsten Verschlechterung. Durch diese stabile und nachvollziehbare Wahl ist es der Default in den meisten HAC-Implementierungen.

HAC berechnet eine vollständige Hierarchie, indem es die Cluster so lange fusioniert, bis nur noch einer übrig ist. Dabei merkt es sich bei jeder Fusion den Abstand der ursprünglichen Cluster. Um tatsächlich Gruppen aus der Hierarchie zu erzeugen, gilt es die Cluster nachträglich zu schneiden. Dazu wird der Binärbaum von der Wurzel aus wieder aufgetrennt, und zwar immer entlang der Fusionen mit dem größten Abstand, bis entwe-



DBSCAN hat mit den gewählten Einstellungen zwei Cluster identifiziert (Farbe der Punkte). Jeder der Kernpunkte (schwarzer Rahmen), die ein Cluster definieren, ist mit mindestens drei anderen Punkten in der Umgebung verbunden. Die Randpunkte sind in der Umgebung (rote Kreise) von mindestens einem Kernpunkt, erweitern den Cluster aber nicht. Ausreißer sind lila gefärbt und haben keinen Kernpunkt in ihrer Umgebung (Abb. 3).

der die gewünschte Anzahl an Clustern entsteht oder ein Schwellwert für den maximalen Abstand unterschritten wird.
Die Vorteile sind:

- HAC benötigt keine Vorabinformationen und keine Parameterwahl,
- es ist einfach zu implementieren und
- erreicht gute Ergebnisse bei hierarchisch strukturierten Daten.

Als Nachteile von HAC schlagen folgende Faktoren zu Buche:

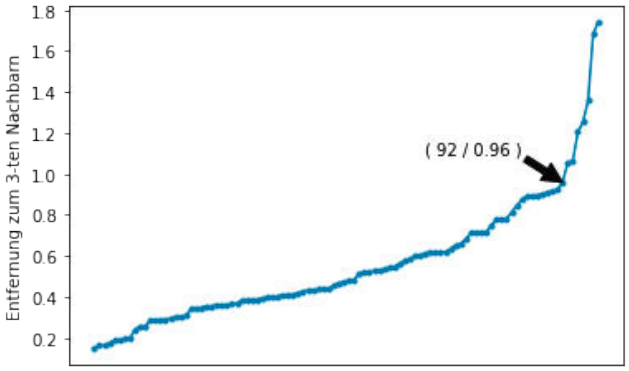
- Wegen der hohen Laufzeitkomplexität $O(n^2 \log n)$ ist es für sehr große Datensätze nicht geeignet,
- je nach Wahl des Fusionstyps ignoriert es Ausreißer oder fragmentiert große Cluster unnötig,
- das Verfahren kann nicht gut mit stark unterschiedlichen Clustergrößen umgehen.

HAC ist dann sinnvoll, wenn der Datensatz selbst eine hierarchische Struktur besitzt. Ein typisches Beispiel sind Ähnlichkeitsanalysen oder Stammbaumforschung unter anderem von Bakterien und Viren. Außerdem lässt sich das Verfahren gut für hierarchische Sortierungen und Visualisierungen verwenden.

Reduzierte Dimensionen

Viele Anwendungsfälle von Machine Learning arbeiten mit unstrukturierten Daten. Während Data Scientists bei tabellarischen Daten wie den beschriebenen Immobilienkennzahlen jedes Feature über seinen Spaltennamen halbwegs verständlich in Bezug zur Problemstellung setzen können und damit zur Not einfach Regelsysteme eigenhändig programmieren könnten, fehlt bei unstrukturierten Daten oft jeglicher Zugang zu einem klassischen Ansatz. Das macht das maschinelle Lernen zu einem besonders gewinnbringenden Ansatz beispielsweise bei Bildern und Texten.

Durch diese fehlende Struktur in den Rohdaten entstehen oft sehr viele Dimensionen, die alle gleichberechtigt nebeneinander stehen. Selbst kleine Bilder setzen sich aus tausenden Bildpunkten zusammen, die jeweils für sich genommen fast keinen Beitrag zum Inhalt des Bilds liefern. Ebenso sind Texte wesentlich mehr als eine Aneinanderreihung von Buchstaben. Auch wenn ein „Haus“ und eine „Maus“ von der Buchstabenfolge ähnlich sind, sollten

