

**Hochschule Karlsruhe
Technik und Wirtschaft**
UNIVERSITY OF APPLIED SCIENCES

GENETISCHE ALGORITHMEN ZUR OPTIMIERUNG VON HYPERPARAMETERN EINES KÜNSTLICHEN NEURONALEN NETZES

Fakultät für Maschinenbau und Mechatronik
der Hochschule Karlsruhe
Technik und Wirtschaft

Bachelorarbeit

vom 01.03.2018 bis zum 31.08.2018
vorgelegt von

Christian Heinzmann

geboren am 18.02.1995 in Heilbronn
Matrikelnummer: 52550

Winter Semester 2019

Professor	Prof. Dr.-Ing. habil. Burghart
Co-Professor	Prof. Dr.-Ing. Olawsky
Betreuer FZI:	M. Sc. Kohout

Eigenständigkeitserklärung und Hinweis auf verwendete Hilfsmittel

Eigenständigkeitserklärung und Hinweis auf verwendete Hilfsmittel. Hiermit bestätige ich, dass ich den vorliegenden Praxissemesterbericht selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen des Berichts, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

Datum: _____ Unterschrift: _____

Ausschreibung

BACHELORARBEIT

Genetische Algorithmen zur Optimierung von Hyperparametern eines künstlichen neuronalen Netzes

Zur Vermeidung der weiteren Ausbreitung von multiresistenten Keimen in Einrichtungen des Gesundheitswesens (insb. Krankenhäuser) werden am FZI Systeme und Methoden entwickelt, welche helfen sollen die Händehygiene-Compliance von Mitarbeitern dort zu erhöhen. Durch technische Unterstützung bei häufig wiederkehrenden Maßnahmen, soll mehr Zeit geschaffen und gleichzeitig auf die Wichtigkeit von Desinfektionsmaßnahmen aufmerksam gemacht werden. Dies soll mit Hilfe von Augmented-Reality umgesetzt werden. Dabei ist das Ziel, bekannte Prozesse, wie beispielsweise den Wechsel von postoperativen Wundverbänden, visuell zu unterstützen und automatisch zu dokumentieren.

AUFGABEN

Im Kontext der Detektion von Aktionen und Objekten, soll die Optimierung von Neuronalen Netzen mit Genetischen Algorithmen durchgeführt werden. Das Ziel hierbei ist es, ein Framework zu entwickeln und zu implementieren, in welchem künstliche neuronale Netze automatisiert trainiert werden. Unter anderem sollen die Hyperparameter mit Hilfe von Genetischen Algorithmen intelligent angepasst und das trainierte Netz anschließend ausgewertet werden. Diese Aufgaben sollen voll automatisiert ablaufen. Daraus ergeben sich folgende Aufgaben:

- Literaturrecherche über aktuelle Genetische Algorithmen und aktuelle Neuronale Netze
- Einarbeiten in vorhandene Frameworks für Genetische Algorithmen und Neuronale Netze
- Konzeptionierung und Implementierung des ausgewählten Ansatzes zur Optimierung von Hyperparameter des Neuronalen Netzes
- Evaluation und Auswertung speziell unter der Beachtung geringer Datenmengen
- Wissenschaftliche Aufbereitung und Dokumentation des Projekts

WIR BIETEN

- Aktuelle Softwaretools im täglichen wissenschaftlichen Einsatz
- eine angenehme Arbeitsatmosphäre
- konstruktive Zusammenarbeit

WIR ERWARTEN

- Grundkenntnisse in maschinellem Lernen
- Kenntnisse in folgenden Bereichen sind von Vorteil: Python, Tensorflow, C/C++
- selbständiges Denken und Arbeiten
- sehr gute Deutsch- oder Englischkenntnisse
- Motivation und Engagement

ERFORDERLICHE UNTERLAGEN

Wir freuen uns auf Ihre PDF-Bewerbung an Herrn Lukas Kohout, kohout@fzi.de, mit folgenden Unterlagen:

