Woche 11 Theorie: Optimierung & Wandern mit Gradientenabstieg

Skript

Erarbeitet von
Joana Grah

Lernziele 1
Inhalt 2
Einstieg 2
Verlustfunktion quantifiziert Optimalität 2
Die Optimierung – (Stochastischer) Gradientenabstieg 4
Take-Home Message 7
Quellen 8
Weiterführendes Material 8
Disclaimer 9

Lernziele

- Die Interpretation der Verlustfunktion erklären können, z.B. anhand des Klassifikationsproblems in Hunde- und Katzenbilder beim Überwachten Lernen
- Erklären können, aus welchen Komponenten der Gradient einer Funktion besteht
- Die Intuition hinter der Optimierungsmethode des (Stochastischen) Gradientenabstiegs beschreiben können
- Den Ablauf des Trainings grob erläutern können
- Beispiele für Stopp-Kriterien des Trainings geben können

Projekt „KI für alle"
HeiCAD – Heine Center for Artificial Intelligence and Data Science

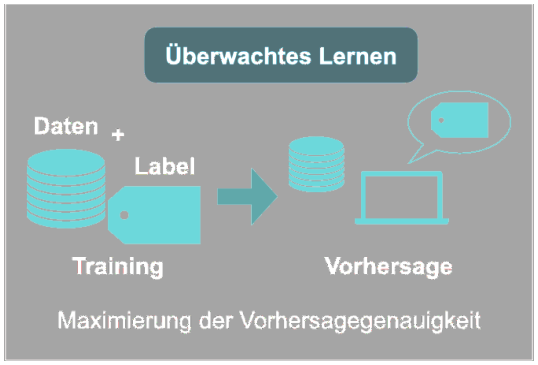
Inhalt

Einstieg

Wie trainieren wir jetzt eigentlich ein neuronales Netzwerk? Oder etwas mathematischer: Wie optimieren wir die Parameter?

Verlustfunktion quantifiziert Optimalität

Einblendung Grafik Überwachtes Lernen

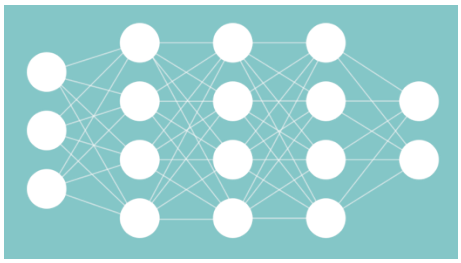


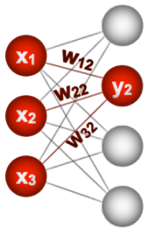
Wir möchten gerne ein neuronales Netzwerk mit einem überwachten Deep Learning Algorithmus trainieren.

Einblendung Grafik Datensatz

Wir haben einen geeigneten Datensatz, den wir bereits vorverarbeitet und bereinigt haben. Anschließend wurde er in Trainings-, Validierungs- und Testdaten aufgeteilt.

Einblendung Grafiken Neuronales Netzwerk





$$a\left(w_{12}\cdot x_1 + w_{22}\cdot x_2 + w_{32}\cdot x_3 + b\right)$$

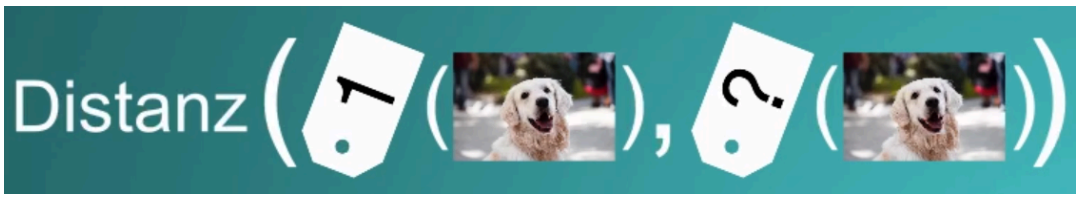
Wir kennen den Weg, den die Daten durch ein sequenzielles Modell als Input durch die Hidden Layers bis zum Output durchlaufen. Hier entscheiden jetzt Qualitäts- bzw. Fehlermaße darüber, wie gut der Output für ein bestimmtes zugrundeliegendes Problem ist.

Aber wie bekommen wir das Modell jetzt dazu, den Output zu verbessern? Wie definieren wir Optimalität? Natürlich müssen wir auch das wieder quantifizieren, und zwar durch die sogenannte „Loss Function“, also Verlustfunktion, oder manchmal auch „Cost Function“, Kostenfunktion, genannt.

