Anwendung neuronaler Netzer zur Verkehrsflussoptimierung

Till Hildebrandt, inf102835*

University of Applied Sciences, Wedel

E-mail: till.hildebrandt@gmail.com

INHALTSVERZEICHNIS

1 EINLEITUNG

Die folgende Ausarbeitung befasst sich mit der Optimierung von Ampelschaltungen in Verkehrsnetzen mithilfe neuronaler Netze, die sich den erweiterten Verkehrsmanagementsystemen (*engl.* Advanced Traffic Management System [1]) zurordnen lässt, und den dazu benötigten Werkzeugen. Zunächst wird das Problem beschrieben, die Motivation hergeleitet und die gegenwärtige Situation aufgezeigt. Anschließend werden neuronale Netze[2] in ihren Arten und Prinzipien sowie die Darstellung von Verkehrsnetzen in mathematischen Systemen behandelt. Diese Abschnitte bilden die Grundlage, um den Optimierungsansatz mit dem Hopfield-Modell[3] vorzustellen und dessen Resultate zu betrachten und Grenzen aufzuzeigen.

Ziel ist es den Verkehrsfluss in Städten so zu optimieren, dass der Verkehr möglichst störungsfrei fließt. Ein besserer Verkehrsfluss bedeutet weniger Staus, heißt weniger Umweltbelastung und weniger Unfälle.

1.1 Motivation

Die aktuelle politische Situation im Kleinen, sowie der Klimawandel im Großen, treiben uns immer weiter dazu nach Möglichkeiten zu streben verantwortungsvoller mit unserer Umwelt umzugehen. Ein verbesserter Verkehrsfluss ermöglicht es nicht nur dem Reisenden (im Durchschnitt) schneller sein Ziel zu erreichen, er geht optimaler Weise auch einher mit weniger Standzeiten, weniger Motoren im Leer-

[1] Wikipedia, Advanced Traffic Management System .
[2] Wikipedia, Künstliches neuronales Netz.
[3] Wikipedia, Hopfield-Netz.

lauf und in energieaufwändigen Prozessen wie dem Anfahren, bei dem viel Energie aufgebracht wird und damit entsprechend viel CO_2 wie andere Schadstoffe produziert werden. Dies bedeutet auch weniger Verschleiß, was zu einem längeren Verwenden des Fahrzeugs und somit zu einer Reduktion der CO_2-Bilanz[1] der Herstellung zugute kommt.

Zudem steigen die Anzahl zugelassener Fahrzeuge und das damit verbundene Verkehrsaufkommen seit Aufkommen des Automobils an fast stetig an[2,3]. Einzig die von der Bundesregierung ausgesprochene Abwrackprämie[4] im Jahr 2008 hat, zumindest in Deutschland, Wirkung gezeigt. Dennoch sind die Zahlen weiterhin am Steigen und somit ist auch künftig mit einer zunehmenden Belastung des Straßennetzes zu rechnen. Ein besserer Verkehrsfluss kann von einer höheren Effizienz der Auslastung der Kapazitäten der Straßen profitieren und dabei helfen Investitionen in infrastrukturelle Projekte im besten Fall zu vermeiden.

Abschließend besteht die sowohl emotionale als auch statistisch begründbare Vermutung, dass ein verbesserter Verkehrsfluss mit weniger und kürzeren Standzeiten zu einer angenehmeren Reise mit mehr Umsicht, Ruhe und weniger Unfällen führt. Statistisch ist eine Korrelation zwischen Fahrtzeit und der Wahrscheinlichkeit, dass ein Unfall eintritt zu erwarten.

$$[P_{strecke-kurz}(Unfall) < P_{strecke-lang}(Unfall)]$$

Abbildung 1: Stimulus Funktion

Es hat sich gezeigt, dass neuronale Netze ein gutes Werkzeug im Umgang mit Problemen sein können, die mit klassischen Verfahren

[1] Wikipedia, CO_2-Bilanz.
[2] Statista, *engl.* Number of vehicles in use worldwide
[3] Statista, PKW-Bestand in Deutschland
[4] Wikipedia, Umweltprämie.

nur schwer adressierbar sind. Dabei handelt es sich in der Regel um Probleme, die mit Daten umgehen, in denen es irgendeine Form an statistischen Zusammenhängen/Mustern gibt. So wurde mithilfe neuronaler Netze Software produziert, die eine künstliche Intelligenz für das Spiel Go[1], bei dem der Mensch lange als ungeschlagen galt, realisiert oder eine, die eine Synchronisation von geschriebenem Text auf Mundbewegungen eines Videos bewerkstelligt[2]. Weitere Beispiele sind:

- Schrifterkennung[3]
- Spracherkennung[4]
- Gesichtserkennung[5]
- Aktienkursanalysen[6]
- Kaufempfehlungen
- Textübersetzungen

Bei dem hier behandelten Problem der Verkehrsflussoptimierung kann, ähnlich wie bei den oben genannten Probleme, ein starker statistischer Zusammenhang angenommen werden. In bestimmten Rhythmen fahren mehr oder weniger Menschen bestimmte Straßen in bestimme Richtungen. Der Verkehr ist dabei unterschiedlich dicht und schnell. Manche Strecken sind zu bestimmten Zeiten mehr ausgelastet, andere weniger. Klingt nach einem großartigen Anwendungsfall für neuronale Netze.

[1] Wikipedia, Go (Spiel).
[2] Joon Son Chung and Andrew Zisserman, Oxford University, Out of time: automated lip sync in the wild.
[3] Pythonprogramming, Image Recorgnition with Python.
[4] Geoffrey Hinton et al., Deep Neural Networks for Acoustig Modeling in Speec Recognition
[5] Steve Lawrence, Face Recognition: A Convolutional Neural-Network Approach
[6] Daniel Handloser, Karlsruhe Institute of Technology, Prädiktion von Aktienkursen mit Neuronalen Netzen

1.2 Gegenwärtige Situation

Derweil gibt es keine einheitlich angewandtes System zur Ampelsteuerung in Deutschland. Ampelphasen können fest definiert sein oder verkehrsabhängig gesetzt werden. Bei festen Definitionen werden die Zeiten in einem Signalzeitenplan[1] festgehalten.