- aktueller Notenauszug
- tabellarischer Lebenslauf

WEITERE INFORMATIONEN

- Start: ab sofort

Inhaltsverzeichnis

Eigenständigkeitserklärung und Hinweis auf verwendete Hilfsmittel	2
Ausschreibung	3
1 Einleitung	6
1.1 Motivation	6
1.2 Aufgabenstellung	6
1.3 Aufbau der Arbeit	7
2 Grundlagen	8
2.1 Optimierungsgrundlagen	8
2.2 Genetische Algorithmen	8
2.3 Standart-GA	9
2.3.1 Initialisierung der Population	9
2.3.2 Bewertung aka Grade / Fittnesfunktion	10
2.3.3 Weiterentwickeln aka Evolve	10
Auswahl der Elternpaare aka Select Parents	11
Paarung aka Breed / Variation	11
2.3.4 Austausch/ Scheife / Exchange	12
2.4 Steady-State-GA	13
2.5 Künstliche Neuronale Netze	14
2.6 Aufbau eines Neurons	15
2.6.1 Eingang aka Input	15
2.6.2 Schwellwert aka bias	15
2.6.3 Aktivierungs Funktion	15
2.6.4 Ausgang aka Output	16
2.7 Struktureller Aufbau eines Neuronalen Netzes	16
2.7.1 Input Layer	16
2.7.2 Hidden Layer	16
2.7.3 Output Layer	16
2.8 Verlustfunktion aka lossfunktion	16
2.9 Optimierer alt Gradientenabstieg	16
2.10 Hyperparameter	17
2.11 Zusammenfassung	17
3 Stand der Forschung und Technik	18
3.1 Forschung	18
3.2 Anwendung	18
3.3 Zusammenfassung	18
4 Konzept	19
4.1 Anforderungsanalyse	19

4.2	Genetischer Algorithmus	19
4.2.1	Eltern auswahl	19
4.3	Zusammenfassung	19
5	Implementierung	20
5.1	Systemaufbau	20
5.2	Zusammenfassung	20
6	Evaluation und Tests	21
6.1	Einleitung	21
6.2	Testszzenarien	21
6.3	Evaluation	21
6.4	Ergebniss und Interpretation	21
6.5	Zusammenfassung	21
7	Zusammenfassung und Ausblick	22
7.1	Einleitung	22
7.2	Zusammenfassung	22
7.3	Bedeutung der Arbeit	22
7.4	Ausblick	22

Abbildungsverzeichnis

1	Beispiel einer Polulation mit 4 induviduen (Chromsomen) welche vier binäre Gene besitzen [1]	10
2	crossover anhand eines einfachen binären Chrms. Das erste zeigt eine 50/50 crossover. Das zweite zeigt eine Zufällige auswahl ders Gens.[1] . . .	12
3	Muation eines Genes um höhere vielfältigkeit zubekommen.[1]	12
4	Der unterschiedliche ablauf des a GA und des B steady state GA wird jeweils mit einem Ablaufschema verdeutlicht. Evolutionäre algorithmen s.129	13
5	Künstliches Neuronales Netz mit drei Schichten je drei Neuronen [1] . . .	14
6	Aufbau eines Neurons [1]	15

Abkürzungsverzeichnis

1 Einleitung

1.1 Motivation

Nach einer Studie der Charité aus dem Jahr 2015 sterben in Europa jährlich 23.000 Menschen an den Folgen einer Infektion mit multiresistenten Keimen. Die Tendenz ist dabei steigend. Hauptursache für die Ausbreitung dieser Keime, wie beispielsweise MRSA, ist eine mangelnde Hygiene der Angestellten in den Versorgungseinrichtungen beim Umgang mit den Patienten. Ziel des Projekts HEIKE ist es neue, technikgestützte Möglichkeiten zu entwickeln, welche die behandelnden Mitarbeiter im Krankenhausumfeld bei Maßnahmen am Patienten unterstützen. [2]

Um diese Automatischen System, meist Deep Learning Methoden, zu trainieren braucht es sehr große Datensätze und viele individuelle Hyperparameter. Momentan werden diese meist nach groben Ermessen des Entwicklers ausgewählt. Das Auswählen und Testen beansprucht sehr viel Zeit und Mühe.

1.2 Aufgabenstellung

Ziel der Arbeit ist es, zunächst die Optimierung von Hyperparametern zu vereinfachen. Dazu ist eine Automatisierter Trainings und Auswertvorgang nötig. Anschließend sollen die Hyperparameter mit Hilfe von Genetischen Algorithmen noch verbessert werden. Um schneller bessere Ergebnisse zu erhalten. Diese Ergebnisse sollen dann einer klassischen Grid Search gegenübergestellt werden.

