Ostfalia
Hochschule für angewandte Wissenschaften
Wolfenbüttel

Fakultät Informatik

Marco Philipp, 70459905
Niklas Röske, 70456600

Python Machine Learning: TensorFlow

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere, dass ich alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, und dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist.

Wolfenbüttel, den 21. Januar 2020

Kurzfassung

Hier sollte eine halbseitige Kurzfassung der Arbeit stehen.

Abstract

Here, an abstract written in English should appear.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

1. Einleitung

MARCO PHILIPP

2. Maschinelles Lernen

MARCO PHILIPP

2.1. Definition und Ziel

Machine Learning ist ein Teilgebiet der Informatik, welches sich mit dem Entwickeln von Maschinen beschäftigt, die Aufgaben erledigen, für die sie nicht explizit programmiert wurden. Mithilfe von Machine Learning sollen Aufgaben gelöst werden, welche mithilfe einfacher Algorithmen nicht lösbar sind. Dies ist zum Beispiel der Fall, wenn viele verschiedene Eingaben möglich sind, welche alle ein bestimmtes Muster besitzen und nach diesem Muster klassifiziert werden sollen. Bei einer Texterkennung wird ein Bild des Textes genommen und in Pixel umgewandelt. Schon bei einer Bildgröße von 8x8 Pixeln entstehen dadurch $2^{64} \approx 18{,}4$ Trillionen Varianten, wenn jedem Pixel der Wert null oder eins für vorhanden oder nicht vorhanden zugeordnet wird. Mit herkömmlichen Algorithmen müsste für jede dieser Varianten ein Zeichen hinterlegt werden, welches diese Darstellung widerspiegelt. Für andere Beispiele wie Spracherkennung wären die möglichen Variationen noch um einiges größer. Da geschriebene Zeichen jedoch ein recht simples Muster aufweisen, sodass sie erkannt werden können, ist es möglich, dieses Muster auch einem Computer beizubringen. Damit ist ein Ziel von Machine Learning die Mustererkennung in Daten. Aufgrund der Muster können verschiedene weitere Ziele verfolgt werden. Diese sind das Vorhersagen von Ereignissen, wie z.B. Aktienkurse, die automatisierte Steuerung von Maschinen, z.B. autonom fahrende Autos, oder Sprachassistenten, z.B. Cortana oder Amazon Alexa.

2.2. Aufbau und Funktionsweise

In diesem Kapitel werden verschiedene Strategien und Ansätze von Machine Learning vorgestellt. Generell wird beim Machine Learning ein Modell mithilfe von Trainingsdaten trainiert. Diese Trainingsdaten können in verschiedenen Formen vorliegen, z.B. Bilder oder Datensätze mit Informationen zu Personen oder Gegenständen. Bei einigen Varianten werden Trainingsdaten auch erst während des Trainings erzeugt. Das Ergebnis eines solchen Prozesses ist ein statistisches Modell, mit dem für neue Eingaben dem Training entsprechende Ausgaben getätigt werden.

2.2.1. Überwachtes Lernen

Überwachtes Lernen wird eingesetzt, wenn eine Menge an Trainingsdaten sowie ein festes Ziel bzw. eine richtige Antwort vorliegen. Beispiele dafür sind Klassifikationsprobleme und Regressionsprobleme. Bei Klassifikationsproblemen sollen Objekte in Klassen eingeordnet werden. Dabei müssen bereits eingeordnete Objekte vorliegen, sodass die Klassifikation daraufhin trainiert werden kann. Neue Objekte können dann einer der bekannten Klassen zugeordnet werden. Die richtige Lösung des Problems sind in dem Fall die bereits eingeordneten Objekte. Bei Regressionsproblemen soll Anhand von vorliegenden Daten eine Vorhersage für neue Daten getätigt werden. Zum Beispiel die Vorhersage von Aktienkursen aufgrund von Kursdaten aus der Vergangenheit. Dabei soll der Zusammenhang zwischen den Daten und dem daraus resultierenden Ergebnis hergestellt werden.

2.2.2. Unüberwachtes Lernen

Unüberwachtes Lernen erfordert im Gegensatz zu überwachtem Lernen keine richtige Antwort für eine Problemstellung oder ein bestimmtes Ziel welches erreicht werden soll. Es werden hauptsächlich Muster in den gegebenen Daten gesucht, welche für den Anwender interessant sein könnten. Eine oft genutzte Anwendung ist das Clustering. Beim Clustering werden Daten in verschiedene Cluster (engl. Haufen) eingeteilt, wenn sie ähnliche Eigenschaften besitzen. Dies ähnelt dem Klassifikationsproblem des überwachten Lernens, jedoch sind hier

weder Anzahl noch Art der verschiedenen Klassen bekannt, sondern werden während des Clusterings selbst gefunden. Das unüberwachte Lernen findet hauptsächlich im Data-Mining Anwendung.