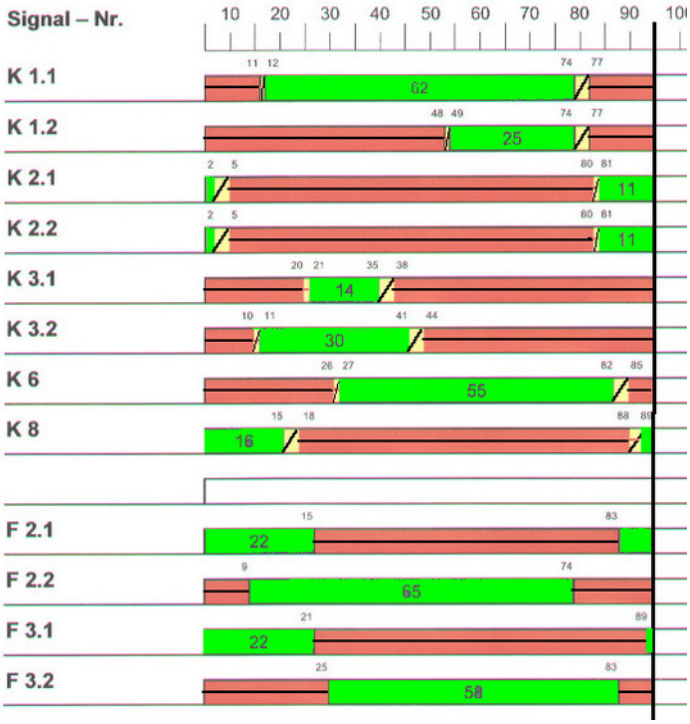


Abbildung 2: Signalzeitenplan

Auf der X-Achse befindet sich die Zeit in Sekunden, auf der Y-Achse die diskreten Werte der beteiligen Signalgeber. Rot- und Grünphasen sind in der jeweiligen Farbe dargestellt. Die übrigen Elemente entsprechen den beiden Gelbphasen Gelb und Rot/Gelb. Zu verstehen ist das Diagramm im Zeitfluss von links nach rechts. Rechts angekommen, geht es links wieder los.

Bei der verkehrsabhängigen Steuerung werden die einzelnen Verkehrsströme je nach Bedarf bedient. Sie funktioniert mithilfe vielerlei Technologien, so gibt es Induktionsschleifen und Bewegungsmelder,

[1] Wikipedia, Signalzeitenplan.

aber auch Videokameras unterstützen die Optimierung des Verkehrsflusses bei manchen Systemen. Vorwiegend handelt es sich dabei um Systeme, die an einzelnen Kreuzungen wirken und keinen größeren Bereich in Betracht ziehen. Sie sind zum Beispiel so konstruiert, dass alle ankommenden Fahrzeuge einer ankommenden Fahrzeugwelle die Kreuzung passieren können, die Ampelphasen werden dementsprechend angepasst. Dabei ist darauf zu achten, dass die Wartezeiten der anderen Verkehrsteilnehmer nicht unzumutbar werden oder ein Rückstau der Abbiegespuren entsteht, die die anderen Spuren einengt.

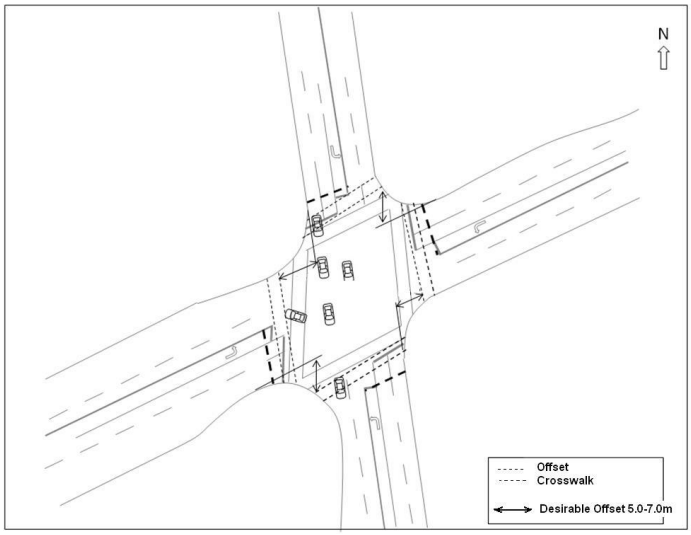


Abbildung 3: Rückstau an einer Kreuzung

Daraus ergibt sich eine variable Umlaufzeit der Ampelphasen. Fest definierte Abläufe können zudem per Automatik zwischen verschiedenen Programmen wechseln, so kann auf verschiedene Verkehrsbelastungen (Berufs-, Tages- und Nachtverkehr usw.) reagiert werden.

Weitere Elemente des Straßenverkehrs, die den Verkehrsfluss direkt beeinflussen können, sind zum Beipiel Fußgängerampeln mit Anforderung oder einem Zebrastreifen. Es ist zwar wichtig, möchte man ein ganzheitliches System implementieren, diese Elemente zu betrach-

ten, in dieser Ausarbeitung werden diese Elemente jedoch noch außen vorgelassen.

Der im Folgenden vorgestellte Optimierungsansatz mit dem Hopfield-Netzwerk betrachtet hingegen ein komplexes System an Straßen. Sozusagen einen Teilgraph des Verkehrsnetzes. Angewandt wurde das System im Zuge der Olympischen Spiele in Atlanta 1996[1].