Um dies zu vereinfachen soll ein Konzept geschaffen werden, welches die Vorgänge automatisiert und optimiert. Dabei geht es hauptsächlich um den Vorgang der Auswahl von Hyperparameter und die Auswahl der Dimension eines Künstlichen Neuronalen Netzes. Diese berechneten Werte sollen gespeichert und anschließend übersichtlich angezeigt werden. Wodurch sich die idealen Parameter herausbilden. Diese Ergebnisse sollen dem momentanen Ansatz gegenübergestellt werden.

(Mit diesem Ansatz kann die Dimensionierung eines Netzes einfacher umgesetzt werden.) Ein weiterer Anwendungsfall ist die Hyperparameterauswahl, mit Hilfe dieses Werkzeugs soll eine einfachere und bessere Auswahl der Hyperparameter erfolgen. Diese berechneten Werte sollen gespeichert und anschließend übersichtlich und intuitiv angezeigt werden. Wodurch sich die idealen Parameter herausbilden. Mit diesem Ansatz soll die Richtigkeit des Netzes erhöht werden, sodass es bessere Ergebnisse liefert. Dieses Werkzeug soll konzipiert und implementiert werden. Anschließend soll eine Evaluation und Auswertung über die mögliche Verbesserung durchgeführt werden.

1.3 Aufbau der Arbeit

Zunächst wird im zweiten Kapitel auf die verwendeten Grundlagen eingegangen. Zunächst wird im zweiten Kapitel auf die Grundlagen zu Genetischen Algorithmen und Künstlichen Neuronalen Netzen eingegangen.

Welche Algorithmen bei dieser Arbeit verwendet werden. Und mit welchen Künstlichen Neuronalen Netzen diese Optimierungs Algorithmen getestet werde. Außerdem wird in Abschnitt 3 auf den Momentanen Stand der Technik und Forschung eingegangen dort werden auch einige Anwendungsbeispiele der Genetischen Algorithmen genannt. Nun folgt in Kapitel 4 die Ausarbeitung des Konzeptes mit Erklärungen der einzelnen Ideen. Darauf aufbauend kommt Implementierung in Kapitel 5 in welcher mit Pseudocode erklärt wird wie die Arbeit umgesetzt wurde. Anschließend wird das implementierte System evaluiert und getestet. Zum Schluss in Kapitel 7 gibt es eine Zusammenfassung

2 Grundlagen

2.1 Optimierungsgrundlagen

Angenommen es soll ein Künstliches Neuronales Netz mit k Layern und L Neuronen zur Klassifizierung von einfachen handgeschriebenen Zahlen erstellt werden. Der Entwickler entscheidet sich für ein 3 Layern Netz mit jeweils 3 Neuronen. Nach dem Training hat es die Genauigkeit von 85 Prozent. Ist dies Akzeptabel? Kann man sagen, dass für $k = 3$ bzw. $j = 3$ die optimale Lösung gefunden wurde? Um dies zu beurteilen müssen viele Experimente durchgeführt werden. Die Frage ist, wie man die besten Werte für k und j finden um die Klassifizierung zu maximieren. Dieser Vorgang wird als Hyperparameter-Optimierung bezeichnet. Bei der Optimierung wird mit einem Initialwert gestartet dieser ist in den seltensten Fällen die exakte Lösung. Dieser Initialwert muss einige male verändert werden um auf ein Optimum zu kommen. Manchmal ist dieses Anpassen/optimieren so Komplex, dass es durch eine Funktion ersetzt werden muss.

2.2 Genetische Algorithmen

Genetische Algorithmen sind heuristische Suchansätze. Im wesentlichen zeichnet sie eine probabilistische Eltern selection als primären Suchoperator aus. Als weiteren Suchoperator kann noch auf die Mutation zurück gegrieffen werden, dieser garantiert eine Erreichbarkeit aller Punkte im Suchraum und erhält die Grunddiversität in der Population. (Genetische Algorithmen haben als theoretischen Grundlage die Schema-Theorie.) Es gibt zwei verschiedene Grundalgorithmen der Standard-GA (2.3). Dieser tauscht nach einer Generation die komplette Elternpopulation durch die Kinderpopulation aus. Im Gegensatz dazu gibt es den Steady-State-GA(2.4) welcher durch seine überlappende Population auszeichnet. Nun werden wir uns genauer den Standard-GA 2.3 und seine einzelnen Schritte genauer betrachten.