Einblendungen Grafik Verlustfunktion

Uns interessiert, wie sehr sich der aktuelle Output des Netzwerkes von dem gewünschten Output unterscheidet. Im Supervised Learning Setting haben wir ja bereits gewünschte Input-Output-Paare in Form der Daten und dazugehörigen Labels. Jetzt wollen wir also unser aktuelles Ergebnis während des Trainings mit dem gewünschten Ergebnis vergleichen – und das können wir mathematisch übersetzen in: die Distanz oder Differenz berechnen.

Einblendungen Grafik Verlustfunktion - Beispiel



Zum Beispiel können wir schauen, ob im aktuellen Trainingsschritt ein Hundebild auch wirklich als Hund klassifiziert wird. Ist das der Fall, ist die Differenz zwischen dem vorhergesagten Label „1“ und dem tatsächlichen Label „1“ gleich null. Würde das Bild inkorrekterweise als „Katze“ erkannt, wäre die Differenz zwischen dem vorhergesagten Label „0“ und dem tatsächlichen Label „1“ 1.

Einblendungen Grafik Verlustfunktion

Machen wir das für jedes Bild aus dem Datensatz, bleibt die Verlustfunktion bei guter Performance, also bei vielen korrekt vorhergesagten Labels, nahe bei null. Machen wir andererseits viele Fehler bei der Vorhersage, summieren sich die Einsen auf und wir erhalten einen hohen Verlust. Das Ziel der Optimierung ist jetzt, die Verlustfunktion zu minimieren.

Die Optimierung – (Stochastischer) Gradientenabstieg

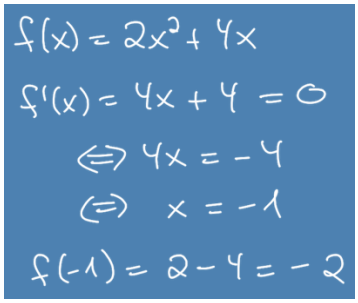
Wie geht das jetzt genau? Ganz genau werden wir das an dieser Stelle nicht erklären, aber zumindest versuchen, eine Intuition zu schaffen.

Einblendungen Grafik Verlustfunktion

Die Parameter, die letztendlich in neuronalen Netzwerken optimiert werden, sind die Gewichte oder Weights und die Biases. Ziel ist also, die Verlustfunktion bezüglich dieser Weights und Biases zu minimieren.

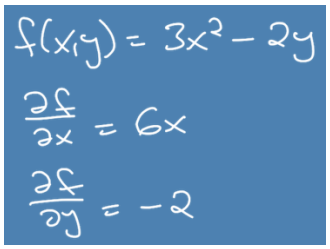
Die Methode, die dafür standardmäßig verwendet wird, ist der so genannte „Gradient Descent“ oder Gradientenabstieg. Vielleicht sollten wir uns kurz den Begriff „Gradient“ in Erinnerung rufen, den wir vielleicht noch aus der Schule kennen. Falls ihr euch wie ich mit Begeisterung an die Kurvendiskussion erinnert, dann hatte Optimierung oder das Auffinden von Hoch- und Tiefpunkten einer Funktion immer irgendwie mit Ableitungen zu tun. Wollten wir den Tiefpunkt oder das Minimum einer Funktion finden, haben wir die Ableitung gleich null gesetzt und dann noch geprüft, dass ein Tiefpunkt vorliegt… Was bedeutet nochmal Ableitung?

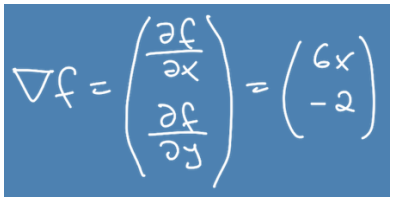
Einblendung handschriftliche Berechnungen, animiert



Nehmen wir die Funktion $f(x) = 2x^2 + 4x$. Die Ableitung davon ist $4x + 4$. Setzen wir das gleich null, erhalten wir $x = -1$. Die Funktion f ist also an der Stelle $x = -1$ minimal und nimmt den Wert -2 an.

Einblendung handschriftliche Berechnungen, animiert





Es gibt aber auch Funktionen mit mehr als einer Variablen. Nehmen wir $f(x, y) = 3x^2 - 2y$. Jetzt können wir zwei Ableitungen berechnen, einmal die bezüglich x und einmal die bezüglich y. Sie heißen auch partielle Ableitungen. Der Gradient von f wird dann geschrieben als Vektor mit zwei Komponenten: der partiellen Ableitung bezüglich x und der partiellen Ableitung bezüglich y. Konkret wäre der erste Eintrag dann $6x$ und der zweite Eintrag -2.

Jetzt sind wir gut vorbereitet, um uns die Methode des Gradientenabstiegs mal ganz vereinfacht und veranschaulicht vorzustellen.

Einblendung Bild Berg-Tal

Nehmen wir an, wir befinden uns in dieser idyllischen Landschaft und wollen von einem Berggipfel bis ganz nach unten ins Tal wandern. Hier entspricht jetzt die Höhe, auf der wir uns befinden, dem Wert der Verlustfunktion, d. h. ganz oben auf dem Berg ist sie sehr hoch und wir wollen den Punkt im Tal finden, der einer sehr niedrigen Zahl oder sogar null entspricht. Wie kommen wir dort hin, wenn wir irgendwo auf einer Bergspitze starten? Wir können natürlich nicht fliegen oder uns in einem Schritt ins Tal beamen. Vielmehr müssen wir lokal schauen, welcher nächste Schritt uns möglichst schnell und effizient nach unten bringt. Intuitiv würden wir dann einen Schritt in Richtung des steilsten Abstiegs gehen – solange wir uns dabei nicht verletzen und das mit einer Schrittlänge machbar ist. So gelangen wir nach und nach oder Schritt für Schritt weiter nach unten, checken immer wieder lokal, was für uns gerade die beste Richtung ist, und kommen schließlich im Tal an. Dabei kann es sein, dass wir nicht *die* tiefste Stelle des gesamten Landschaftszugs gefunden haben, aber die für uns lokal tiefste Stelle. Das ist in den meisten Fällen bei neuronalen Netzwerken auch ausreichend, und man spricht dann von einem „lokalen“ statt „globalen“ Minimum, das für die Verlustfunktion gefunden wurde.

Einblendung Animation Gradientenabstieg

Hier sehen wir eine Animation, die den Gradientenabstieg noch einmal veranschaulicht. Stellen wir uns vor, dass wir unsere Landschaft nun von oben betrachten. Wir sehen Höhenlinien und die Farbe Gelb entspricht einem hohen Wert der Verlustfunktion oder Höhe und Dunkelblau einem niedrigen Wert. Die roten Punkte sind Menschen, die Richtung Tal wandern möchten, und alle von verschiedenen Positionen aus starten. Das nennt man bei neuronalen Netzwerken auch Initialisierung. Wir müssen ja am Anfang den Weights und Biases bestimmte Werte zuordnen, die dann im Laufe der Zeit optimiert werden.
Hier sehen wir schön, wie alle nach und nach ein lokales Minimum erreichen.

Die Schrittweite während des Wanderns wird in Analogie beim Gradientenabstiegsverfahren im Deep Learning auch als „Learning Rate“ bezeichnet. Sie kann fix gewählt werden oder sich im Laufe der Optimierung verändern.

In der Praxis bestehen neuronale Netzwerke manchmal aus Millionen von Parametern, also den Weights und Biases, d. h. die Verlustfunktion muss nicht bezüglich einer oder zweier Variablen, sondern manchmal Millionen davon abgeleitet werden. Wir müssen also einen riesigen Gradienten ausrechnen. Die Methode, die mathematisch dafür verwendet wird, heißt Backpropagation und ergibt sich aus der ebenfalls aus der Schule bekannten Kettenregel. Wir werden an dieser Stelle aber nicht näher auf die Details eingehen.

In der Praxis wird zur Optimierung von neuronalen Netzwerken eine bestimmte Methode des Gradientenabstiegs verwendet, nämlich der „Stochastic Gradient Descent“ bzw. der stochastische Gradientenabstieg. Anstatt den Gradienten für den gesamten Datensatz zu berechnen, wird er für eine zufällige Teilmenge davon berechnet und liefert trotzdem eine gute Approximation des tatsächlichen Gradienten. Das beschleunigt die Optimierung natürlich auch erheblich.

Das Training – oder die Parameteroptimierung – eines neuronalen Netzwerkes ist ein iterativer Prozess, d. h. Schritt für Schritt nähern wir uns dem Optimum.

Einblendung Grafik Datensatz

Wir beginnen mit dem Vorbereiten der Daten und dem Aufteilen in Trainings-, Validierungs- und Testdaten.