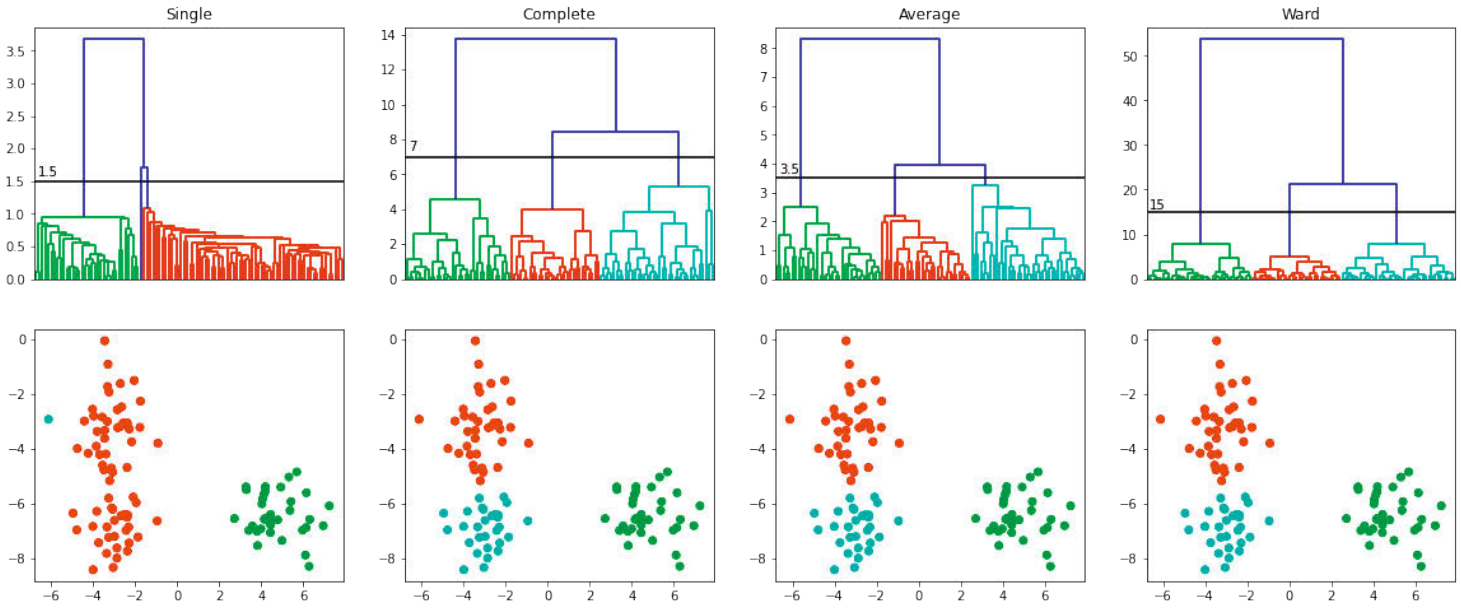


Die meisten Punkte (92 von 100) haben ihre drei Nachbarn in einer Umgebung von ca 1.0 oder weniger. Dann steigt der Abstand rapide an. Die Randpunkte scheinen also einen Abstand von mindestens 1.0 zum nächsten Cluster zu haben, was sich deshalb als Startpunkt für Epsilon anbietet (Abb. 4).

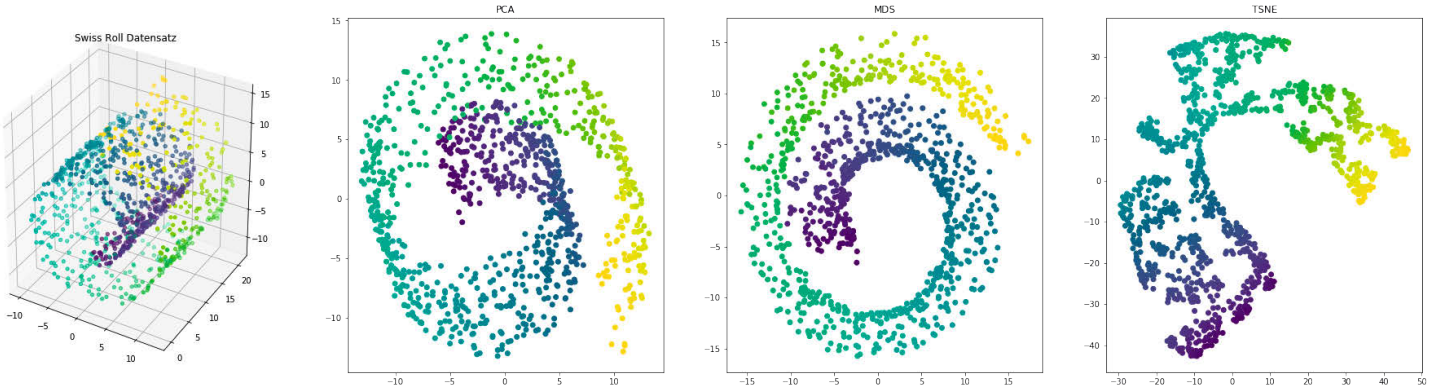
sie auf Datenebene eine größere Distanz haben als beispielsweise „Haus“ und „Gebäude“. Daher kommen bei Texten meist Wörterbücher zum Einsatz, und jedes Wort im Text bekommt eine eigene Dimension zugewiesen. So lässt sich zumindest garantieren, dass keine „falschen Freunde“ entstehen, weil jedes Wort den gleichen Abstand zu jedem anderen Wort hat. Leider können dabei je nach Bandbreite der Texte Hundertausende Dimensionen entstehen.