2.3 Standart-GA

Der Standart-GA ist folgende Schritte aufgeteilt und wird anschließend genau besprochen: Schritt 1 Initalisieren einer Population. Schritt 2 Fitness brechnen. Schritt 3 Weiterentwicklen hier werden zuerst die passenden Eltern ausgesucht und anschließend mit Crossover und Mutation die Kinder Population erstellt. Schritt 4 Elternpopulation durch die neue Kinderpopulation ausgetauscht. Hier noch einamal als Pseudocode.

Algorithm 1 Basic Genetic Algorithm

```
1: initialize population
2: repeat
3: repeat
4: fitness computation
5: parent selection
6: breed
7: crossover
8: mutation
9: until population complete
10: selection of parental population
11: until termination condition
```

2.3.1 Initizialisierung der Population

Der klassische Genetische Algorithmus basiert auf einer Reihe von Kandidatenlösungen. Die gröÙe der Populaton ist somit auch die Anzahl der Lösungen. Jede Lösung kann als einzelnes Individuum gesehen werden und wird durch ein Chromsomenstang oder kurz durch seine Chromsomen repräsentiert. Ein Chromsomen besteht wiederum aus vielen Genen welche die Parameter repräsentieren. Es gibt verschiedene Möglichkeiten diese Gene darzustellen, wie z.B. binär oder Dezimal. Um die Grundlagen nahe der später Folgenden Konzepts zuhalten wird alles per Dezimal Genen erklärt. Figure 4 veranschaulicht den Aufbau noch einmal.

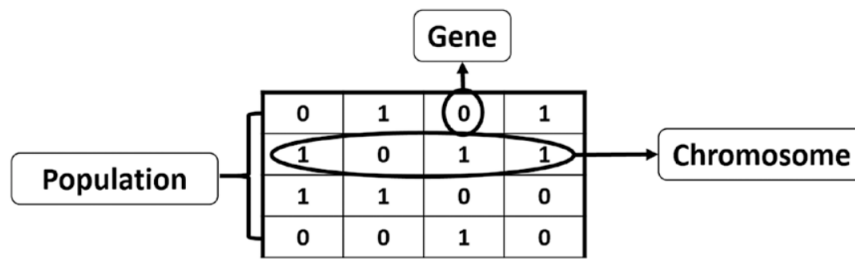


Abbildung 1: Beispiel einer Population mit 4 Individuen (Chromosomen) welche vier binäre Gene besitzen [1]

Diese anfangs Population wird meist Zufällig initialisiert. Hierdurch ist es möglich die größte Abdeckung des Suchraums zu gewähren. Dadurch besitzt die erste Generation eine sehr geringe Fitness, die sich aber im Lauf des Trainings verbessert. Durch Selektion werden die nicht unnötigen/Contra-produktiven Individuen aussortiert. Bevor dies passiert muss aber erst die Bewertung durchgeführt werden.

2.3.2 Bewertung aka Grade / Fitnessfunktion

Nun besteht die erste Generation (Generation 0) aus einer Population mit völlig zufälligen Individuen. Diese werden anhand einer für seine Anwendung speziellen Fitnessfunktion bewertet. Dabei werden nicht einzelnen Gene bewertet sondern das ganze Genom/Chromosom/Individuum. Es wird also nicht berücksichtigt welches Gene sich positiv bzw. negativ auswirken. Als Rückgabewert gibt die Fitnessfunktion uns einen Fitnesswert, dabei steht ein höherer Fitnesswert stets für eine höhere Qualität an Individuum. Da nun alle Individuen der Population bewertet wurden kann eine neue Generation erstellt werden.

2.3.3 Weiterentwickeln aka Evolve

Dazu werden aus zwei Elternpaaren ein neues Kind erstellt. Um die Elternpaare auszuwählen gibt es verschiedene Optionen. Da nun die Eltern fest stehen, wird per Crossover aus den beiden Elternpaaren oder aus dem Elternpool ein neues Kind generiert. Um bei den Genen eine höhere Diversität zu gelangen werden die Kindergene noch mit einer Mutation versehen. Somit kann man einen höheren Suchraum (Abdeckungsgrad) abdecken. Nach dem eine neue Kindergeneration erstellt wurde wird der ganze Vorgang so lange wiederholt bis die geforderte Fitness erreicht wurde.