Einblendung Grafik Training

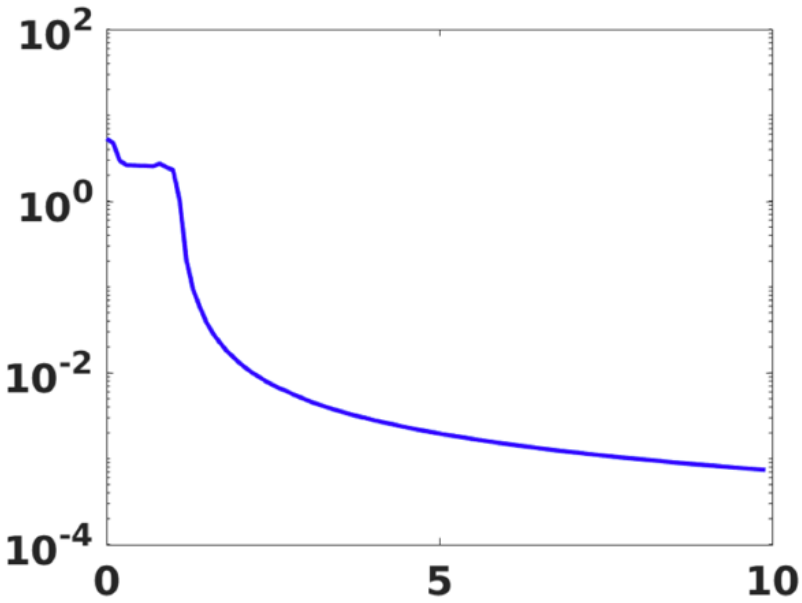
Dann durchlaufen wir immer jeweils eine Epoche. Epoche bedeutet, dass der gesamte Trainingsdatensatz genau einmal im Optimierungsprozess verwendet wird.
Pro Epoche können wir die Daten noch in sogenannte „Batches“, also Teilmengen des Datensatzes, aufteilen.
Dafür werden dann jeweils die Parameter optimiert.

Die Frage, die sich zum Schluss noch stellt, ist: Wann stoppen wir das Ganze?

Eine gute Strategie ist es, den Verlauf der Optimierung zu visualisieren.

Wir können z. B. die Anzahl an Epochen gegen den Wert der Verlustfunktion auftragen. Im besten Fall sieht das dann in etwa so aus wie hier:

Einblendung Grafik Epochen vs. Verlustfunktion



Quelle [1]

Am Anfang sinkt die Verlustfunktion ziemlich schnell und bei zunehmender Anzahl der Epochen sinkt sie langsamer und nähert sich einem Wert an. Also könnte ein Kriterium die Optimierung zu stoppen lauten: Wenn die Verlustfunktion sich zwischen zwei Epochen nicht mehr stark ändert, stoppe die Optimierung. Und diesen Wert könnte man sich vorher überlegen.

Eine andere Möglichkeit, die man eigentlich parallel verfolgen sollte, ist, sich die Qualitätsmaße für die Validierungsdaten anzuschauen. Hier könnten wir die Anzahl der Epochen gegen die Accuracy bei Klassifikationsproblemen oder die Peak Signal-to-Noise Ratio beim Bildentrauschen auftragen. Im Idealfall steigen die Qualitätsmaße und ein Kriterium zum Stoppen könnte dann die Überschreitung eines im Vorhinein festgelegten Wertes sein.

Take-Home Message

Jetzt könnt ihr das Training bzw. die Parameteroptimierung eines neuronalen Netzwerkes anhand der folgenden Checkliste erklären.

Quellen

Quelle [1] Higham, C. F., & Higham, D. J. (2019). Deep learning: An introduction for applied mathematicians. *Siam review*, *61*(4), 860-891.

Weiterführendes Material

KI-Campus: Gradient Descent – Interaktive Browsersimulation. https://ki-campus.org/simulations/gradient-descent

Elements of AI: Building AI
Getting started with AI – Optimization.
https://buildingai.elementsofai.com/Getting-started-with-AI/optimization
Getting started with AI – Hill climbing.
https://buildingai.elementsofai.com/Getting-started-with-AI/hill-climbing

Plus Magazine. Maths in a minute: Gradient descent algorithms.
https://plus.maths.org/content/maths-minute-gradient-descent-algorithms

Disclaimer

Transkript zu dem Video „Woche 11 Theorie: Optimierung & Wandern mit Gradientenabstieg“, Joana Grah.