



MACHINE / DEEP LEARNING – WIE LERNEN KÜNSTLICHE NEURONALE NETZE?

KI BASICS



Ein wichtiges Merkmal von Systemen mit künstlicher Intelligenz ist die Fähigkeit, selbständig zu lernen. Anders als bei klassischer Software, die Probleme und Fragen auf Basis von vorher festgelegten Regeln abarbeitet, können selbstlernende Machine Learning Algorithmen die besten Regeln für

Suche

JAAI wird
präsentiert
von:



Wir sind
Partner von:



die Lösung bestimmter Aufgaben selber lernen. In diesem Artikel wird am Beispiel künstlichen neuronalen Netzen erklärt, was „lernen“ in diesem Zusammenhang heißt, und wie es funktioniert.



Ein [künstliches neuronales Netz \[https://jaai.de/kuenstliche-neuronale-netze-aufbau-funktion-291/\]](https://jaai.de/kuenstliche-neuronale-netze-aufbau-funktion-291/) besteht aus vielen einzelnen Neuronen, die meistens in mehreren miteinander verbundenen Schichten (Layern) angeordnet sind. Die Zahl der Layer bestimmt unter anderem den Grad der Komplexität, den ein künstliches neuronales Netz abbilden kann. Viele Layer machen ein neuronales Netz „tief“ – daher spricht man in diesem Zusammenhang auch vom *Deep Learning* als einer Unterkategorie des *Machine Learning* (*Maschinenlernen* bzw. *maschinelles lernen*).

Im Überblick: So lernen künstliche neuronale Netze

Das Lernen funktioniert dann wie folgt: Nachdem die Netz-Struktur aufgebaut wurde, erhält jedes Neuron ein zufälliges (!) Anfangs-Gewicht. Dann werden die Eingangs-Daten in das Netz gegeben, und jedes Neuron gewichtet die Eingangs-Signale mit seinem Gewicht und gibt das Ergebnis weiter an die Neuronen des nächsten Layers. Am Output-Layer wird dann das Gesamt-Ergebnis berechnet – und dieses wird in der Regel wenig mit dem bekannten tatsächlichen Ergebnis zu tun haben, da ja alle Neuronen ein zufälliges Anfangsgewicht haben. Man kann jedoch die Größe des Fehlers berechnen, und den Anteil, den jedes Neuron an diesem Fehler hatte, und

NEUESTE BEITRÄGE

[Google BERT](#)
=
[Sprachverarbeitung \(NLP\) durch Transformer-Modelle](#)
[Künstliche Intelligenz in der Bundesliga – SV Werder Bremen nutzt die intelligente Scouting Plattform SCOUTASTIC Word Embeddings – Methoden zur Repräsentation](#)

dann das Gewicht jedes Neurons ein kleines bisschen in die Richtung
ver  den Fehler minimiert. Dann erfolgt der nächste Durchlauf,
eine  essung des Fehlers und Anpassung der Gewichte und so
weiter. So „lernt“ ein neuronales Netz zunehmend besser, von den Input-
Daten auf die bekannten Output-Daten zu schließen.

Die nachfolgenden Passagen beschreiben die einzelnen Phasen und
Elemente dieses Lernprozesses im Detail.

Der Forward Pass

Auf der einen Seite des neuronalen Netzes werden die Input-Daten
eingespeist. Dabei wird jedes Input Signal an jedes einzelne Neuron des
ersten Layers verteilt. Jedes Neuron gewichtet dann das ankommenden
Signal mit einem Input-spezifischen Gewicht (das zu Beginn zufällig vergeben
wurde!), addiert einen sogenannten *Neuron-spezifischen Bias-Term* hinzu
und summiert alle so gewichteten Input Daten zum Output dieses einen
Neurons.