Auswahl der Elternpaare aka Select Parents Um eine konvergenz richtung Optimalen Maximum bzw. Minimum zu schaffen werden die besten Elternteile der gerade bewerteten Generation ausgewählt. Für die Auswahl gibt es verschiedene Ansätze, die bedeutensten werden genannt und erläutert.

- **Auswahl proportional zu Fitness eng. Fitness Proportional Selection(FPS)**, hierbei spielt die im vorherigen Schritt berechnete Fitness eine große Rolle. Die Eltern werden nach dieser Fitness proportional als Elternteil ausgewählt und zum Elternpool hinzugefügt.
- **Ranking Selektion**, diese Selektion wurde als Verbesserung der Fitness Proportional Selection entwickelt. Dabei werden die Eltern nicht direkt nach ihrer Fitness ausgewählt sondern über die Fitness im Vergleich zur gesamten Population. Sprich sie werden in einer Rangliste aufgestellt wobei das beste Individuum den Rang $y-1$ belegt und das schlechteste den Rang 0. Dieses Ranking kann in vielen Varianten vorgenommen werden, Linear oder exponentiell umgesetzt werden.
- **Turnier selektion**, in diesem Verfahren werden k Individuen der Population ausgewählt. Diese k Individuen treten dann wie in einem Turnier gegeneinander an. Das Individuum mit dem besten Fitnesswert ein Elternteil ausgewählt. Hierbei wird auf den Elternpool verzichtet und direkt ein Kind aus zwei Gewinnern erstellt. Eingesetzt wird dies bei eher kleinen Populationen.
- **Comma selection**, Genetic algorithm essentials S.17
- **Plus selection**, Genetic algorithm essentials S.17

Nun wurden die Eltern ausgewählt aus der Eltern-Generation kann jetzt eine neue Kinder-Generation erstellt werden.

Paarung aka Breed / Variation Aus dem Elternpool/ Paarungspool werden nun Nachkommen(Kinder) geschaffen. Allein durch die Paarung(Crossover) von qualitativ hochwertigen Individuen wird erwartet, dass die Nachkommen eine bessere Qualität besitzen als die ihrer Eltern. Dennoch kann es zu negativen man nur die Eigenschaften der Eltern übernimmt kann es sogar dazu kommen dass negative Eigenschaften mit übernommen werden. Da dies natürlich nicht gewollt ist gibt es eine einfache Verbesserungsmöglichkeit. Die Mutation, hier wird jedes Gen noch einmal mit einer zufälligen Mutation versehen welches ähnliche aber andere Lösungen hervorbringt. Nun gehen wir noch einmal genauer auf Operation Crossover und Mutation ein.

- **Crossover**, werden die Chromosomenstränge der Kinder Individuen bestimmt. Beim Crossover gibt es mehrere Varianten eine ist die Two-Point-Crossover bei der 50

Prozent des ersten Elternteils und 50 Prozent des zweiten elternteils verwendet, wie es im Oberenteil der Abbildung 2 zu sehen ist. Ein zweiter Ansatz ist die Uniform-Crossover, hier werden die Gene ganz zufällig und unabhängig von einander ausgewählt, wie im Unterteil der Abbildung dies nennt man 2 zu sehen ist.

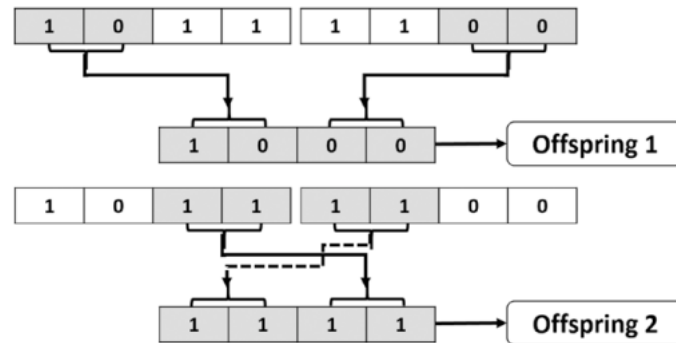


Figure 4-3. Single-point crossover between two parents to produce two offspring

Abbildung 2: crossover anhand eines einfachen binären Chroms. Das erste zeigt eine 50/50 crossover. Das zweite zeigt eine Zufällige auswahl ders Gens.[1]

- **Mutation**, hierbei wird jedes gens des Individium zufällig mit einer zufälligen Mutation versehen. Durch diese Mutation wird eine neue Inforamtion/Lösung in die nachfolgende Generation übergeben. Durch diese Mutation ist es möglich einen größeren Suchraum abzudenken. Es ist auch möglich die Werte genauer anzupassen um so auf die Optimalste Lösung zu kommen.

