Algorithmic Bias and Fairness

SS 22

Teilnehmer: Tom Sühr, Josef Pelz

Motivation

Algorithmen finden längst in nahezu jedem Lebensbereich Anwendung. Insbesondere werden sie zur Entscheidungsfindung und in Empfehlungssystemen eingesetzt. Häufig werden dabei künstliche neuronale Netzwerke (NN) verwendet. Diese werden beispielsweise auf existierenden Datensätzen trainiert und angewendet um neue Datenpunkte gewissen Klassen zuzuordnen (Klassifizierung). Hierbei hat die Wahl der Datenbank, etwa die verwendeten Eigenschaften (Features) einzelner Datenpunkte, und der Aufbau/ die Architektur des NN großen Einfluss auf die Ergebnisse. Es ist daher wichtig, dass sowohl die Entwickler:innen als auch die Nutzer:innen dieser Systeme sich die Frage stellen: „Wie Gerecht sind diese Algorithmen?“.

Zielsetzung

Ziel des Projekts „Algorithmic Bias and Fairness“ (Algorithmische Vorurteile und Gerechtigkeit) ist es, die Relevanz und Komplexität dieser Frage greifbar zu machen. Zukünftige Entscheidungsträger:innen sollen einen kritischen Blick auf die Anwendung schwer durchschaubaren neuronalen Netzwerke erlangen.

Umsetzung

Um möglichst vielen Menschen die Auseinandersetzung mit der Thematik zu ermöglichen, haben wir uns für eine Website als Medium entschieden. Die potentielle Reichweite ist groß und die Verknüpfung mit anderen Webseiten via Hyperlinks ermöglicht es Nutzer:innen, je nach Vorwissen und Interesse, auf grundlegenderes oder tiefergehendes Wissen zuzugreifen. Zusätzlich können wir mittels JavaScript und WebGL eine interaktive Erfahrung kreieren, die eine spielerische Auseinandersetzung möglich macht und ein intuitives Verständnis vermitteln kann.

Um den Zugang niedrigschwellig zu gestallten, schafft der Einstieg in die Webseite einen Überblick über die Thematik. Es werden grundlegende Begriffe wie Algorithmus, Ungerechtigkeit und Diskriminierung eingeführt. Mehrere Links bieten Zugang zu Hintergrundwissen an. Anschließend werden Kriterien für (Un)Gerechtigkeit von Algorithmen eingeführt und Ansätze präsentiert um diese zu einzuhalten. Beispiele von Anwendungsbereichen Neuronaler Netzwerke stellen deren Relevanz dar und verdeutlichen: Die von einem Algorithmus getroffenen Entscheidungen sollten so fair wie möglich sein.

Um die grundlegende Funktionsweise Neuronaler Netzwerke zu greifbar zu machen und zu verdeutlichen führen wir ein interaktives Beispiel heran: Auf einer zweidimensionalen Fläche sind Datenpunkte einer Datenbank abgebildet, angeordnet nach zwei Eigenschaften/Features. Jeder Datenpunkt ist entweder als Kreis (positiv) oder als Kreuz (negativ) markiert. Dies könnte beispielsweise die Kreditwürdigkeit darstellen. Nutzer:innen werden aufgefordert, eine Linie durch die

Datenmenge zu ziehen, die so gut wie möglich Kreuze von Kreisen trennt. Aufgrund der Datenlage ist dies nicht fehlerfrei möglich. Das ist zwar zu erwarten, die Frage ist jedoch auf Kosten vom wem die Fehler gemacht werden. Im nächsten Schritt wird ein weiteres, soweit unbekanntes Feature durch Farbgebung sichtbar. Datenpunkte gehören nicht nur zur Klasse Kreuz oder Kreis, sondern zusätzlich auch entweder zu Gelb oder Blau. Es wird ersichtlich, dass die meisten Fehler bei Datenpunkten der Gruppe Gelb gemacht wurden. Es wird dadurch klar, dass wir unabsichtlich und unwissend diskriminieren können.

Aufbau

Die Webseite basiert auf einem W3.css Template. Auf der scroll-baren Startseite sind Blöcke vertikal angeordnet, die portionsweise Informationen beinhalten. So wird beispielsweise zuallererst grundlegendes Wissen zu den Themen Algorithmen, Ungerechtigkeit und Diskriminierung angeboten. Nutzer:innen können, je nach Wissensstand, Links folgen um sich weiter in die Thematik einzulesen, oder dem linearen Aufbau der Seite folgen.

Der zentrale Teil der Seite ist ein WebGL Canvas. Dieser beinhaltet die oben beschriebene interaktive Komponente. Wir nutzen hierfür ausschließlich die JavaScript WebGL API, keine bestehenden Bibliotheken/Frameworks. Für jeden Datenpunkt wird ein Punkt gerendert, der einem Quadrat aus mehreren Pixeln entspricht. Informationen über dessen Position, Zugehörigkeit zu verschiedenen Klassen etc., werden in einem JavaScript Array festgelegt und mithilfe von einem gl.ARRAY_BUFFER an den Vertex-Shader und von diesem an den Fragment-Shader übergeben. Die Position und Größe des Datenpunktes werden im Vertex-Shader ausgelesen und angewendet, während der Fragement-Shader die entsprechenden Pixel in Form eines Kreises oder eines Kreuzes einfärbt.