Dass hochdimensionale Daten Probleme mit sich bringen, hat den Begriff „Fluch der Dimensionalität“ geprägt. Drei naheliegende Effekte sind im Folgenden beschrieben:

- Verständnis der Daten: Menschen haben Schwierigkeiten, sich mehr als vier Dimensionen vorzustellen. Bei Tausenden nichtssagenden Dimensionen ist die Arbeit mit den Daten ohne Hilfsmittel schlicht unmöglich. Auch wenn sich einzelne Bilder oder Texte noch anzeigen lassen, sind Aussagen über eine Menge von Datensätzen schwierig. In diesen Fällen sind intuitive Aggregationen wie Durchschnitte oder Mediane nicht mehr sinnvoll einsetzbar.
- Effizienz der Berechnung: Jedes Modell muss letztlich jeder Kombination von Eingabedimension und Ausgabedimension mindestens ein Gewicht zuordnen. Damit wächst die Menge an Gewichten exponentiell mit den Dimensionen. Bei mehrschichtigen Modellen wie den meisten neuronalen Netzen kommen



Die Dendrogramme helfen bei der Wahl des Hierarchieschnitts (wie viele Cluster oder welchen Abstand als Threshold). Single Linkage kann die beiden linken Cluster nicht trennen weil sie ineinander übergehen (Abb. 5).



Der Swiss Roll Datensatz ist verschlungen. PCA findet zwar einen guten Schnitt durch die Daten, kann aber Überlappungen nicht vermeiden. MDS kann die Überlappung auseinanderziehen und trotzdem die globale Struktur erhalten. TSNE schafft es den Datensatz wieder komplett abzurollen (Abb. 7).

noch Zwischenschichten hinzu. Dadurch werden die Modelle äußerst groß und rechenintensiv im Training. Netzwerkstrukturen wie Convolutional Layer ermöglichen zwar das Teilen von Gewichten, sind jedoch nur sinnvoll einsetzbar, wenn die Nachbarschaft von Dimensionen relevant ist. Glücklicherweise ist das bei Bildern und Texten mit benachbarten Pixeln beziehungsweise aufeinanderfolgenden Worten der Fall, aber bei rein kategorischen Daten wie Produkten im Online-Versandhandel sind sie nicht anwendbar.

- Datenhunger: Mit jeder zusätzlichen Dimension steigt der Feature-Raum, also der Raum der möglichen Ausprägungen und Eigenschaften, multiplikativ. Um genauso verlässliche Modelle trainieren zu können, müsste also der Datensatz exponentiell mitwachsen. Mit der Vorstellung, dass ein Modell den Erfahrungsschatz einer KI wiedergibt, wäre die Analogie, dass sich bei einem sehr großen Feature-Raum nie eine passende Erfahrung abrufen lässt: Alles ist neu und alle halbwegs ähnlichen Beispiele sind in vielen Dimensionen verschieden.

Insbesondere aufgrund des letzten Punkts sind unüberwachte Verfahren so relevant für künftige KI-Anwendungen und in nahezu allen aktuellen Verfahren als Pre-Training involviert. Niemand kann genug Daten labeln, um die Komplexität der Welt zu erfassen, aber das Internet bietet sehr viele ungelabelte Daten, um komplexe Modelle zu trainieren. Das Thema Pre-Training ist umfangreich genug, dass es einen separaten Artikel ergeben würde.

Auf das Wesentliche konzentrieren

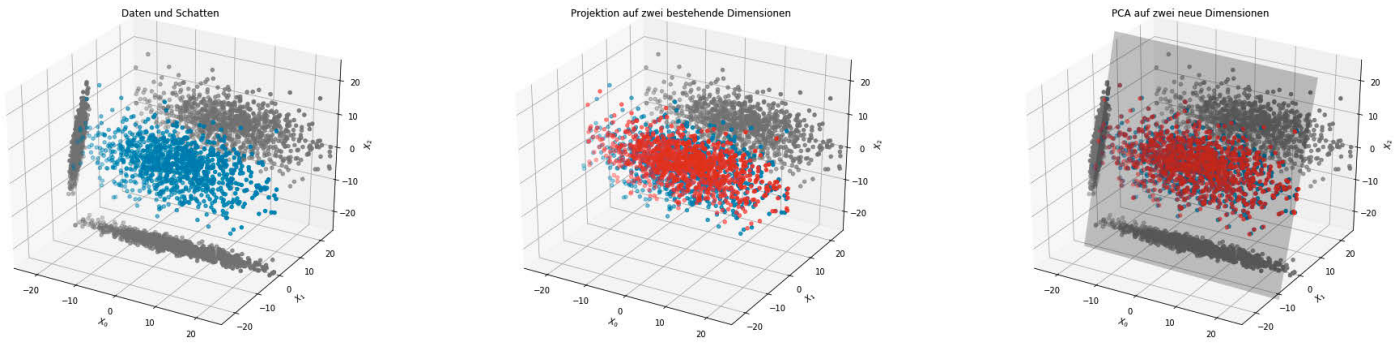
Mit dem Wegwerfen von Dimensionen, um ihre Anzahl zu reduzieren, ist immer ein gewisser Informationsverlust verbunden. Verfahren zur Dimensionsreduktion codieren diesen und versuchen ihn zu minimieren. Die gewünschte Zahl der verbleibenden Dimensionen lässt sich entweder explizit aus der Zielsetzung ableiten (für die Visualisierung von Daten typischerweise die Höhe und die Breite) oder implizit anhand des tolerierbaren Informationsverlusts wählen.