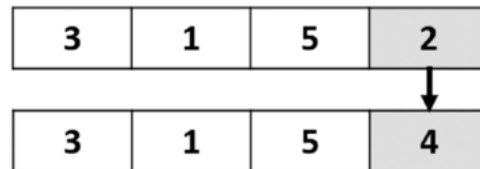


Figure 4-5. Uniform mutation for a solution with more than two values for its genes

Abbildung 3: Muation eines Genes um höhere vielfältigkeit zubekommen.[1]

2.3.4 Austausch/ Scheife / Exchange

Die neue Generation aus Kindern tauscht nun die alte Generation aus. Nun folgen die gleichen Schritte: Grade, Slection, crossover und mutation. Diese Schleife wird so lange

durchgeführt bis die Populationsfitness das zuvor festgelegte Maximum erreicht. Wenn dies geschieht gibt es viele Lösungen welche alle sehr ähnlich sein sollten. Aus dieser kann dann das beste Individuum ausgesucht werden und als beste Lösung eingesetzt werden.

2.4 Steady-State-GA

vt weg lassen ?

Dieser Ansatz unterscheidet sich von dem Standard GA vor allem in der überlappenden Population. Denn hier wird nicht die gesamte Population nach einer Generation ausgetauscht sondern nur das schlechteste Individuum. Auch hierbei gibt es die genannten Elternselectionen, Crossover und Mutation als Auswahl.

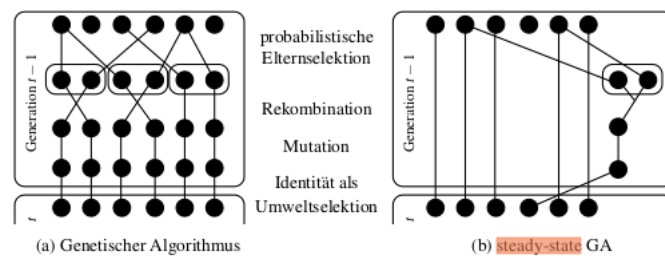


Bild 4.1. Der unterschiedliche Ablauf des (a) GA und des (b) steady state GA wird jeweils mit einem Ablaufschema verdeutlicht.

Abbildung 4: Der unterschiedliche ablauf des a GA und des B steady state GA wird jeweils mit einem Ablaufschema verdeutlicht. Evolutionäre algorithmen s.129

2.5 Künstliche Neuronale Netze

Künstliche Neuronale Netze sind dem natürlichen Vorbild der neuronalen Netze im Gehirn nachempfunden. Beide Netze setzen sich aus einzelnen Neuronen zusammen, welche mit einander verbunden sind. Wie man in Figure 5 sieht, ist jede Schicht aus einzelnen Neuronen aufgebaut, welche mit den Neuronen der nächsten Schicht verbunden sind, diese Verbindungen repräsentieren die Gewichte, über diese kann einem Netz verschiedene Zusammenhänge von Input und Output antrainiert bzw. angelernt werden.