2.2.3. Bestärkendes Lernen

Bestärkdenes Lernen kann eingesetzt werden, wenn ein Problem vorliegt, welches einige Eigenschaften besitzt. Die Systeme, auf die sich das Problem bezieht, können sich in bestimmten Zuständen befinden. Weiterhin gibt es meist einen günstigen Zustand, welcher erreicht werden soll. Um diesen günstigen Zustand zu erreichen, muss regelmäßig eine Aktion aus einer Menge von Aktionen gewählt werden. Das Ziel dieser Variante ist das Entwickeln einer möglichst guten Strategie, sodass der günstige Zustand erreicht wird. Ein Beispiel, wo so ein System vorliegt, ist das Spiel TicTacToe. Das Spiel selbst befindet sich immer in einem bestimmtem Zustand, nämlich dem Zustand des Spielfeldes. Dieses besteht aus neun Feldern und hat entweder ein Kreuz, einen Kreis oder nichts darauf. Diese Werte können in 1, 0 oder -1 umgewandelt werden und als Vektor zum darstellen des Zustands genutzt werden. Der zu erreichende günstige Zustand ist das Gewinnen des Spiels, welcher dadurch erreicht wird, das ein Spieler eine Reihe mit seinen Symbolen füllt. Die Aktionen, welche zur Verfüguing stehen sind das setzen des eigenen Sysmbols auf eines der noch freien Felder. Um einem Modell beizubringen, dass es bei diesem Spiel gewinnen soll und wie es gewinnt, wird eine sogenannte reward-Funktion definiert. Diese gibt dem Modell eine Belohnung, wenn es das gewünschte Verhalten zeigt und gibt eine negative Belohnung bzw. eine Bestrafung, wenn es ungewünschtes Verhalten zeigt. Dabei gilt, dass das Modell eine möglichst hohe Belohung anstrebt. Ein Vorteil des bestärkenden Lernens ist, dass keine Trainingsdaten erforderlich sind, sondern diese Daten während dem Trainieren des Netzes generiert werden. In dem Beispiel weden Daten durch das Spielen des Spiels generiert. Ein Nachteil ist, dass ein System mit spezifischen Anforderungen vorliegen muss, sodass diese Art des Lernens nur in bestimmten Fällen Anwendung finden kann.

2.2.4. Parametrisiertes Lernen

Beim parametrisierten Lernen wird eine Funktion für ein Problem entwickelt, welche die Eingabewerte in Ausgabewerte transformiert. Dabei hat diese Funktion fest vorgegebene Anzahl an Parameter, welche durch das trainieren des Modells herausgefunden werden. Dies kann z.B. für Regressionsprobleme verwendet werden, um eine Funktion für eine Menge von Punkten in einem Koordinatensystem zu ermitteln, die eine betimmte Form haben soll. Das parametrisierte Lernen eignet sich jedoch nur für simplere Zusammenhänge, da es durch die fest vorgegebenen Parameter stark eingeschränkt ist. Dafür ist das Trainieren des Modells und die spätere Anwendung schneller als beim nicht-parametrisierten Lernen, da keine neuen Parameter gefunden, sondern nur vorhandene ermittelt werden müssen.

2.2.5. Nicht-Parametrisiertes Lernen

Beim nicht-parametrisierten Lernen soll ebenfalls eine Funktion gefunden werden, welche Eingabewerte in Ausgabewerte transformiert. Jedoch wird für diese Funktion keine feste Anzahl an Parametern vorgegeben, sondern die Parameter werden durch das Trainieren des Modells selbst ermittelt. Dadurch können beim nicht-parametrisierten Lernen viel mehr Parameter auftreten als beim parametriesiertem Lernen, da theoretisch jeder Eintrag der Trainingsdaten ein neuen Parameter ergeben könnte. Nicht-parametrisiertes Lernen ist im Allgemeinen langsamer als parametrisiertes Lernen, da zusätzlich zu den Werten der Parameter noch die Parameter selbst ermittelt werden müssen. Dafür eignet es sich, um komplexere Problemstellungen zu lösen. Ein Beispiel für nicht-parametrisches Lernen ist das Clustering. Beim Clustering wird nicht vorgegeben, wie viele Cluster es gibt, sondern die Anzahl der Cluster wird während des Trainierens ermittelt, sodass keine feste Parametrisierung möglich ist.

