Daraus folgt, dass wir die Produktregel L mal anwenden müssen. Wir gehen wie folgt vor: Wir betrachten den Gradienten für die l-te Schicht, welcher gegen ist durch D, siehe Gleichung 3.

Wir differenzieren in G nun D bzgl. x, wobei wir dies elementweise machen, da x in jedem Diagonalelement von D auftaucht. Natürlich müssen wir auch hier die Kettenregel wieder anwenden:

Um die Ableitung nun

Um die Notation zu vereinfachen, definieren wir die folgende Matrix:

In einer Implementierung wird der Gradient gegeben durch Gleichung 3 rekursiv berechnet.

Bei einer Implementierung des Gradienten gegeben durch Gleichung 3, kann dieser rekursiv berechnet werden, wobei man ähnlich wie bei der Backpropagation vorgeht.

Um den Gradienten in einem Computerprogramm zu berechnen, geht man rekursiv vor.

Wir bemerken, dass die abbildung natürlich selbst auch ein neuronales Netzwerk ist.

Dabei wird zunächst das neuronale Netzwerk an dem Punkt x ausgewertet und während die Information vorwärts durch das Netz propagiert, wird in jeder Schicht die Aktivierung gespeichert. Dann wird der sogenannte Fehler rückwärts durch das Netz propagiert, indem man die Kettenregel von der letzten Schicht zur ersten Schicht in umgekehrter Richtung anwendet. Die folgende iterative Berechnung kann zu diesem Zweck verwendet werden

Wir bemerken, dass D gilt.

Um die Hesse matrix für f zu erhalten, müssen wir den gradienten nochmals bzgl. x ableiten. Da x für die Berechnung von jedem D gebraucht wird, müssen wir jedes D nach x ableiten.

Wir sehen in Gleichung 3, dass wir nl mal die Produktregel anwenden müssen.

Das ist möglich, weil der Output

In diesem Kapitel werden verschiedene Preliminaries eingeführt und mathematische Grundlagen geschaffen, welche für die Betrachtung von neuronalen Netzwerken benötigt werden. Zuerst stellen metrischen Graphen und ihre Eigenschaften vor. Dann definieren wir auf so einem metrischen Graphen eine Menge von Differentialgleichungen, welche Dift-Diffussiongleichungen genannt werden und den Verkehrsfluss auf einem kompakten Straßennetz modellieren. Danach stellen wir neuronale Netzwerke mit ihren Eigenschaften und ihrer Relevanz als Funktionenapproximatoren vor und gehen am ende dieses Kapitels auf PINNs ein, welche eine Methode zur Approximation der Lösung verschiedener Differentialgleichungen bieten und im Moment ein heiß diskutiertes Thema in der angewandten Mathematik sind.

Als nächstes berechnen wir die Ableitung 2. Ordnung des neuronalen Netzwerkes bzgl. einer Inputvariable. Dies ist ein , welcher gegeben ist durch

Da man in partiellen Differentialgleichungen, wie der Name schon sagt, nur einzelne

Desweiteren verwenden wir für ein numerisches Experiment ein neuronales Netzwerk, welches durch folgende Formel definiert ist

In welchem die Dichte

für die numerische Lösung von potenziellen MFG- und MFC-Modellen

zur Lösung hochdimensionaler Probleme durch die Kombination von Lagrangeschen und Eulerschen Gesichtspunkten und die Nutzung der jüngsten Fortschritte beim maschinellen Lernen.

Die Idee stammt aus Paper 3, in welchem hochdimensionale MFG- und MFC-Modelle annähernd gelöst werden, indem Lagrangesche und Eulersche Sichtweisen kombinieren werden und eine maßgeschneiderte Parametrisierung des neuronalen Netzes für das Potenzials der MFG/MFC-Lösung verwendet wird, um jegliche räumliche Diskretisierung zu vermeiden.

Wir lösen hochdimensionale Probleme annähernd, indem wir Lagrangesche und Eulersche Sichtweisen kombinieren und die jüngsten Fortschritte des maschinellen Lernens nutzen.

Schließlich hilft uns eine maßgeschneiderte Parametrisierung des neuronalen Netzes für das Potenzials der MFG/MFC-Lösung, jegliche räumliche Diskretisierung zu vermeiden.

Wir stellen nun unsere Parametrisierung des neuronalen Netzes für das Potenzials

Für das dritte numerische Experiment verwenden wir einen ganz anderen Ansatz neuronale Netzwerke einzusetzen. Wir

Die Benutzung eines einzelnen neuronalen Netzwerkes ist durch die Hoffnung motiviert, dass die Neuronen in den versteckten Schichten sich die Struktur des Graphen und die daraus folgenden Bedingungen für alle kanten antrainieren, d.h. dass alle Wechselwirkungen im Graph von den Neuronen berücksichtigt werden.

Außerdem hoffen wir, dass sich der Rechenaufwand verringert, wenn das Netz die Ausgabe für alle Kanten gleichzeitig erzeugt.

Dass das Netzwerk die Ausgabe für alle Kanten gleichzeitig erzeugt ist ebenfalls ein Vorteil, da in einer Implementierung nur die Ausführung eines Netzwerkes anstatt mehrere notwendig ist.