Neuro-Fuzzy-Layers

October 17, 2019

1 Neuro-Fuzzy-Modell: Implementierung der ANFIS-Ansatz

In der folgenden Ausarbeitung stelle ich meine Aufsetzung des ANFIS-Ansatzes in Python vor. Infolge meiner Ausarbeitung habe ich mich mit der Bibliothek für Neuronale Netze Tensorflow zusammengesetzt. Tensorflow ist ein Produkt von Google und eins der verbreitendsten APIs für Erstellung von Neuronalen Netzen. Wegen zahlreicher Nutzung ist das Web befüllt mit Guides und Tutorials über das Bauen von Neuronalen Netzen mit der Bibliothek Tensorflow. Aus diesem Grund habe ich mich für ihre Verwendung entschieden. Eine weitere Funktionalität, die die Bibliothek anbietet, ist die einfache Migration von erstellten Neuronalen Netzen. Tensorflow bietet die Möglichkeit, einfach Modelle nach den unterstützten Programmiersprachen zu exportieren, in disem Sinne auch nach C++.

In diesem Artikel wird so vorgegangen, dass zu jedem Programmcode eine Erklärung gegeben wird. Vorrigen Artikeln gehen auf die eigentliche Struktur und theoretische Aufsetzung eines ANFIS-Models in einem Neuronalen Netz. Laut einem dieser Artikeln besteht ein System aus sechs Schichten (zwei äuSSere und vier inneren Schichten). Über die erste Schicht erfolgt die Eingabe im Netz. Die weiteren fünf Schichten führen einfache mathematische Funktionen aus.

1.1 ANFIS-Klasse

Das Neuro-Fuzzy-Model wird nach dem bekannten ANFIS-Model eingerichtet. Das Model hat insgesammt 6 Schichten, 2 AuSSen- und 4 Innenschichten. Das ANFIS-Model wird in einer Klassendatei ausgelagert. Die Klasse verfügt über mehrere Methoden, einige davon Hilfsmethoden. Folglich werden die Wichtigsten davon in Unterkapiteln gestellt.

1.1.1 Konstruktor

Die ANFIS-Klasse vefügt über einen einzigen Konstruktor. Er hat einen Pflicht- und drei Optionalparameter - *num_sets* und entsprechend *path*, *mf_type*, *gradient_type*. Der Parameter *num_sets* besagt wie viele Fuzzy-Mengen pro EingangsgröSSe zu erstellen sind. Der Pfad wird durch ein String - *path* - gegeben. Weiterhin unterscheide ich zwischen zwei Weisen der Berechnung von Zugehörigkeit, deren Vergleich wurde in der weitere Dokumentation erläutert. Anschliesend ergibt die Variable *gradient_type* die GröSSe der Trainingsdaten, oder die Art des Gradient Descent. Es unterscheiden sich drei Arten von Gradient Descent Verfahren - Stochastic, Mini-Batch und Batch Gradient Descent Verfahren. Die gegebene Anordnung der Begriffe entspricht der in dem Programmcode, sprich *gradient_type=0* entspricht dem stochastischen Verfahren. Der Wert *1* heiSSt, dass das Mini-Batch ausgewählt wird und *2* - Batch Gradient Descent. Die drei Arten unterscheiden sich in dem Punkt, dass die Variablen zu Unterschiedlichen Zeitpunkten angepasst