2.3. Anwendungsbereiche

Machine Learning findet in vielen verschiedenen Bereichen Anwendung. Ein sehr großes Schwerpunkt befasst sich mit dem Steuern von Maschinen, explizit mit dem autonomen Steuern von Autos. Dabei werden verschiedene Sensoren eingesetzt, welche dem Auto umfangreiche Informationen der Umgebung geben. Dazu gehören Kameras, Radarsysteme oder Ultraschallsensoren. Diese Eingabewerte werden durch ein Modell in Steuerungsanweisungen umgewandelt, sodass das Auto von allein fährt. Dabei muss vor allem auf bewegliche Hindernisse reagiert in Echtzeit reagiert werden können, was eine schnelle Auswertung der Daten verlangt. Eine weitere Anwendung ist Data-Mining. Dabei werden viele Daten von z.B. Kunden auf Muster untersucht. Diese Muster werden dann genutzt, um Kunden weitere Angebote zu machen, die sie wahrscheinlich interessieren werden. Somit können höhere Absätze erzielt werden. Allerdings wird Data-Mining auch im Marketing genutzt, um zu analysieren, welche Zielgruppe angesprochen werden soll und was für Möglichkeiten es gibt, diese zu erreichen. Machine Learning wird auch für verschiedene Assistenzsysteme genutzt. Die bekanntesten sind Sprachassistenten. Diese nehmen das gesprochene des Nutzers auf und interpretieren es, sodass enstsprechende Anweisungen ausgeführt werden können, z.B. einen Wecker stellen oder einen Anruf starten. Machine Learning kann ebenfalls in der Medizin eingesetzt werden, um Hinweise auf Krankheiten früher erkennen zu können. Dies hilt, Krankehieten wie Krebs oder Herzstörungen eher zu diagnostizieren, sodass diese in einem frühem Stadium behandelt werden. In Email-Programmen werden Emails oft in verschiedene Unterordner einsortiert, z.B. Werbung oder Spam. Dies ist ein klassisches Klassifikationsproblem basierend auf Schlüsselwörtern in der Email.

3. TensorFlow

NIKLAS RÖSKE

Worum geht es in Ihrer Arbeit [1] Insgesamt 2 Seiten

3.1. Allgemeines

"Die Daten fließen als Tensoren durch einen Graphen von Rechenoperationen, aus denen unser Deep-Learning-Netz besteht."[1]

Berechnung als Datenflussgraph

Knoten Darstellung von Operationen

Kanten Daten (Tensoren)

Berechnungen mit Datenflussgraphen

für maschinelles Lernen entwickelt

Portierbarkeit (unterschiedliche Hardware)

Hauptkomponenten in C++

Python am weitesten entwickelt und am meisten genutzt

Flexibilität, Modelle sind einfach zu erstellen

viele Optimierungsverfahren, der Nutzer wird bei Verwendung dieser unterstützt

Prozess kann durch Tensorboard beobachtet werden

Abstraktionsbibliotheken, wie Keras vereinfachen Benutzung Modelle können schneller in den Betrieb gehen

Modell für einzelnen Prozessor kann durch wenige Änderungen auf Prozessorclustern laufen

Spezielle Tensor Processing Units von Google

Wird ständig verbessert in den Bereichen Benutzerfreundlichkeit, Leistungsfähigkeit und Nützlichkeit Hier wird auf die Textstelle 4.1 verwiesen, die sich auf der Seite 9 befindet.

3.2. Unterschiede zu anderen Frameworks

4. Arbeitsweise

4.1. Tensoren

NIKLAS RÖSKE

Insgesamt 1-2 Seiten

4.1.1. Mathematische Definition

Bild 5.9 Deep-Learning Seite 136 Ein Tensor ist eine Bezeichnung für ein n-dimensionales Feld. In der Abbildung sind unterschiedliche Dimensionen von Feldern dargestellt. "Beispielsweise ist ein 1-x-1-Tensor ein Skalar, ein 1-x-n-Tensor ein Vektor, ein n-x-n-Tensor eine Matrix, und ein n-x-n-x-n-Tensor ist einfach ein dreidimensionales Feld"[1].

4.1.2. Tensoren in TensorFlow

In TensorFlow hingegen werden die Daten, die durch den Graphen fließen Tensoren genannt. Da diese Daten wie die Tensoren in der Mathematik verschiedene Dimensionen annehmen können, wie zum Beispiel einfache Zahlenwerte oder dreidimensionale Felder eines Bildes, basiert der Begriff auf dem mathematischen Begriff.
Tensor-Objekte entstehen bei der Ausführung einer Operation.