[1] John F. G. and Khalid J. E., Traffic Management Applications of Neural Networks

2 NEURONALE NETZE

Die nachfolgenden Ausführungen und Grafiken zum Schaffen eines grundlegenden Verständnisses für neuronale Netze, die hier verwendet werden, basieren im wesentlichen auf dem E-Book "Neural Networks and Deep Learning"[1].

Im Rahmen des maschinellen Lernens stellen die neuronalen Netze einen elementaren Ansatz dar, der in vielen weiteren Modellen Verwendung findet. Neuronale Netze wie das maschinelle Lernen an sich stellen eine andere Herangehensweise dar, als die klassischer, deterministischer Algorithmen. Anstatt dem System eine eindeutige Abfolge von Anweisungen mitzuteilen, um eine konkrete Problemstellung zu lösen, wird ein Modell definiert und dieses mit verschiedenen Beispielen konfrontiert - die Beispiele sind dabei Tupel aus Eingangsgröße und erwarteter Ausgangsgröße. Die Dimensionen von Eingangs- und Ausgangsgröße können sich dabei gleich sein, müssen es aber nicht. So können als Eingabe Bilder dienen und als Ausgaben konkrete Klassen, um beispielsweise Hunde von Katzen unterscheiden zu können[2]. Anstatt nun algorithmisch zu definieren, was einen Hund von einer Katze unterscheidet, wird es dem zuvor erstellten Modell überlassen, anhand der gegebenen Eingaben und erwarteten Ausgaben, eigenständig Regeln abzuleiten, um mit dessen Hilfe auch unbekannte Eingaben klassifizieren zu können.

Dieser Ansatz wird auch als *Soft Computing*[3] bezeichnet.

---

[1] Michael Nielsen, Neural Networks and Deep Learning

[2] Andrea Baraldi und Flavio Parmiggiani, A Neural Network for Unsupervised Categorization of Multivalued Input Patterns: An Application to Sattellite Image Clustering

[3] Wikipedia, *engl.* Soft Computing.

2.1 Grundprinzipichen

Neurnale Netze beziehen sich eigentlich auf biologische Nervenzentren. In der Informatik hat man nur das grundlegende Prinzip adaptiert und versucht Strukturen zu erzeugen, die anhand von Daten lernen können. Es wird nicht versucht ein *echtes* Gehirn nachzuempfinden, in dem Metastrukturen auftreten, in denen bestimmte Bereiche bestimmte Aufgaben haben. An dem Bereich wird natürlich auch geforscht, ist aber Gegenstand der Computational Neuroscience[1]. Dennoch sei die biologische im Folgenden beschrieben wie Neuronen funktionieren:

Die Arbeitsweise ist erstaunlich einfach: Immer wenn die Summe der Eingangssignale einen bestimmten Schwellenwert überschreitet, sendet die Zelle ein Ausgangssignal. Bleibt die Eingangserregung unter der Grenze, reagiert die Zelle nicht. Am Ende der axonalen Verzweigungen stellt eine besondere Struktur, die Synapse, den Kontakt zu anderen Neuronen her. Die meisten Synapsen funktionieren so: Je stärker die Erregung im Axon, desto mehr Moleküle einer Überträgersubstanz schüttet die Synapse aus. Der Überträgerstoff (Neurotransmitter) wandert zur Zielzelle. Manche Neurotransmitter erhöhen die elektrische Erregung der ängefunktenSZelle, andere hemmen sie.

Das Netzwerk der Neuronen in der Großhirnrinde (wegen ihrer Form auch „Pyramidenzellen“) ist im Gegensatz zu einem Computer nicht nach einem detaillierten Plan geknüpft, sondern weitgehend zufällig organisiert. Sind miteinander verbundene Zellen gemeinsam aktiv, verstärken sich die Synapsen. Demnach aktiviert das Lernen immer wieder eine Anzahl miteinander verknüpfter Pyramidenzellen. Deren Verbindung verstärkt sich nach

[1] Wikipedia, Computational Neuroscience.

und nach, „neuronale Netzwerke" entstehen. Je öfter sich der synaptische Lernprozess wiederholt, desto leichter lässt sich dieses „Netzwerk" aktivieren.

2.1.1 *Perzeptron*

Als elementaren Bestandteil eines neuronalen Netzes dient das *Perzeptron* - dieses stellt die kleinste Einheit eines neuronalen Netzes dar und wird auch als "künstliches Neuron" bezeichnet.

Grundsätzlich akzeptiert ein Neuron einen beliebig großen Input bestehend aus Features *x1, x2, ..., xn* und berechnet daraus ein Ergebnis.

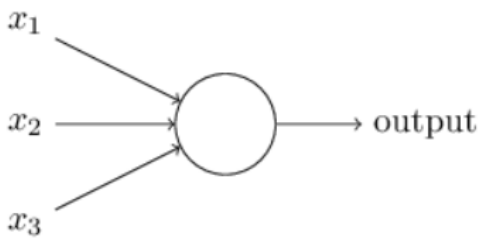


Abbildung 4: Perzeptron

Im gezeigten Bild ist beispielsweise ein Neuron dargestellt, das drei Inputgrößen akzeptiert und daraus einen Output produziert. Um den Output zu berechnen werden Gewichte (*engl. weights*) eingeführt. Ob das Neuron 0 oder 1 als Output liefert, hängt dann davon ab, ob die gewichtete Summe der Eingangsgrößen einen zu definierenden Schwellwert überschreitet.

Dies kann anhand der nachfolgenden Formel verdeutlicht werden:

$$\text{output} = \begin{cases} 0 & \text{if } \sum \omega_i x_i \leqslant \text{ threshold (Schwellwert)} \\ 1 & \text{if } \sum \omega_i x_i > \text{ threshold (Schwellwert)} \end{cases}$$

Abbildung 5: Berechnung des Outputs.