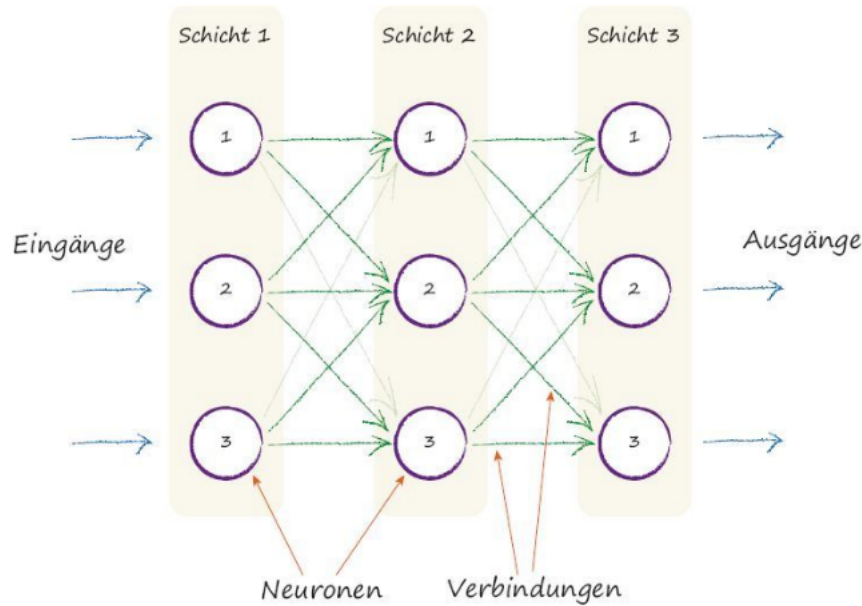


Abbildung 5: Künstliches Neuronales Netz mit drei Schichten je drei Neuronen [1]

Im folgenden Kapitel wird zu erst der Aufbau eines Neurons erklärt. Anschließend wird auf den Stukturellen aufbau eines Künstlichen Neuronalen Netzes nähergebrach. Zum Schluss werden noch wichtige eigenschaften wie die Verlustfunktion und der Gradienten abstieg eingegangen sowie auf die Hyperparameter welche für die nachfolgenden kapitel wichtig sind.

Grundlagen aus dem buch ArificalNeuroalNetworks s.11

2.6 Aufbau eines Neurons

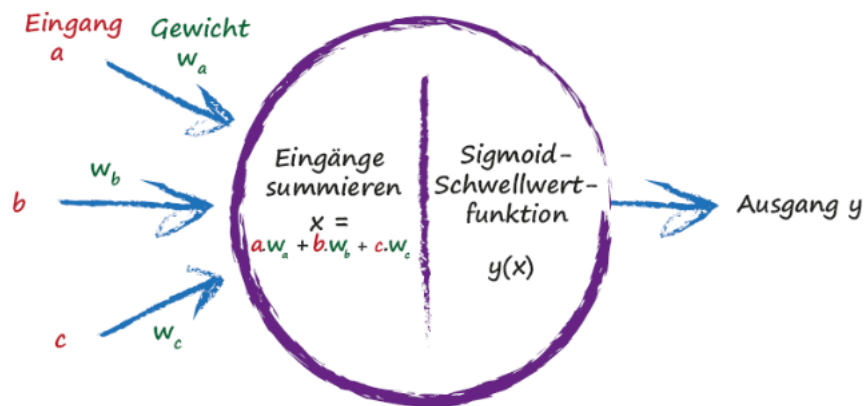


Abbildung 6: Aufbau eines Neurons [1]

2.6.1 Eingang aka Input

Bei dem Input handelt es sich um einfache xxxFloatwert dieser wird mit den einzelnen Gewichten verrechnet. Ein Neuron hat meist mehrere Eingangsgrößen, welche alle zusammen mit den Gewichten aufsummiert werden. Diese Werte werden zufällig initialisiert und per Training verbessert, somit handelt es sich um einen angelernten Werte, welche durch die Backproagation(Fehlerrückführung) verbessert werden.

2.6.2 Schwellwert aka bias

Auf dieses Aufsummiertes Ergebniss wird anschließend ein Bias gerechnet, dieser führt zu einem besseren Verhalten beim Trainieren. Bei diesen Werten handelt es sich um angelernte Werte, die per Backpropagation verbessert werden und die Flexibitlität der Netze erhöht.

2.6.3 Aktivierungs Funktion

Die Aktivierungsfunktion kann man sich als Schwellwert vorstellen, ab wann das Neuron den Input weiter gibt. Es gibt verschiedene Funktionen, um diesen Schwellwert zu definieren. Je nach Aufgabe des Neuronalen Netze werden andere Aktivierungsfunktionen verwendet. Bei Klassifizierungen werden heute meist ReLu-Layer oder ein Weakly-ReLu

Layer benutzt, diese verhindern das Vanishing- bzw. Exploding- gradientproblem beim Trainieren.