Im einfachsten Fall, der Feature Selection (FS), sortieren Data Scientists die Dimensionen und entfernen so lange die unwichtigste Dimension, bis das Ziel erreicht ist. Hierfür benötigen sie ein Maß für die Wichtigkeit der einzelnen Merkmale. Während sie für überwachte Verfahren die Korrelation zwischen dem Merkmal und der Zielvariable (Ground Truth) heranziehen können, lässt sich im unüberwachten Fall die Varianz als Basis verwenden. Allein genommen hat sie jedoch keine hohe Aussagekraft. Da für die meisten Lernverfahren die Merkmale standardisiert, also auf den Mittelwert zentriert und durch die Standardabweichung geteilt werden, haben Merkmale normalerweise jeweils eine Varianz von eins. Daher schließt der einzige FS-Algorithmus in scikit-learn für unüberwachte Problemstellungen (`VarianceThreshold`) von Haus aus nur Merkmale aus, die alle denselben Wert und damit gar keine Varianz oder Aussagekraft haben.

Sinnvoller lassen sich die Merkmale durch das Betrachten im Zusammenhang gewichten. Saúl Solorio Fernandez hat zusammen mit anderen Autoren [4] eine anschauliche Übersicht über Feature-Selection-Verfahren für unüberwachtes Lernen erstellt. Dabei stellen sie sowohl Filtermethoden, die die Zusammenhänge explizit erheben, als auch Wrapper-Methoden vor. Letztere messen, wie viel schlechter bestehende Clustering-Verfahren abschneiden, wenn aus dem Datensatz ein Merkmal gezielt entfernt wird.

Kombinieren vor Projizieren

Eine beliebte Möglichkeit, sowohl die Beziehungen zwischen den Dimensionen als auch den Informationsverlust zu evaluieren, ist die sogenannte Principal Component Analysis (PCA). Im Prin-



Die Originalpunkte in blau und die aus der niedrigeren Dimension rekonstruierten Punkte in rot sind für Projektion und PCA unterschiedlich deckungsgleich (Abb. 6).

zip ist PCA wie FS eine Projektion auf eine Hyperebene mit weniger Dimensionen. Aber während bei FS die Projektion orthogonal zu einer Dimension erfolgt und dadurch die Informationen dieser Dimension komplett verloren gehen, versucht PCA den Datensatz vorher optimal zu drehen, sodass möglichst wenig Varianz auf der Strecke bleibt. Als Analogie lässt sich eine Punktwolke so lange im Licht drehen, dass die Schatten der Punkte einen möglichst großen Bereich abdecken. Mathematisch betrachtet entspricht es der Fragestellung, welche lineare Kombination der Dimensionen die größte Varianz erhält – und damit die größte Unterscheidbarkeit der Datenpunkte.

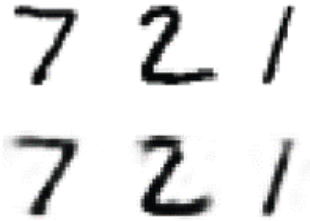
Eine reine Projektion des in Abbildung 6 dargestellten Datensatzes aus blauen Punkten im dreidimensionalen Raum mit den Dimensionen x_0, x_1 und x_2 auf eine zweidimensionale Ebene, würde ihn auf einen der grauen „Schatten" reduzieren. Entlang der Dimension x_1 geht die wenigste Information verloren. Jedoch sieht man an den aus dem hinteren Schatten rekostruierten roten Punkten, dass sie durch die verlorene Information nicht auf den selben Positionen wie die Ursprungsdaten landen. Die PCA würde vorher eine schiefe Ebene aus x_0 und x_2 erzeugen (graue Ebene) und dann erst projezieren. Die Rekonstruktion ist dadurch wesentlich näher am Original.

Somit lässt sich in vielen Fällen mit PCA die Zahl der Dimensionen reduzieren und trotzdem eine hohe Aussagekraft beibehalten, da sinnvolle neue Dimensionen vor der Reduktion abgeleitet werden. Ähnlich zu den Problemen bei k-Means, das keine verschlungenen Cluster finden kann, gibt es auch Fälle, in denen sich keine gute lineare Kombination für eine Projektion finden lässt. Der Swiss-Roll-Datensatz zeigt das sehr schön.