Dies ist das grundlegende Modell. Grundsätzlich kann sich das Perzeptron auch als ein "Entscheidungs-Unterstützer" vorstellt werden, der eine Entscheidung trifft, in dem er konkrete Fakten mit einem bestimmten Gewicht versieht.

Das gezeigte Modell ist augenscheinlich sehr simpel und noch sehr weit von dem entfernt, was als ein neuronales Netz bezeichnet werden würde. Es ist allerdings ohne Weiteres denkbar, das gezeigte Model komplexer zu gestalten, indem mehrere Perzeptrons miteinander verknüpft werden, so dass beispielsweise das nachfolgende Netzwerk entstehen könnte:

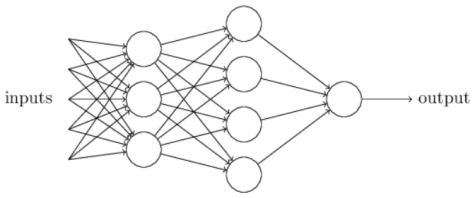


Abbildung 6: Mehrschichtiges neuronales Netz.

In Grafik[5] wurde ein Schwellwert eingeführt, der überschritten werden muss, damit ein Perzeptron aktiviert wird. Um das Modell zu vereinheitlichen, kann der *Bias* definiert werden, der den negativen Schwellwert darstellt. Durch diese Maßnahme kann die Aktivierungsfunktion des Perzeptrons dann geschrieben werden als:

$$\text{output} = \begin{cases} 0 & \text{if } \sum \omega \cdot x + b \leqslant 0 \\ 1 & \text{if } \sum \omega \cdot x + b > 0 \end{cases}$$

Abbildung 7: Berechnung des Outputs bei Verwendung eines Bias.

Inhaltlich kann der Bias als ein Maß verstanden werden, aus dem hervorgeht, wie *leicht* ein Perzeptron aktiviert werden kann. Nimmt der Bias einen großen Wert an, so kann das Perzeptron einen Wert von 1 annehmen, auch wenn das Produkt aus den Gewichten und den

Eingangsgrößen einen negativen Wert annimmt. Gleiches gilt selbstverständlich auch für einen kleinen Bias, der zur Folge hat, dass ein Perzeptron träger reagiert.

2.1.2 *Sigmoid-Neuronen*

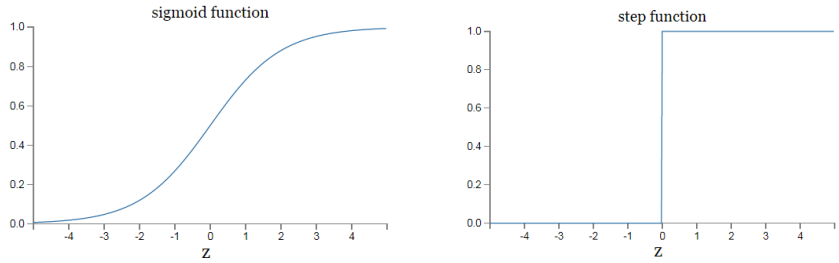
Eine Weiterentwicklung des zuvor vorgestellten Modells stellen Sigmoid-Neuronen dar. Diese Weiterentwicklung wird dann erforderlich, wenn das Anpassen der Gewichte - also letztlich das Lernen - betrachtet wird. Dabei ist das Ziel, dass eine kleine Anpassung eines Gewichts auch nur eine kleine Änderung des Outputs zur Folge hat. Das zuvor betrachtete Perzeptron ist lediglich in der Lage 0 oder 1 als Output zu liefern, so dass Änderungen an den Gewichten keine stetige Änderung des Outputs zur Folge haben, sondern folgenlos bleiben können bis irgendwann ein Sprung von 0 auf 1 oder umgekehrt stattfindet, was wiederum eine große Änderung darstellt.

Die Weiterentwicklung besteht nun in einer Verfeinerung der Aktivierungsfunktion. Anstatt eine Sprungfunktion[9b] zu verwenden, die lediglich 0 und 1 als Funktionswert annehmen kann, wird die Sigmoid Funktion[9a] eingeführt, die die folgende Form hat:

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}$$

Abbildung 8: Sigmoid-Funktion.

Der entscheidende Unterschied kann an den beiden nachfolgenden Grafiken verdeutlicht werden, die jeweils die Kurve der entsprechenden Funktion darstellen:



Sigmoid Funktion. Step Funktion.

Abbildung 9: Vergleich der Aktivierungsfunktionen *Sigmoid* und *Step-Funktion*

2.1.3 *Architektur neuronaler Netze*

Mit diesen Bestandteilen als Ausgangspunkt können nun tatsächlich konkretere Neuronale Netze und deren Architekturen eingeführt werden. Neuronale Netze bestehen üblicherweise aus mehreren Schichten, den *Layern*. Diese lassen sich grundsätzlich in drei Kategorien aufteilen: Input, Hidden und Output. Neuronale Netze beinhalten für gewöhnlich ein Input-Layer und ein Output-Layer sowie dazwischen beliebig viele Hidden-Layer. Die Form der Input- und Output-Layer ist dabei sehr naheliegend: das Input-Layer hat die gleiche Struktur wie die des Inputs und das Output-Layer hat entsprechend die gleiche Struktur wie der Output.

Angenommen es sollen Bilder der Größe 28×28 Pixel klassifiziert werden und es gibt 10 mögliche Klassen, dann besteht das Input-Layer aus $28 \times 28 = 784$ Neuronen und das Output-Layer aus 10 Neuronen.