2.6.4 Ausgang aka Output

Wenn der Schwellwert überschritten wird, wird am Output durchgeschaltet. Dieser Output kann entweder mit einer neuen Schicht Neuronen verbunden sein oder direkt als Ausgang gesehen werden. Über welchen man anhand von xxxVariablenwerten/Kommawerten die Von Input nach Output nennt sich ein Single-Forward-Pass. Wie hier beschrieben wird, kann ein Netz verschieden viele Layer besitzen mit verschiedenen Anzahlen von Neuronen.

2.7 Struktureller Aufbau eines Neuronalen Netzes

Wie kann ich Neuronen zusammenbauen usw.

2.7.1 Input Layer

2.7.2 Hidden Layer

2.7.3 Output Layer

2.8 Verlustfunktion aka lossfunktion

Die Verlustfunktion stellt ein ausgesuchtes Maß der Diskrepanz zwischen den beobachteten und den vorhergesagten Daten dar. Sie bestimmt die Leistungsfähigkeit des neuronalen Netzes während des Trainings und der Ausführung. Ziel ist es, im laufenden Prozess der Modellanpassung, die Verlustfunktion zu minimieren.

2.9 Optimierer alt Gradientenabstieg

Um die Fehlerfunktion zu minimieren wird als Werkzeug der Gradienten Abstieg benutzt. Diese ist nur möglich da ein Künstliches Neuronales Netz aus verketteten differenzierbaren Gewichte der Neuronen(Tensoroperationen) aufgebaut ist, die es erlauben durch anwendung der Kettenregel die Gradientenfunktion zu finden, die den aktuellen Parametern

des Datenstapels werte des Gradienten zuordnet. Es gibt auch hier verschiedene Ansätze von Optimierern, welche die genauen Regeln wie der Gradient der Verlustfunktion zu Aktualisierung der Parameter verwendet wird hier könnte Beispielsweise den RMSProp-Optimierer, der die Trägheit des Gradientenabstiegsverfahren berücksichtigt.

Seite 83 - Deep Learning chollet

2.10 Hyperparameter

Als Hyperparameter werden, in Bezug auf KNN's, meist die Anfangsbedingungen bezeichnet. Für diese Hyperparameter gelten keine universellen Werte sondern müssen je nach Daten und Funktion bzw. künstliches Neuronales Netz speziell angepasst und verändert werden. Deshalb gibt es nur einige Regeln und grobe Abschätzungen in welchen Grenzen sich diese Hyperparameter befinden. Zu diesen Hyperparametern gehören folgende:

- **Learningrate**, blablaa
- **Dropout**
- **Lossfunktion**
- **Optimizer**
- **Model Architektur**

2.11 Zusammenfassung

3 Stand der Forschung und Technik

3.1 Forschung

3.2 Anwendung

NEAT, HyperNEAT, GLEN

3.3 Zusammenfassung

4 Konzept

4.1 Anforderungsanalyse

4.2 Genetischer Algorithmus

4.2.1 Eltern auswahl

Ranking Selction mit 50 übernahme als Eltern. Zusätzlich werden die besten 4 Eltern ganz übernommen. das die Generationen sich nicht zu sehr ähnlich sind wird hoh mutation bei den Kindern. Zusätzlich werden bei jeder neuen Generation immer ein paar neu und damit zufällige Individuen hinzu gefügt.

Best 50 prozent, heißt aus der oberen hälfte der alten Generation werden alle Individuen dem Elternpool hinzugefügt. Aus welchen dann zufällig die einzelnen Elternteile ausgewählt werden. Es mssen natürlich nicht immer 50 prozent sein, es kann sich auch um einen anderen Prozentsatz handeln.

4.3 Zusammenfassung

5 Implementierung

5.1 Systemaufbau

5.2 Zusammenfassung

6 Evaluation und Tests

6.1 Einleitung

6.2 Testszenarien

6.3 Evaluation

6.4 Ergebniss und Interpretation

6.5 Zusammenfassung

7 Zusammenfassung und Ausblick

7.1 Einleitung

7.2 Zusammenfassung

7.3 Bedeutung der Arbeit

7.4 Ausblick

Literatur

- [1] T. Rashid. *Neuronale Netze selbst programmieren: Ein verständlicher Einstieg mit Python*. Animals. O'Reilly, 2017.
- [2] Projekt Heike.