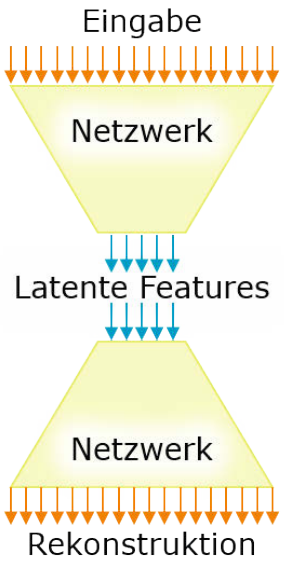
Verschlungene Daten sinnvoll reduzieren

Die Struktur einer Biskuitrolle (engl. Swiss Roll) ist eigentlich ein rechteckiger Biskuitboden. Er wurde jedoch aufgerollt. Punkte die im ursprünglichen Boden weit entfernt lagen, liegen plötzlich aufeinander. Wenn der Backofen eine ungleiche Hitzeverteilung hatte, ist diese Eigenschaft wild im dreidimensionalen Raum verteilt. Es hilft auch nicht, die Rolle wie bei einer PCA zu drehen, da immer Punkte, die eigentlich nicht in Beziehung zueinander stehen nebeneinander liegen. Man müsste den Teig erst wieder abrollen, um das Muster zu erkennen. Genau dieses Ziel verfolgen die Verfahren des Manifold Learning (MF), indem sie zugunsten der lokalen Nachbarschaften die globale Struktur vernachlässigen. Während PCA nur lineare Kombination von Dimensionen erlaubt, können MF-Verfahren lokal unterschiedlich gewichten und damit auch nichtlineare Strukturen abbilden. Die prominentesten Vertreter sind das Multi-Dimensional-Scaling (MDS) und das T-Stochastic Neighbor Embedding (TSNE).

MDS nimmt als Basis die Distanzen im hochdimensionalen Raum und versucht sie im niedrigdimensionalen Raum so gut wie möglich beizubehalten. Als Bild ist jeder Datenpunkt mit jedem anderen Datenpunkt über eine Metallfeder zu einem vollständigen Netzwerk verbunden. Die Länge der Federn entspricht dabei dem Abstand der verbundenen Punkte im hochdimensionalen Raum



Die oberen Ziffern wurden durch einen Autoencoder jeweils auf 4 % der Ausgangsgröße komprimiert und dann wieder zu den unteren Ziffern decodiert (Abb. 9).



Struktur eines Autoencoders: Das Lernziel ist die orangene Eingabe möglichst unbeschadet durch den blauen Flaschenhals zu bekommen (Abb. 8).

und ist dort ohne jegliche Spannung. Beim Drücken des Graphen in eine zweidimensionale Ebene müssten sich die Federn dehnen und stauchen. Die Konfiguration, bei der die Spannung in den Federn minimal ist, erhält die Abstände der Datenpunkt am besten. Lokale Überdeckungen werden dadurch etwas entzerrt, aber die globale Struktur trotzdem beibehalten.

TSNE ist in der Visualisierung beliebt, weil es Zusammenhänge in jeder Größenordnung darstellen kann. Bereiche hoher lokaler Dichte bläht es stark auf, während es leere Räume staucht. Auf die Weise nutzt es die vorhandene Fläche gut aus und erhält gleichzeitig die vorhandenen Cluster. Leider ist TSNE nicht parameterfrei, sodass es mehrere Anläufe benötigen kann, bis die richtigen Werte für eine gute Darstellung gefunden sind.

Autoencoder: Struktur durch Kompression

Eine etwas besondere Rolle nehmen Autoencoder (s. Kasten 2) ein. Strukturell handelt es sich dabei um neuronale Netze mit zwei namensgebenden Eigenschaften. Erstens bilden sie die Eingabe auf sich selbst ab (**Auto**). Das bedeutet, dass die gewünschte Ausgabe des Netzwerks exakt der Eingabe entspricht. Damit sind keine Labels nötig, und der Autoencoder ist ein unüberwachtes Verfahren im weiteren Sinne, auch wenn es sich mit dem neuronalen Netz eines klassisch überwachten Lernverfahrens bedient. Zweitens hat das Netzwerk in der Mitte des Informationsflusses einen Flaschenhals, an dem das Signal in einen komprimierten Code **enkodiert** wird. Dieser Code hat eine fixe Dimensionalität, die geringer als die der Eingabe ist.

Das Lernziel des Autoencoder (s. Abb. 8) ist schlicht, die Gewichte des Netzwerks so zu wählen, dass der Unterschied zwischen Ein- und Ausgabe minimal ist. Die beiden Eigenschaften zusammen zwingen den Autoencoder sowohl eine Kompression als auch die dazugehörige Rekonstruktion zu erlernen. Je kleiner der Flaschenhals ist, desto mehr Strukturinformationen muss der Autoencoder nutzen, um sein Ziel zu erreichen. An der dünnsten Stelle, dem als latente Features bezeichneten Code, muss die Aussagekraft der Dimensionen deutlich höher sein als bei der Ein- und Ausgabe.

Was die latenten Features darstellen, ergibt sich aus den Daten und der Struktur des Netzes. Beim Anwenden eines Autoencoder auf Bilder von handgeschriebenen Ziffern beschreiben die Eingaben beispielsweise noch einzelne Pixel. Beim MNIST Datensatz mit 28 x 28 Pixel (s. Abb. 9) ergeben sich somit 784 verschiedene Dimensionen. Der Encoder wandelt sie in charakteristische Bögen, Linien und Ecken um. Damit ist er für das Beispiel in der Lage, auch mit wenigen Dimensionen – konkret 32 latenten Features, die etwa vier Prozent der Ursprungsgröße entsprechen – eine Ziffer gut zu beschreiben und durch den Decoder wieder in das Ausgangsbild umzuwandeln, ohne dabei starke Artefakte zu erzeugen.