Lediglich der Bereich zwischen Input- und Output-Layer - die Hidden-Layer - lässt sich nicht ohne Weiteres aus dem Input oder dem Output ableiten. Es gibt lediglich Heuristiken, die beim Design der Hidden-Layer angewandt werden können, allerdings keine konkreten Regeln, die befolgt werden müssen. Diese Struktur kann anhand des nachfolgenden Bilds verdeutlicht werden, bei dem - um die

Übersichtlichkeit zu wahren - das Input-Layer etwas komprimiert dargestellt wird:

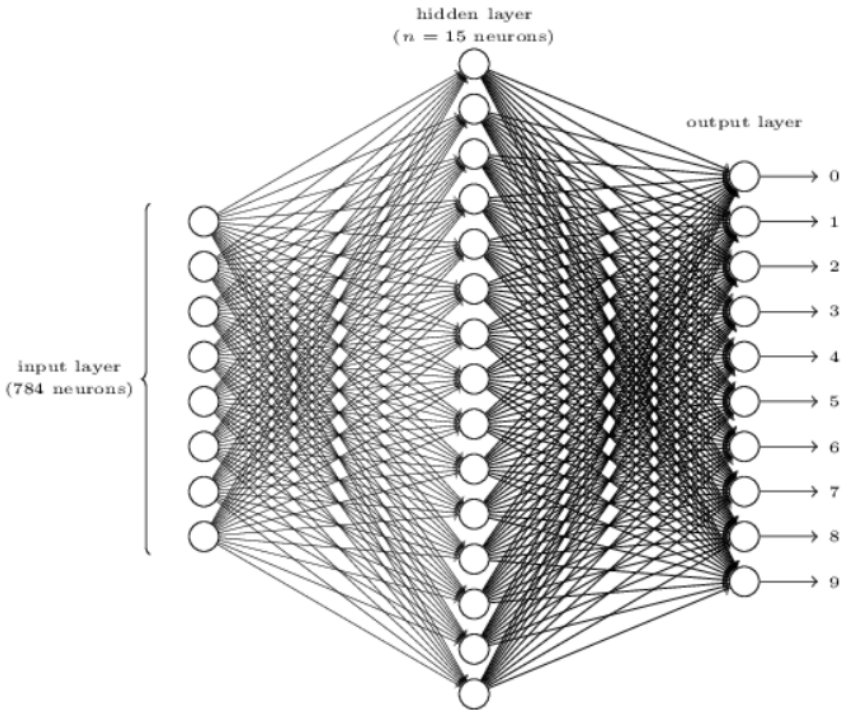


Abbildung 10: Hidden-Layer Darstellung.

2.2 Lernen - Das Anpassen der Gewichte

Das Lernen stellt den zentralen Ansatz von neuronalen Netzes dar. Eng im Zusammenhang mit dem Lernen steht eine Kosten-Funktion, die häufig auch als Verlust-Funktion bezeichnet werden kann. Diese stellt letztlich den Fehler zwischen dem Erwartungswert und dem tatsächlichen Wert, den das neuronale Netz berechnet, dar. Mathematisch betrachtet ist das grundlegende Prinzip des Lernens, diese Funktion zu minimieren, also zu gewährleisten, dass die Abweichungen zwischen Erwartungswert und tatsächlichem Wert möglichst gering sind. Es sind grundsätzlich viele verschiedene Verlust-Funktionen denk-

bar, eine, die jedoch eine breite Verwendeung findet, ist die quadratische Kosten-Funktion - auch als *mean squared error* (MSE) bezeichnet.

$$C(w, b) \equiv \frac{1}{2n} \sum_{x} \| y(x) - a \|^2$$

Abbildung 11: Mean Squared Error (MSE).

Dabei beschreiben w und b die Gewichte bzw. die Bias des neuronalen Netzes und n stellt die Anzahl der Trainingsdaten dar. Der Vektor a beschreibt den Output des Netzes und $y(x)$ stellt den Erwartungswert zu einem Input x dar. Das Ziel besteht nun darin, die Gewichte und Bias so zu manipulieren, dass die gezeigte Funktion einen möglichst kleinen Wert annimmt. Beim Lernen wird zusätzlich zwischen überwachtem und unüberwachtem Lernen unterschieden. Beim Überwachten Lernen kommt die Information, wie gut ein produziertes Resultat ist, von außen ins Netz. Das unüberwachte Lernen kommt ohne diese direkte Infromation aus. Hier dienen dem Netz bekannte Funktionen dazu die Güte eines Ergebnisses zu bewerten und anhand dessen zu Lernen (die Gewichte anzupassen und neue Resultate zu erzeugen).

2.3 Arten und Anwendungsgebiete

Es gibt sehr viele, die Arten an neuronalen Netzen vollumfassend aufzulisten, ist nicht möglich. Quellen unterscheiden zwischen der Struktur des Netzes, also der Fragestellung, wie die Neuronen angeordnet sind oder den Anwendungsfällen. Die beiden am weitesten verbrei-

teten Typen sind *Convolutional Neural Networks*[1] und *Recurrent Neural Networks*[2]

Convolutional Neurnal Networks

Die *Convolutional Networks* sind beim Thema Bildverarbeitung erste Wahl. Die Verarbeitung der Bildinformationen wird durch die Struktur des Netzes beibehalten und benachbarte Informationen (Pixel) werden miteinander verarbeitet bzw. gefaltet/reduziert. Diese Netze kommen zum Beispiel bei Gesichtserkennung oder Bildkategorisierungen zum Einsatz.

Recurrent Neurnal Networks

Die *Recurrent Neural Networks* werden in der Regel dazu verwendet über Sequenzen an Daten Vorhersagen zu treffen. So können zum Beispiel Wetterdaten oder Aktienkurse analysiert und vorhergesagt werden.

Prinzipiell können neuronale Netze gut in Systemen verwandt werden, welche eine logische Struktur haben, die funktional jedoch nicht beschrieben werden kann. Weitere Anwendungsgebiete in denen diese schon heute zum Einsatz kommen sind Marketing, Verkauf und Medizin.