Oft wird der Output dabei noch durch eine nicht *lineare Aktivierungsfunktion*
geleitet, um z.B. einen bestimmten Wertebereich des Outputs zu erzwingen
(z.B. Sigmoid, tanh, etc.). Der Output jedes Neurons wird dann als Input an
alle Neuronen des folgenden Layers weitergegeben. Dieser Prozess setzt sich
fort, bis der Output-Layer erreicht wird, der das Ergebnis aller Berechnungen
liefert.

von Wörtern
in
Algorithmen
und
neuronalen
Netzen
Transfer
Learning – So
können
neuronale
Netze
voneinander
lernen
Convolutional
Neural
Networks –
Aufbau,
Funktion und
Anwendungsg

KATEGORIE

Allgemein
Amazon
IBM Watson

Die Messung des Fehlers



Bis hierhin hat das künstliche neuronale Netz noch nichts gelernt. Da alle

Gewichte bei der Initialisierung eines neuronalen Netzes zufällig innerhalb eines vorgegebenen Wertebereichs gewählt werden (z.B. zwischen -1 und 1), wird das Ergebnis ein rein zufälliger Wert sein. Die derzeit am meisten genutzte Variante ein Netz lernen zu lassen, ist das sogenannten *Supervised Learning*, womit das Lernen anhand von Beispielen gemeint ist.

Ein Beispiel bedeutet in diesem Fall eine Kombination von echten Input-Output Datenpaaren. Diese Beispiele werden im Rahmen des Trainings von künstlichen neuronalen Netzen verwendet, um alle Gewichte und *Bias Terms* optimal einzustellen, sodass das Netz am Ende des Trainings für alle Input Daten und auch für bisher noch nicht gesehene neue Input Daten das korrekte Ergebnis berechnen kann.

Und das funktioniert so:

Für einen Satz von Input-Daten (auch *Features* genannt) errechnet das noch untrainierte neuronale Netz jeweils ein Ergebnis. Dieses Ergebnis wird dann mit den bekannten Ergebnissen des Beispiel-Datensatzes (auch Targets oder Label genannt) verglichen, und es wird die Größe der Abweichung bzw. des Fehlers berechnet. Um sowohl positive als auch negative Abweichungen gleichzeitig abbilden zu können, wird zum Beispiel der Mittelwert der quadratischen Abweichung (Squared Mean Error SME) oder eine andere Fehlerfunktion verwendet.

[KI Basics](#)

[KI Events](#)

[KI News](#)

Der Backward Pass



Dann beginnt das eigentliche „Lernen“: Der gemessene Fehler wird rückwärts zurück in das künstliche Neuronale Netz geleitet (der sog. *Backward Pass* oder *Backward Propagation*), und jedes Gewicht und jeder Bias Term wird ein kleines Stückchen in die Richtung angepasst, die den Fehler kleiner macht. Die Größe dieser Anpassung wird erstens über den Anteil, den ein bestimmtes Neuronen-Gewicht am Ergebnis hatte (d.h. über sein aktuelles Gewicht) berechnet, und zum zweiten über die sogenannte Learning Rate, die zu den wichtigsten Einstellgrößen (*Hyperparameter*) von neuronalen Netzen gehört.

Gängige Learning Rates sind z.B. 0,001 oder 0,01, d.h. nur ein Hundertstel bis ein Tausendstel des errechneten Fehlers wird pro Durchlauf korrigiert. Ist die Anpassung pro Durchlauf zu groß, kann es dazu kommen, dass das Minimum der Fehlerkurve verfehlt wird und die Abweichungen immer größer statt kleiner werden (*Overshooting*). Manchmal wird die Learning Rate daher während des Trainings zunehmend verkleinert (*Learning Rate Decay*), um das Minimum der Fehlerfunktion besser zu bestimmen.

Eine andere mögliche Problematik sind Fehlerfunktionen mit lokalen Minima, in denen das neuronale Netz „hängen“ bleibt und daher das eigentliche Minimum nicht findet. Die Richtung der Korrektur wird durch die Ableitung der jeweiligen Funktionen errechnet, deren negativer Wert die Richtung vorgibt, in welcher die Fehlerfunktion minimiert wird. Die Minimierung der gewählten Fehlerfunktion ist das Ziel des Trainings bzw. des Lernens.

Epochen und Batches



Nachdem die Gewichte angepasst sind, erfolgt ein weiterer Durchlauf aller Input-Daten und die erneute Messung des Fehlers sowie die Backward Propagation dieses Fehlers zur erneuten Anpassung der Gewichte. Ein kompletter Durchlauf aller Input-Daten wird dabei jeweils als *Epoche* bezeichnet. Die Anzahl der Trainings-Epochen ist ebenfalls ein wichtiger Hyperparameter für das Training von neuronalen Netzen. Dabei können die Input-Daten je nach Größe des Datensatzes auch in gleich große Gruppen (*Batches*) eingeteilt werden, und das Training kann jeweils pro Batch durchgeführt werden.