Anomalie-Erkennung

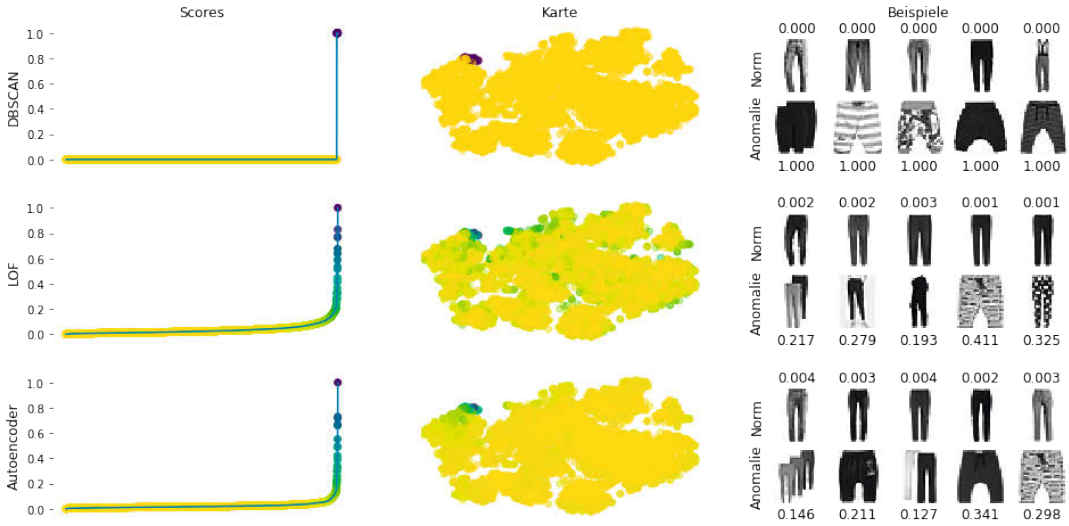
Eine Anomalie ist gemeinhin als Abweichung vom Normalen oder als Unregelmäßigkeit definiert. Anomalien sind oft interessant, weil sie entweder auf ein Problem beispielsweise in den Systemen oder in der Datenqualität oder auf eine Neuigkeit hinweisen. Um die Norm oder die Regel zu definieren, eignen sich die beschriebenen Aufgabenstellungen: Clustering, Dimensionsreduktion und insbesondere die Autoencoder.

Anomalien sind schwer zu beschreiben, weil sie von der Natur der Sache eher unbekannt und überraschend sind. Deshalb eignen sich zum Erkennen unüberwachte Lernverfahren wesentlich besser als beispielsweise eine Klassifikation in „normal“ und „nicht normal“, da keine Beispiele für nicht normale Fälle zur Hand sind. Stattdessen versucht das System, die Struktur der normalerweise beobachteten Daten zu erlernen und schlägt Alarm, wenn die aktuellen Daten nicht mehr in diese Struktur passen.

Für die im Folgenden vorgestellten Verfahren dient der Fashion-MNIST-Datensatz [5] als Grundlage. Er ist analog zum originalen MNIST-Datensatz einfach strukturiert (28 x 28 Pixel mit Grauwerten), aber mit etwas anspruchsvolleren Klassen (Modeartikel statt Zahlen). Der Vergleich erfolgte nur anhand der Trainingsbilder der Klasse „Hosen". Damit ist der Berechnungsaufwand für diejenigen, die die Bilder des Vergleichs selbst berechnen wollen, überschaubar. Für den Vergleich wurden alle Werte in den Bereich 0 bis 1 normiert.

In der oberen Zeile von Abbildung 10 sind die Ergebnisse der Ausreißererkennung von DBSCAN zu sehen. Die Bewertung ist rein binär, da die Bilder entweder als normal (0) oder als „Rauschen" (1) markiert sind. Die Karte zeigt eine Dimensionsreduktion aller 784 Pixelwerte auf zwei Werte mittels TSNE. Das zeigt, dass alle Ausreißer (Outlier) aus einer kleinen Region kommen und sich untereinander scheinbar doch ähneln. Das spiegelt sich auch in den Beispielen wider, in denen vor allem kurze Hosen und Babykleidung als Ausreißer erkannt wurden, wärend alle langen Hosen als normal gelten. Der LOF-Score (Local Outlier Factor) in der zweiten Zeile gibt eine feingranularere Einschätzung und kann lokale Unterschiede besser berücksichtigen. Er findet damit in jedem Bereich des Datenssatzes ungewöhnliche Bilder wie doppelte Hosen oder Personen. Der Autoencoder hatte ebenfalls vor allem mit den kurzen Hosen Schwierigkeiten, in erster Linie wegen wegen des verwendeten, einfachen Modells.