[1] Wikipedia, Convolutional Neural Network
[2] Wikipedia, Recurrent Neural Network

3 VERKEHRSMODELLE

Verkehrsmodelle bestehen zunächst aus dem Streckennetz oder Verkehrsgraph und den Teilnehmern. Natürlich kann es, je nach Anwendungsfall noch Sinn machen, noch andere Einflussfaktoren mit zu berücksichtigen, darauf wird jedoch im Folgenden jedoch verzichtet. Die Fragestellung: Wie lassen sich diese System in mathematischen Systemen abbilden?

Bei Verkehrsflussporblemen gibt es zwei grundsätzliche Ansätze[1] [2]. Einmal die mikroskopischen Modelle[3], die das Gesamtsystem sehr feingranular abbilden. Sie basieren auf individuellem Verhalten und bilden Verkehrsteilnehmer als einzelne Objekte ab. Dem gegenüber stehen die makrsokopischen Modelle[4], die mehr Interesse an Durchschnittsverhalten haben. Sie betrachten etwas wie Verkehrsdichte und Durschnittsgeschwindigkeit.

Mit dem mikroskopischen Ansatz ist eine sehr präzise Arbeit möglich, die jedoch viel Rechenleistung erfordert, da die Position jedes Objekt in jedem Schritt neu berechnet werden muss. Wohingegen die makroskopischen Ansätze zwar etwas unpräziser sind, dafür, weil sie weniger Details haben, entsprechend günstiger im Bezug auf die benötigte Rechenleistung.

---

[1] Sven Maerivoet and Bart De Moor, 2008. Traffic Flow Theory. Department of Eletrical Engineering ESAT-SCD (SISTA), Katholieke Universiteit Leuven

[2] Springer, 2008. Traffic Flow for 1-D. Pedestrian Dynamics, Feedback Control of Crowd Evacuation

[3] Wikipedia, Microscopic Traffic Flow Model

[4] Wikipedia, Macroscopic Traffic Flow Model

3.1 Mikroskopische Modelle

Im Gegensatz zu den makroskopischen, werden bei den mikroskopischen Ansätze einzelne Fahrzeuge als Objekte simuliert. Die dabei simulierten Dimensionen beziehen sich dementsprechend auf mikroskopische Eigenschaften wie der Position und der Geschwindigkeit eines Verkehrsteilnehmers.

Eine prominente Gruppe an Vertretern dieser Modelle sind die Car-FollowingModelle[1]. Hier gibt es mehrere Variationen[2], Grundlegendes Prinzip ist jedoch die Abhängigkeit der Beschleunigung und Geschwindigkeit vom Vorausfahrenden auf den dahinter Fahrenden. Mit anderen Worten jeder Fahrer reagiert auf den ihn umgebenden Verkehr.

$$[\mathrm{Response}]_n \ \alpha \ [\mathrm{Stimulus}]_n$$

Abbildung 12: Grundlegende Philosophie von Car FollowingModellen

Als Reaktion kann ein Fahrer beschleunigen oder abbremsen. Was passiert und wie intensiv, hängt von Stimulus ab. Der Stimulus ist dabei eine Funktion, die abhängig von der Geschwindigkeit, dem Abstand zum Vordermann und einer Menge andere Faktoren ist.

$$a_n^t = f_{stimulus}(v_n, \Delta x_n, \Delta v_n)$$

Abbildung 13: Stimulus Funktion

An dieser Stelle scheiden sich die unterschiedlichen Modelle[2].

Dem Beispiel des von General Motors vorgeschlagenem Follow-the-leaderKonzepts[2] liegen zwei Annahmen zugrunde: umso höher die Geschwindigkeit, umso höher der Abstand zum Vordermann, aber

[1] Springer, 2008. Traffic Flow for 1-D. Pedestrian Dynamics, Feedback Control of Crowd Evacuation

[2] Mathew T.V., 2014. Transportation Systems Engineering.

niemals unterhalb des Sicherheitsabstands. Gleiches gilt für die Höchstgeschwindigkeit, welche einer asymptotische Grenze für die Geschwindigkeit bei geringen Verkehrsdichten entspricht.

Sei Δx_{n+1}^t den Abstand für das $(n+1)$-te Fahrzeug, den Sicherheitsabstand Δx_sicher. v_{n+1}^t und v_n^t die Geschwindigkeiten, der notwendige Abstand ergibt sich dann zu:

$$x_{n+1}^t - x_n^t = \Delta x_{sicher} + \tau v_{n+1}^t$$

Abbildung 14: Abstandsfunktion

bei dem τ einen Elastizitätskoeffizienten darstellt.

Leitet man diese Funktion jetzt über die Zeit ab, erhält man die dazugehörigen Gleichungen für die Geschwindigkeit und die Beschleunigung:

$$v_{n+1}^t - v_n^t = \tau a_{n+1}^t$$

Abbildung 15: Geschwindigkeitsfunktion

$$a_{n+1}^t = 1/\tau(v_{n+1}^t - v_n^t)$$

Abbildung 16: Beschleunigungsfunkton

General Motors hat einige Varianten an Elastizitäts-, auch Sensitivitätskoeffizienten, veröffentlich. Das grundlegendste Modell ergibt sich zu:

$$a_{n+1}^t = \frac{\alpha_{l,m}(v_n^t)^m}{x_{n+1}^t - x_n^t}[v_{n+1}^t - v_n^t]$$

Abbildung 17: Beschleunigungsfunkton

mit

l - Abstand zum Vordermann, [-1, +4]

m - Geschwindigkeitsexponent, [-2, +2]

α - Sensitivitätskoeffizient

3.2 Makroskopische Modelle

Makroskopische Modelle (auch Verkehrsflussmodelle) sind Modelle, die im Gegensatz zu dem vorher beschriebenen, die auf Durchschnittswerten über die Verkehrssituation arbeitet. Maßgeblich werden dabei die Dimensionen Geschwindigkeit, Dichte und Fluss unterschieden.