Dies kann z.B. sinnvoll sein, um ein künstliches neuronales Netz schneller lernen zu lassen, oder um Begrenzungen der Rechenkapazität des ausführenden Computers Rechnung einzuhalten. Wichtig ist bei der Aufteilung in Batches eine Normalverteilung der Werte innerhalb jedes Batches im Vergleich zum gesamten Datensatz. Wenn alle Batches das neuronale Netz ein Mal durchlaufen haben, ist eine Epoche vollendet.

Convergence, Overfitting und Hyperparameter-Tuning

Je mehr Beispiele ein künstliches neuronales Netz für das Training bekommt und je öfter es diese gesehen hat, desto kleiner wird der Fehler bei den Ergebnissen. Das Hinlaufen und Anlehnen der Fehlerkurve an die 100% Linie

wird dabei als *Convergence* bezeichnet, und während des Trainings wird der Verlauf der Fehlerkurve beobachtet, um ggfs. das Training zu stoppen und Anpassungen an den Hyperparametern vornehmen zu können. Doch nicht immer bedeutet ein kleiner Fehler auch eine gute allgemeine Performance des neuronalen Netzes.

Denn wenn es während des Trainings alle bekannten Daten sehr oft gesehen hat, kann es dazu kommen, dass das künstliche neuronale Netz diese Daten eher auswendig lernt, statt ein abstraktes Konzept zu lernen. Dieses Problem wird auch als *Overfitting* bezeichnet. Da neuronale Netze auch hochkomplexe Funktionen abbilden können, besteht die Gefahr, dass sie irgendwann die perfekte Funktion für jeden bekannten Datenpunkt gefunden haben, diese Funktion aber für neue Daten nicht gut funktioniert.

Um sicherzustellen, dass ein neuronales Netz von bekannten Beispieldaten abstrahieren und auch korrekte Ergebnisse für bisher nicht gelernte Input-Daten liefern kann, werden die Beispieldaten vor dem Training unterteilt in Trainingsdaten, Testdaten und Blind-Testdaten, z.B. im Verhältnis 70 / 20 / 10.

Während des Trainings werden dann nur die Trainingsdaten verwendet, und die Fehlerquote (*Error Rate*) wird jeweils sowohl für die Trainingsdaten, als auch für die Test-Daten gemessen. Der gemessene Fehler der Test-Daten wird jedoch nicht in das künstliche neuronale Netz zurück gespeist. Dann wird das neuronale Netz durch Anpassungen aller Variablen so verbessert, dass es die maximale Performance in Bezug auf Trainings- und Test-Daten erreicht (*Hyperparameter Tuning*). Dieses „tunen“ von neuronalen Netzen gehört zu den Kerntätigkeiten von Ingenieuren für künstliche Intelligenz.

Erst wenn das Netz vermeintlich vollständig trainiert ist, kommen die Blind-Test-Einsatz. Wenn das künstliche neuronale Netz auch im *Blind-Test* scheitert, ist die Wahrscheinlichkeit hoch, dass es ein abstraktes Konzept gut gelernt hat.

Tote Neuronen und Dropout Layer

Eine weitere Methode zur Vermeidung des Overfitting, d.h. des Auswendiglernens von Daten, sowie auch zur Vermeidung von inaktiven oder „toten“ Neuronen, deren Gewichte im Training dauerhaft auf Null stehenbleiben, sind die sogenannten *Dropout Layer*.

Das Konzept von Dropout Layern ist auf den ersten Blick radikal: Während des Trainings werden 50% oder mehr Neuronen eines Layers einfach abgeschaltet. Die Abschaltung erfolgt jeweils pro Durchlauf auf zufälliger Basis. Alle Neuronen des Layers werden dadurch gezwungen, weniger spezielle und mehr abstrakte Konzepte zu lernen, um trotz der reduzierten Anzahl gut zu performen.

Nach Abschluss des Trainings, wird der Dropout Mechanismus für den laufenden Betrieb jedoch deaktiviert, d.h. dann stehen alle per Dropout trainierten Neuronen für die Berechnung zur Verfügung. Die Ergebnisse werden dadurch in der Regel deutlich besser und das neuronale Netz ist hinterher oft robuster im Hinblick auf neue unbekannte Daten.

Trainings-Ergebnisse in Checkpoints speichern und Fine-Tuning

Wenn das künstliche neuronale Netz fertig trainiert ist, werden alle Gewichte und Bias Terms als sogenannter Checkpoint gespeichert. Das neuronale Netz kann dann jederzeit mit diesen optimierten Gewichten und Bias Werten erneut gestartet werden. Während das Training oft sehr rechenintensiv ist und insbesondere bei Bildern als Input Daten meist eine GPU oder einen GPU Cluster erfordert, ist der eigentliche Betrieb eines trainierten neuronalen Netzes deutlich schlanker und schneller und kann z.B. auch auf Mobilgeräten oder normalen Laptops/PCs nahezu in Echtzeit erfolgen.

Ein fertig trainiertes künstliches neuronales Netz kann mit Hilfe des *Checkpoints* auch jederzeit mit neuen Daten nachtrainiert werden. Dazu werden initial die bestehenden Werte aus dem Checkpoint in das Netz geladen, und die neuen Daten zum Training verwendet. Es ist auch möglich, ein vortrainiertes neuronales Netz als Basis für das Training mit eigenen Daten zu verwenden, indem z.B. der letzte Layer eines mit sehr vielen Bildern trainierten Netzes durch einen eigenen neuen Layer ersetzt wird, etwa zur Klassifizierung von eigenen Objekten.

Bei diesem sogenannten *Fine-Tuning* von künstlichen neuronalen Netzen kann so auf bereits gelernte allgemeine Strukturen zurückgegriffen werden und das Netz muss nur noch die neuen Klassen lernen. Dies ist insbesondere bei der sehr rechenintensiven Verarbeitung von Bildern von großem Vorteil, aber auch bei der Verwendung von Sprache und Texten. Für das Fine-Tuning

wird deutlich weniger Zeit und Rechenleistung benötigt, als normalerweise für das Training von Anfang an notwendig wäre. Entsprechend vortrainierte künstliche neuronale Netze werden z.B. von Google angeboten, und auch nahezu alle individuell anpassbaren API Services von IBM (<https://jaai.de/ibm-watson/>), Microsoft (<https://jaai.de/microsoft/>), Amazon (<https://jaai.de/amazon/>), Google (<https://jaai.de/google/>) und Co basieren auf diesem Verfahren.

Trainings-Daten für das Supervised Learning

Für das beschriebene Supervised Learning wird entsprechend eine große Menge an Beispieldaten benötigt. Eine große Menge bedeutet hier z.B. eine Million Beispiele. Zwar können künstliche neuronale Netze teilweise auch mit kleineren Datensätzen schon beachtliche Ergebnisse erzielen, aber je mehr Daten zur Verfügung stehen, desto besser. Für die Klassifizierung von Bildern werden z.B. ab ca. 1.000 Beispielbildern pro Klasse brauchbare Ergebnisse erzielt. Eine ganze Forschungsrichtung der künstlichen Intelligenz beschäftigt sich auch mit Verfahren für das sogenannte *One-Shot-Learning*, d.h. das Lernen anhand von sehr wenigen oder nur einem Beispiel.

Das Supervised Learning (*überwachtes lernen*) selber kann noch weiter unterteilt werden in verschiedene Methoden der Datenverwendung und -Weitergabe innerhalb von künstlichen neuronalen Netzen: Bei sogenannten *Recurring Neural Networks* fließt z.B. das Ergebnis der vorherigen Input-


Daten in die Berechnung des aktuellen Outputs ein, so dass z.B. Zeitreihen analysiert und verarbeitet werden können, so z.B. auch bei *Long-Short-Term Memory Netzen (LSTM)* und *Sequence-to-Sequence Netzen*, die z.B. für die Spracherkennung und für die Übersetzung von Texten verwendet werden. Für die Bildverarbeitung werden sogenannte *Convolutional Neural Networks (CNN)* verwendet, welche Bilder mit einem Raster abtasten und von tieferen Ebenen (Punkten, Linien etc.) in jedem Layer weiter abstrahieren zu höheren Konzepten (ein Gesicht, ein Haus etc.).

Unsupervised und Reinforcement Learning

Weitere Verfahren, wie künstliche neuronale Netze lernen können, sind das *Unsupervised Learning*, (unüberwachtes lernen) bei dem Systeme nur Input-Daten erhalten und selber versuchen, diese sinnvoll zu klassifizieren, sowie das *Reinforcement Learning*, bei dem ein neuronales Netz selber die Input-Daten steuern kann (z.B. die Tasten eines Gaming Controllers) und dynamische Output-Daten zurückerhält zusammen mit einer Aufgabe bezüglich dieser Output-Daten (z.B. einen Punktestand zu maximieren).

Ausblick

Zusammenfassend läßt sich sagen, dass das beschriebene Lernen anhand von Beispielen (Supervised Learning) in Kombination mit tiefen künstlichen

neuronalen Netzen (Deep Learning) aktuell die Basis für den Großteil der pro  esetzen Applikationen künstlicher Intelligenz ist.

Um die teilweise mangelnde Verfügbarkeit von Beispieldaten zu beheben, bieten große Unternehmen teilweise Dienste kostenlos oder sehr günstig an, nur um an die entsprechenden Nutzungsdaten der Dienste zu gelangen. Ein anderer aktueller Trend ist die Erzeugung von synthetischen Daten und die Verwendung von Simulationen im Training: So können z.B. Roboter-Arme in einer virtuellen Umgebung trainiert werden, wodurch sich der Zeitaufwand und die Kosten für das Training drastisch verkürzen. In der realen Welt können die so trainierten Systeme dann ihre Erfahrung aus der künstlichen Welt einsetzen und echte Roboter-Arme zielgenau bewegen.

27. SEPTEMBER 2018 / VON ROLAND BECKER

Trackbacks & Pingbacks

1. Word Embeddings – Methoden zur Repräsentation von Wörtern in

Algorithmen und neuronalen Netzen – JAAI.de

24. Mai 2018 um 23:26 Uhr

[...] auftauchen, auch einen ähnlichen Vektor haben. Solche Algorithmen verwenden das sogenannte Unsupervised Learning, d.h. sie suchen ohne externe Vorgaben selbständig das Repräsentations-Modell, welches die [...]

2. Recap: Hamburg AI Gathering #4 – JAAI.de

24. Mai 2018 um 21:04 Uhr

[...] DarwinAI – Automatische Generierung hoch optimierter neuronaler
Netze und erklärbares Deep Learning [...]

3. Transfer Learning – So können neuronale Netze voneinander lernen –

JAAI.de

5. März 2018 um 18:43 Uhr

[...] und zeitaufwändig. Insbesondere im Bereich der Erkennung von
Objekten in Bildern mit Deep Learning wird eine enorme Rechenleistung
benötigt, um das Netz z.B. mit Millionen von Bildern jeweils für [...]

4. Genetische Algorithmen und Deep Learning – Die Evolution von neuronalen Netzen mit evolutionären Algorithmen – JAAI.de

12. Februar 2018 um 8:46 Uhr

[...] bestimmte Arten von Fragen zu finden. Insbesondere im sich sehr
rasant entwickelnden Bereich des Deep Learnings kann es dabei dazu
kommen, dass die verwendeten Optimierungs-Algorithmen beim Versuch,
die [...]

5. Längerfristige Hausaufgabe – S4 – Neuronale Netzwerke – Begleitender Informatikblog – Max von Stein

8. Februar 2018 um 10:57 Uhr

[...] <https://jaai.de/machine-deep-learning-529/> [...]

6. Convolutional Neural Networks – Aufbau, Funktion und Anwendungsgebiete – JAAI.de

6. Februar 2018 um 7:16 Uhr

[...] Convolutional Neural Network (kurz „CNN“) ist eine Deep Learning
Architektur, die speziell für das Verarbeiten von Bildern entwickelt wurde.
In der JAAI hat sich [...]

7. Recap: Hamburg AI Meetup #3 – JAAI.de

22. Januar 2018 um 10:03 Uhr

[...] D. Lawrence, Machine Learning Professor an der University of Sheffield,
entwickelte mit den Data Readiness Levels einen Ansatz, [...]

8. Adobe Photoshop stellt Auswahl-Werkzeug mit künstlicher Intelligenz vor –
JAAI.de

29. November 2017 um 14:30 Uhr

[...] im überwiegenden Teil der Fälle eine manuelle Korrektur der Auswahl
unerlässlich. Mit dem auf Machine Learning basierenden Feature soll sich
das nun [...]

9. Künstliche Intelligenz generiert hochauflösende Bilder von Menschen, die
nicht existieren – JAAI.de

2. November 2017 um 10:34 Uhr

[...] beiden künstlichen neuronalen Netze trainieren sich also gegenseitig,
die zugrundeliegende Architektur wird GAN (Generative Adversarial
Network) [...]

10. Künstliche Intelligenz erkennt Vorläufer von Darmkrebs innerhalb einer
Sekunde – JAAI.de

1. November 2017 um 10:54 Uhr

[...] Trainingsdaten dienten den Wissenschaftlern mehr als 30.000
endoskopische Bildgebungen von Krebszellen und deren [...]

11. Conversational Interface mit Deep Learning – So funktionieren Chatbots



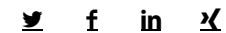
Dialogbaum – JAAI.de

2. um 15:30 Uhr

[...] einen neuen Ansatz beim Design von Conversational Interfaces
möglich: Durch die Verwendung von Deep Learning anstatt fester [...]

Kommentare sind deaktiviert.

Entdecke die Welt der künstlichen Intelligenz in Bremen.



Diese Website verwendet Cookies. Mit der weiteren Nutzung der Seite stimmst du der Verwendung zu. In den Datenschutz-Einstellungen kannst du festlegen, welche Cookies du zulassen möchtest und erhältst weitere Informationen.

COOKIES AKZEPTIEREN **EINSTELLUNGEN ÖFFNEN**

Cookie und Privatsphäre-Einstellungen

Wie wir Cookies verwenden

Wir können verlangen, dass Cookies auf Ihrem Gerät gesetzt werden. Wir verwenden Cookies, um zu messen, wenn Sie unsere Websites besuchen, wie Sie mit uns interagieren und um Ihre Nutzererfahrung zu verbessern.

Klicken Sie auf die verschiedenen Rubriken, um mehr zu erfahren. Sie können auch einige Ihrer Einstellungen ändern. Beachten Sie, dass das Blockieren eines Teils von Cookies die Funktionalität unserer Website und der verwendeten Dienste beeinträchtigen kann.



Essentielle Cookies

Diese Art von Cookies sind unbedingt erforderlich, um Ihnen die über unsere Website verfügbaren Dienste zur Verfügung zu stellen und einige ihrer Funktionen zu nutzen.

Da diese Cookies unbedingt notwendig sind, um die Website korrekt auszuliefern, können Sie sie nicht ablehnen, ohne die Funktionsweise unserer Website zu beeinträchtigen. Möchten sie diese Cookies trotzdem sperren oder löschen, so müssen Sie dies in Ihren Browsereinstellungen ändern und die Sperrung aller Cookies auf dieser Website erzwingen.

Google Analytics Cookies

Diese Cookies sammeln Informationen, die entweder in aggregierter Form verwendet werden, um zu verstehen, wie unsere Website genutzt wird, wie effektiv unsere Marketingkampagnen sind, oder um uns zu helfen, unsere Website und Anwendung für Sie anzupassen, um Ihr Nutzererlebnis zu verbessern.

Wenn Sie nicht möchten, dass wir Ihren Besuch auf unserer Website verfolgen, können Sie das Tracking in Ihrem Browser hier deaktivieren:

☐

Click to enable/disable google analytics tracking.

Weitere externe Dienste

Wir nutzen auch verschiedene externe Dienste wie Google Webfonts, Google Maps und externe Videoanbieter. Da diese Anbieter möglicherweise personenbezogene Daten wie Ihre IP-Adresse sammeln, erlauben wir Ihnen, diese hier zu sperren.

Bitte beachten Sie, dass dies die Funktionalität und das Erscheinungsbild unserer Website stark beeinträchtigen kann. Änderungen werden wirksam, sobald Sie die Seite neu laden.

☐

Click to enable/disable google webfonts.

☐

Click to enable/disable google maps.

☐

Click to enable/disable video embeds.



Datenschutzerklärung

Weitere Informationen zum Datenschutz und zu Cookies erhalten Sie in der Datenschutzerklärung:

[Datenschutzerklärung](#)