Der clusteringbasierte Ansatz zur Anomalie-Erkennung geht davon aus, dass sich normale Daten sinnvoll gruppieren lassen, während sich abnormale Daten als Ausreißer darstellen. Daher eignet sich jede Methode, die Ausreißer erkennen kann, prinzipiell für die Anomalie-Erkennung. Die Unterscheidung, etwas als Anomalie oder als Outlier zu bezeichnen, betrifft eigentlich lediglich den Zeitpunkt der Erkennung. Letztere fallen beim Lernen eines Modells an, wenn Datenpunkte zu ungewöhnlich sind und deshalb die gelernten Strukturen verfälschen würden. Deshalb gilt es typischerweise, sie aus dem Datensatz zu entfernen. Eine Anomalie ist ein Ausreißer während des Produktivbetriebs, und der Umgang mit ihnen (verwerfen oder alarmieren) ist daher anwendungsfallabhängig.



Anwendung der Methoden auf den Fashion-MNIST-Datensatz (Abb. 10)

Manche Clusteringverfahren haben bereits eine explizite Outlier-Erkennung integriert. Beim oben vorgestellten DBSCAN steht das N im Namen für den bei der Clusterdefinition zu ignorierenden Noise. Letzterer beschreibt alle Datenpunkte, die zu isoliert stehen, um in einer definierten Nachbarschaft die Mindestanzahl an Nachbarn aufweisen zu können. Dieses Maß lässt sich auch nach dem Training nutzen, um neue Datenpunkte als passend oder nicht passend zu identifizieren.

Ohne den Clusteringaspekt, aber mit derselben Idee arbeiten dichtebasierte Ansätze wie der sogenannte Local Outlier Factor. Er vergleicht die lokale Dichte (Nachbarschaft) eines Datenpunkts mit der Dichte seiner Nachbarn. Wenn die Nachbarn eines Punktes untereinander sehr nahe liegen, der Punkt selbst aber einen großen Abstand zu ihnen hat, scheint er nicht der lokalen Norm zu entsprechen. Die Entscheidung ist daher nicht binär wie bei DBSCAN, sondern durch das Verhältnis der Dichten entsteht ein Ranking der Datenpunkte nach Ungewöhnlichkeit beziehungsweise Isolation (s. Abb. 11).

Kasten 2: Prominente Varianten von Autoencodern

Für unterschiedliche Aufgabestellungen existieren verschiedene Varianten von Autoencodern. Durch eine modifizierte Zielfunktion lassen sich gewisse Eigenschaften des Codes erzwingen:

- Sparse Autoencoder: Eine Regularisierung der Codeneuronen erzwingt, dass nur wenige Stellen im Code gleichzeitig hohe Werte haben. Das Ziel sind binäre, aussagekräftigere Dimensionen, die für Klassifikationsaufgaben nützlich sind.
- Variational Autoencoder: Durch das Auftrennen jeder Codedimension in einen Mittelwert und eine Varianz soll der Entstehungsprozess der Daten modelliert werden (generatives Modell). Das erzeugt kontinuierliche Wertebereiche, in denen jeder Zwischenwert sinnvolle Ergebnisse liefert.
- Denoising Autoencoder: Durch das Verfremden der Eingabedaten bei unveränderten erwarteten Ausgaben lernt der Autoencoder, Rauschen in den Daten zu ignorieren.

Beim sogenannten Fuzzy Clustering ist die Entscheidung, in welchen Cluster ein Datenpunkt gehört, nicht binär: Ein Datenpunkt kann anteilsweise zu mehreren Clustern oder auch zu keinem gehören. Von den Fuzzy-Clustering-Verfahren eignet sich das Gaussian-Mixture-Modell besonders zur Anomalie-Erkennung. Es modelliert den Datensatz als Mischung mehrerer Wahrscheinlichkeitsverteilungen und kann damit zu jedem neuen Datenpunkt berechnen, wie wahrscheinlich sein Auftreten ist. Mit einem Schwellwert lässt sich auf einfache Weise eine Anomalie-Erkennung realisieren. Für das Lernen der Wahrscheinlichkeitsfunktion gilt es, mehrere Werte für den Parameter zu testen und zu bewerten.

Passendes Vokabular

Eine Anomalie-Erkennung auf Basis der Dimensionsreduktion nutzt die Tatsache aus, dass Autoencoder ein höherwertiges Vokabular (die latenten Features) zur Repräsentation der Daten erlernen müssen. Dieses wird sich schlecht eignen, um etwas anderes als die erwarteten Daten zu beschreiben. Um das Beispiel mit den Gesichtsmerkmalen vom Anfang des Artikels aufzugreifen: Ein Phantombildzeichner kann ein Gesicht anhand der Beschreibung über die gelernten Merkmale wie Haarfarbe, Alter, Kopfform, Bart zumindest so erstellen, dass es dem Original ähnlich ist.