Geschwindigkeit entspricht einer zurückgelegten Strecke pro Zeiteinheit. Da es in der Praxis - und im (Gedanken-)Ansatz der Makroskopie - nicht möglich oder sinnvoll ist auf tatsächlich, exakt gemessene Werte zurück zu greifen, wird die Durchschnittsgeschwindigkeit entweder im Bezug zur Zeit oder im Bezug zur zurückgelegten Strecke gesehen. Man spricht von *Time Mean Speed* und *Space Mean Speed*.

$$v_t = \frac{1}{m} * \sum i = 1^m v_i$$

Abbildung 18: Geschwindigkeit abh. der Zeit

$$v_s = (\frac{1}{n} * \sum i = 1^n \frac{1}{v_i})^{-1}$$

Abbildung 19: Geschwindigkei abh. der zurückgelegten Strecke

Die Dichte (k) entspricht der Anzahl an Fahrzeugen pro Längeneinheit, es wird zwischen der kritischen Dichte (k_c) und der Staudichte (k_j) unterschieden. Dabei entspricht die kritische Dichte der maximal erreichbaren Dichte eines freien FlussTODO, die Staudichte der Dichte, die in Stausituationen erreicht wird. Die Dichte im Allgemeinen kann auch als Inverses des Raums zwischen zwei Fahrzeugen verstanden werden ($k = 1/s$.

Die Dichte auf einem Streckenabschnitt zu einer gegebenen Zeit entspricht dem Inversen des mittleren Abstands der Fahrzeuge:

$$K(L, t_1) = \frac{n}{L} = \frac{1}{\bar{s}(t_1)}$$

Abbildung 20: Dichte eines Streckenabschnitts

Der Fluss (q) ist die Anzahl an Fahrzeugen, die einen Referenzpunkt in einer gegebenen Zeit passiert. Der Umkehrwert dazu (h) ist in dichtem Verkehr ($k = k_c$) konstant und beschreibt die Zeit, die zwischen dem Vorbeifahren des *n*ten und *(n+1)*ten vergeht. In Stausituationen gilt: $\lim_{q \to \infty} h = 1/q \to \infty$.

$$q = kv$$
$$q = 1/h$$
$$q(T, x_1) = \frac{m}{T} = \frac{1}{\bar{h}(x_1)}$$

Abbildung 21: Fluss eines Streckenabschnitts

Der Dichte-Fluss-Zusammenhang lässt sich grafisch darstellen, als:

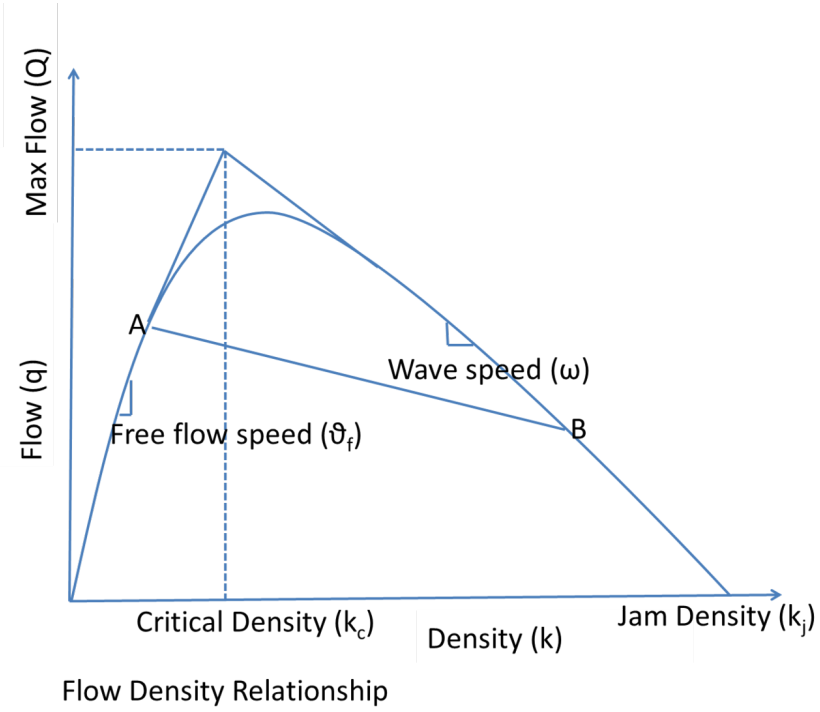


Abbildung 22: Hidden-Layer Darstellung.

4 DAS HOPFIELD NETZWERK UND DER VERKEHRSFLUSS

Ein Verkehrssystem kann als eine verwobene Anordnung von Straßen, deren Fluss durch Ampelschaltungen kontrolliert wird, verstanden werden. Um den maximalen Fluss für eine gegebene Dichte in einem gegebenen Graphen (Verkehrsnetz) zu ermöglichen, ist die Kontrolle der Ampeln grundlegend. Dies gibt insbesondere in diesem stark vereinfachten Gedankenmodell, aber auch in komplexeren Darstellungen kann von einem hohen Einfluss der Schaltungen auf den Verkehrsfluss ausgegangen werden. Da die optimale Schaltung vorher nicht verfügbar ist, ist ein Ansatz mit überwachtem Lernen nicht passend. Benötigt wird in diesem Fall eine Lösung, die mit unüberwachtem Lernen auskommt. In dem Fall, der hier vorgestellt wird, wurde sich für ein Hopfield-NetzTODO entschieden, da sich dessen Struktur sehr gut auf Verkehrsflussprobleme projizieren lässt.