4.2. Operationen

MARCO PHILIPP

Ziele der Arbeit

4.2.1. Typen

Was ist die Motivation hinter dieser Arbeit

4.2.2. Beispiele

Was ist die Motivation hinter dieser Arbeit

4.3. Graphen

NIKLAS RÖSKE

Insgesamt 3 Seiten

4.3.1. Aufbau

Wie schon in Kapitel 3.1 erwähnt wurde, arbeitet TensorFlow auf einem Datenflussgraphen. Dabei handelt es sich um ein gerichtetes Diagramm, das ein mathematisches Problem darstellt. Diese Datenflussgraphen bestehen aus Tensoren 4.1 und Operationen 4.2, die miteinander verbunden sind. Durch eine Aneinanderreihung von unterschiedlichen Operationen, mit den dazugehörigen Tensoren als Werte, entsteht ein Datenflussgraph. Dieser wird als mathematische Grundlage für das Neuronale Netz genutzt.

4.3.2. Funktionsweise

Hier Bild 5.11 aus dem Buch Deep-Learning einfügen.
Die Abbildung zeigt eine einfache Berechnung als Datenflussgraphen, um den Ablauf zu verdeutlichen. Die Berechnung eines Graphen kann nur in einem *tf.Session()*-Block ausgeführt werden. Am Anfang dieses Blocks wird zusätzlich eine *tf.Session()* Variable angelegt. Dies wird durch den Ausdruck *with tf.Session() as sess* realisiert. Das *Session()*-Objekt wird später benötigt, um diese zu starten. Bevor die *Session* gestartet wird, werden die Konstanten, oder auch Tensoren, a, b, c und d erstellt und ein Wert zugewiesen. Die letzte Operation wurde dem Tensor *result* in der Form *result = tf.sqrt(x3, "Wurzel")* zugewiesen. Für den Start muss

der Operation der Tensor übergeben werden, der berechnet werden soll. Diese Operation muss zusätzlich einer Variablen übergeben werden, damit der Wert weiter verwendet werden kann. In diesem Fall sieht die Funktion wie folgt aus: *res = sess.run(result)*. Über die Variable *res* kann nun das Ergebnis ausgegeben werden.

Hier Bild 3-4 aus Buch EInfuehrung einfügen Vor der Ausführung eines *Session*-Blocks enthalten die wird durch das Aufrufen einer Operation nur eine Instanz dieses Objekts erstellt. Dieses Objekt enthält nur Referenz der Daten und noch keine richtigen Daten. Dies ändert sich mit dem Start der Ausführung und dem Durchlaufen der Operationen. Wie in der Abbildung zu sehen ist werden die Referenzen der Tensor-Objekte in A durch Tensor-Objekt-Instanzen ersetzt.

4.3.3. Darstellung von Operationen

Im Kapitel 4.2 wurde im Allgemeinen auf die Operationen in TensorFlow eingegangen. Dies wird nun noch für die Darstellung vertieft.

4.3.4. Vor- und Nachteile von Graphen zur Berechnung

Durch die Verwendung von Graphen ist es möglich einzelne Operationen ohne große Umstände auszutauschen.

4.4. Trainieren des Netzes

NIKLAS RÖSKE

Insgesamt 1-2 Seiten

4.4.1. Eingaben

Für das Trainieren eines Neuronales Netzes werden, wie auch bei anderen Frameworks Trainingsdaten benötigt. Platzhalter werden für die Eingaben verwendet.

4.4.2. Ablauf

4.4.3. Ergebnis

5. Praktisches Beispiel

Worum geht es in Ihrer Arbeit

5.1. Zielsetzung

MARCO PHILIPP

5.2. Planung

NIKLAS RÖSKE

1 Seite

5.3. Datensätze

MARCO PHILIPP

5.4. Erstellen des Netzes

NIKLAS RÖSKE

1-2 Seiten

5.5. Ergebnisse auswerten

MARCO PHILIPP

5.6. Probleme

NIKLAS RÖSKE

1 Seite

6. Fazit

NIKLAS RÖSKE

Insgesamt 1-2 Seiten

6.1. Vor- und Nachteile von TensorFlow

6.2. Ausblick

6.2.1. TensorFlow Lite

Literaturverzeichnis

[1] I. L. Tom Hope, Yehezkel S. Resheff, *Einführung in TensorFlow Deep-Learning-Systeme programmieren, trainieren, skalieren und deployen*. O'REILLY, 2018.

A. Anhang A