Ein getragener Mund-Nasen-Schutz lässt sich jedoch nicht mit dem bekannten Vokabular mitteilen, und das Phantombild wird höchstens einen Bart erhalten. Aus der Differenz zwischen Gesicht und Phantombild lässt sich schließlich die Anomalie erkennen. Das Beispiel geht davon aus, dass im Trainingsdatensatz des Autoencoder wenig bis gar keine Masken enthalten waren, da dieses Merkmal vermutlich sonst Bestandteil des erlernten Vokabulars wäre.

Zusätzlich zum Trainieren des Autoencoder gilt es bei der Anomalie-Erkennung, die Differenz zwischen Ein- und Ausgabe zu berechnen und bei einem zu hohen Wert Alarm zu schlagen. Als Indikator für den Schwellwert dient die Varianz der Differenz über den Trainingsdatensatz.

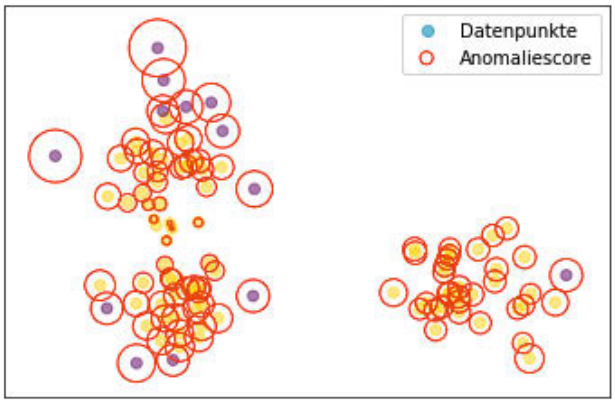
Blick nach vorn

Ebenfalls unüberwacht ist die Anomalie-Erkennung durch Vorhersage. Dabei sind die Messwerte von `x` Zeitpunkten die Eingabe und der `x+1`te Wert die erwartete Ausgabe. Damit lässt sich eine Regression trainieren, die den zukünftigen Wert vorhersagt. Mit demselben Vorgehen wie bei der Anomalie-Erkennung mit Autoencoder können Data Scientists den vorhergesagten und den tatsächlichen Wert vergleichen und bei zu hoher Abweichung Alarm schlagen.

Das mutet zunächst so an, als ob ein einfacher, gleitender Durchschnitt dasselbe Ergebnis liefern würde. In eine Regression lassen sich jedoch zusätzliche Kontextinformationen wie der Wochentag, die Uhrzeit oder die am System angemeldeten Personen integrieren und so auch plötzliche, aber erwartbare Schwankungen beispielsweise durch die Halbzeitpause in einem Fußballfinale vorhersagen und als normal erkennen.

Weites Feld

Die unüberwachten Lernverfahren und zugehörigen Aufgabenstellungen sind ein breites Feld im Machine Learning. Überall dort, wo ein menschliches Labeln nicht möglich oder kaum praktikabel ist, zeigt Unsupervised Learning seine Stärke. Dabei haben die genannten Verfahren und Beispiele teilweise lediglich die Oberfläche angekratzt.



Die mit dunkler Farbe markierten Ausreißer am Rand der Cluster haben einen höheren Local Outlier Factor (rote Umkreise), da sie weiter von ihren Nachbarn entfernt sind als ihre Nachbarn untereinander (Abb. 11).

Darüber hinaus existieren weitere interessante und teils komplexe Algorithmen wie Self Organizing Maps (SOMs) als Teilbereich des Manifold Learning. Neben der Kombination von unüberwachten und überwachten Lernverfahren im Pre-Training lassen sich viele Problemstellungen als unüberwachte Aufgabe modellieren, darunter Generative Adversarial Networks (GANs). Da die Entwicklung erst in den Anfängen steckt, gilt frei nach Stephen Hawking: Bleiben Sie neugierig! (rme@ix.de)

Literatur

[1] Tom B. Brown et al.; "Language Models are Few-Shot Learners"; arXiv preprint 2020

[2] Alec Radford et al.: "Improving language understanding by generative pre-training." 2018

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu; A density-based algorithm for discovering clusters in large spatial databases with noise. In: Evangelos Simoudis, Jiawei Han, Usama M. Fayyad (Hrsg.): Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press, 1996, ISBN 1-57735-004-9, S. 226–231

[4] Saúl Solorio-Fernández, J. Ariel Carrasco-Ochoa & José Fco. Martínez-Trinidad; A Review of Unsupervised Feature Selection Methods; Springer 2019

[5] Han Xiao and Kashif Rasul and Roland Vollgraf; "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms"; arXiv 2017

Harald Bosch

forschte für seine Promotion im Themenfeld Visual Analytics und hatte dadurch oft den Bedarf unbekannte Datensammlungen greifbar zu machen. Seine Erfahrungen bringt er seit 2017 in der Novatec Consulting GmbH in den Themengebieten AI & ML ein.

Arthur Varkentin

promovierte 2018 in Physik und setzte sich im Rahmen seiner Doktorarbeit mit Segmentierungs- und Klassifizierungsproblemen auseinander. Seit 2019 wirkt er bei der Novatec Consulting GmbH in den Bereichen AI & ML mit.