Hopfield-Netze gehören zu den Feedbacknetzen. Sie bestehen aus nur einer Schicht, die sowohl als Ein- wie Ausgabelayer dient. Jedes Neuron ist mit allen anderen, ausgenommen sich selbst, verbunden. Sie geben -1 oder $+1$ aus, was im Allgemeinen bedeutet, dass das Neuren *schaltet* oder eben nicht *schaltet*. In diesem Anwendungsfall entspricht das der Grün- bzw. Rotphase. Die Verbindungen zwischen den Neuronen sind gewichtet und symmetrisch, heißt:

$$w_{ii} = 0 \forall_i$$
$$w_{ij} = w_{ji}$$

Abbildung 23: Gewichtssymmetrie

Das Gewichte *zwischen* zwei Neuronen haben einen starken Einfluss auf die benachbarten Neuronen. Sie ziehen sich an oder stoßen

sich ab. Ist $s_j = 1$, ist der Beitrag zum Schalten des nächsten Neurons positiv. Ist das Gewicht negativ, werden auch die Gewichte der Nachbarn negativ beeinflusst. In Verkehrsnetzen, kann dieses Verhalten interpretiert werden als Erhöhung der Wahrscheinlichkeit, dass die nächste Kreuzung umschaltet, sofern sich die Werte von s_i und s_j unterscheiden. Meint, die eine Ampel ist auf Grün, die nächste auf Rot.

Das Aktualisieren der Neuronen, also die Entscheidung, ob ein Neuron *schaltet* oder nicht, kann symmetrisch oder asymmetrisch durchgeführt werden. Im Allgemeinen ergibt sich das Signal eines Neurons zu:

$$s_i = \begin{cases} +1 & \text{if } \sum_j w_{ij} s_j + b_{ij} >= \theta_i \\ -1 & \text{sonst} \end{cases}$$

Abbildung 24: Aktualisierung eines Neurons

mit:

$w_{i,j}$ Gewichte der Straßensegmente

s_i Interner Status

θ_i Schwellwert

Da die Struktur des Netzes flach und vollständig verbunden (*engl.* fully connected) ist, also jedes Neuron von jedem Neuron Bescheid weiß, muss zwischen symmetrischem und asymmetrischen Aktualisieren unterschieden werden. Beim synchronen Aktualisieren werden alle Neuronen gleichzeitig geschaltet, beim asynchronen diskret hintereinander oder zufällig. An dem symmetrischen Aktualisieren wird häufig kritisiert, dass dieses Verfahren für die mit dem Netz untersuchten Systeme, meist unrealisitisch ist.

Hopfield-Netze haben einen globalen, skalaren Wert, den man als Energie bezeichnet. Die Energie eines Netzes, ergibt sich dabei aus

einer Funktion, der Energie-Funktion. Die Energie-Funktion ist ein grundlegender Parameter beim Entwerfen dieser Netze:

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$

Abbildung 25: Energiefunktion

mit:

$w_{i,j}$ Gewichte der Straßensegmente

s_i Interner Status

θ_i Inputbias

Über die Konvergenz der Energiefunktion

Durch die Symmetrie der Gewichte in Kombination mit asynchronem Updaten wird gewährleistet, dass diese Energie-Funktion monoton abnimmt. Daraus folgt, dass nach mehrmaligem Updaten die Energiefunktion ein lokales Minimum erreicht.[1] Dies wurde von Hopfield selbst bewiesen.

Der Beweis zeigt die Existenz einer Ljapunow-Funktion im Hopfield-Netz[2]. Eine solche Funktion hilft Stabilitätseigenschaften eines konkreten Systems zu überprüfen und ist Gegenstand der Stabilitätstheorie[3]. Diese bemerkenswerte Eigenschaft wurde in verschiedenen weiteren Papern behandelt, sehr schön auch von Jehoshua Bruck im Oktober 1990 [4].

[1] Jehosuha B., 1990. On the Convergence Properties of the Hopfield Model
[2] David J.C. MacKay, Information Theory, Inference, and Learning Algorithms, Seite 508 ff.
[3] Wikipedia, Stabilitätstheorie
[4] Jehoshua Bruck, On the Convergence Properties of the Hopfield Model

5 FAZIT

Ziel dieser Arbeit war es ein grundlegendes Verständnis für eine mögliche Verwendung neuronaler Netze in Verbindung mit Verkehrssystemen zu erlangen. Es konnte gezeigt werden, dass die Verwendung von Hopfield-Netzen einen validen Ansatz vermuten lässt. Jedoch hat sich auch herausgestellt, dass es hier noch mehr bedarf und weitere Aspekte genauer untersucht werden müssen, um eine finale Aussage zu treffen. Hierzu gehört noch vor einer technischen Implementierung als Proof of Concept, ein Konzept zu einem Modell von Verkehrsnetzen in Verbindung mit Hopfield-Netzen und weiteres Verständnis zu dem Zusammenhang der Stabilität der in den Netzen verwendeten Energiefunktion.

Die Frage nach einer technischen Abbildung von Verkehrsnetzen in Verbindung mit Hopfield-Netzen sollte dabei folgende Fragen beantworten:

1) Wie exakt können Verkehrsetze abgebildet werden? Beachte: Fahrspuranzahl, Abbiegespuren, Rückstaus, Bedarfsampeln, Einbahnstraßen, Baustellen etc. 2) Wie granular sollten diese Modelle sein, um zum einen noch performant zu sein und zum anderen überhaupt mit daten belieferbar?

Danach müsste ein Proof of Concept erstellt werden, um das Modell zu verifizieren. Hier kann es hilfreich sein weiteres Verständnis über die Stabilitätseigenschaften zu haben, um ggf. z.B. Parameter des Netzes zu optimieren.

in die mail Im Rahmen dieses Seminars bestand Kontakt zum Landesamt für Straßen, Brücken und Gewässer. Interesse an weiterem scheint gegeben Daten liegen vor.

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS