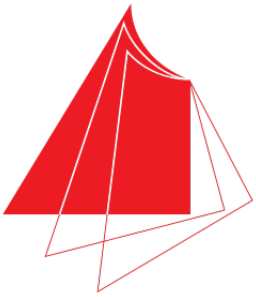

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

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
Martrikelnummer: 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 inteligent 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

FZI Forschungszentrum Informatik | Embedded Systems and Sensors Engineering (ESS)
Lukas Kohout | kohout@fzi.de | Tel. +49 721 9654 - 167
Mehr Informationen unter www.fzi.de/karriere

Inhaltsverzeichnis

Abbildungsverzeichnis

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, technikgestütze 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 Automatiesierter Trainings und Auswerte vorgang nötig. Anschließend sollen die Hyperparameter mit hilfe von Genetischen Algorithmen noch verbessert werden. Um schneller bessere Ergebnisse zu erhalten. Diese Ergebnisse sollen dann einer klassichen 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 anzeigt 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 weiter Anwendungsfall ist die Hyperparameterauswahl, mit hilfe dieses Werkzeugs soll eine einfachere und bessere Auswahl der Hyperparameter erfolgen. Diese berechneten Werte sollen gespeicher und anschließend übersichtlich und intuitiv anzeigt werden. Wodurch sich die idealen Parameter herausbilden. Mit diesem Ansatz soll die Richtigkeit des Netzes erhöht werden, sodass es bessere Ergebnisse liefesert. Dieses Werkzeug soll Konzeptioniert und Implementiert werden. Anschließend soll eine Evaluation und Auswertung über die mögliche Verbessung durchgeführt werden.

1.3 Aufbau der Arbeit

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

Welche Algorythmen bei dieser Arbeit verwendet werden. Und mit welchen Künstlichen Neuronalen Netzen diese Optimierungs Algorithmen getestet werde. Außderdem wird in Abschnitt 3 auf den Momentanen Stand der Technik und Forschung eingegangen dort werden auch einige Anwendungbeispiele 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 psyodocode 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 Neuronales Netz mit k Layern und jNeuronen zur Klassifizierung von einfachen Handgeschrien-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 das für k 3 bzw. j 3 die optimale Lösung ist? Um dies zu beurteilen müssen viele Experimente durchgeführt werden. Die Frage ist, wie man den die Besten werte für k und j findet, der die Klassifizierung maximiert. Dies wird als Hyperparameter Optimierung bezeichnet. Bei der Optimisierung wird mit einem Inizialwert gestartet dieser ist meist nicht der beste. Dieser Inizialwert muss einige male geändert werden um auf einen Optimum zu kommen. Manchmal ist dieses Anpassen/optimieren so Komplex, dass es durch eine Funktion ersetz werden muss.

2.2 Genetische Algorithmen

Genetische Algorithmen sind heuristische Suchansätze. Im wesentliche zeichnet sie eine probilistische Eltern selection als primären Suchoperator aus. Als weitern suchoperator kann der Ga noch auf die Mutation zurückgreifen, dieser grantiert eine Erreichbarkeit aller Punkte im suchraum und erhält so die Grunddiversität in der Population. Genetische Algorithmen haben als theoretischen Grundlage die Schama-Therorie. Es gibt zwei verschiedene Grundalgorihmen der Standart-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 Standart-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 Inizialisieren einer Population. Schritt 2 Fittness brechnen. Schritt 3 weiterentwicken 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 Initzialisierung der Population

Der Klassiche Genetische Algorithmus bassiert auf einer Reihe von Kandidatenlösungen. Dee größe der Populaton ist auch die Anzahl der Lösungen. Jede Lösung kann als einzelnes Induvidum gesehen werden und wird durch ein Chromsone representiert. Es gibt verschiedene möglichkeiten diese Gene dazustellen, wie z.B binär oder Dezimal. Figure 4 veranschaulicht ein Beispiel, wie eine Population aus vier Individuen(chromosomes)mit je einem Chromosom. Ein Chromso, besitzt wiederum vier gene. Jedes dieser Gene ist durch eine binären zahl repräsentiert.

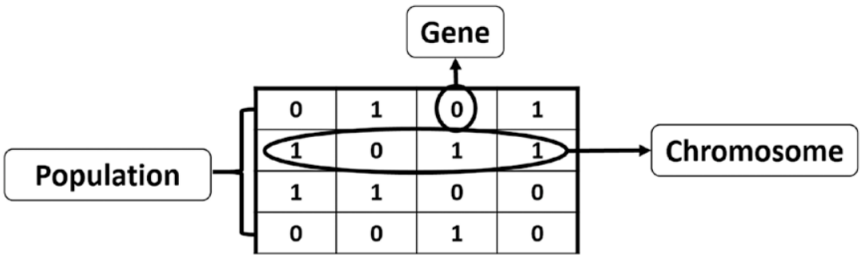

Abbildung 1: Beispiel einer Polulation mit 4 induviduen (Chromsomen) welche vier binäre Gene besitzen [1]

Diese anfangs Poulation wird meißt Zufällig inizialisiert. Hier durch ist es möglich die größte abdeckung des Suchsraums zu gewähren. Dadurch besitzt die erste Generation eine sehr geringe Fittness dies verbessert sich aber im Lauf des Trainings. Durch Selection werde die nicht unnötigen/Contra-produktiven Individumen aussotiert. Bevor dies passiert muss aber erst die Bewertung durchgeführt werden.

2.3.2 Bewertung aka Grade / Fittnesfunktion

Nun besteht die erste Generation(Generation 0) aus einer Population mit völlig zufälligen Induviduen. Diese werden anhand einer für sine anwendung speizellen Fitnessfunktion bewerte. Dabei werden nicht einzelnen Gene bewertet sondern das ganze Genom/Chromoson/Idividum. Es wird also nicht berücksichtigt welches Gene sich positiv bzw. negativ auswirklen. Als Rückgabewert gibt die Fittnesfunktion uns einen Fittneswert, dabei steht ein höherer Fittnesswert stehts für eine höher Qualität an Individum. 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 auszusuchen 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 Kind generation erstellt wurde wird der ganze vorgang so lange wiederhollt bis die geforderte Fintess ereicht wurde.

Auswahl der Elternpaare aka Select Parents Um eine convergenz richtung Optimalem Maximum bzw. Minmum zu schaffen werden die besten Elternteile der geraden bewertenen Generation ausgewählt. Für die Auswahl gibt es verschiedene Ansätze, die bedeutesten werden genannt und erläutert.

- **Auswahl proportional zu Fittnes eng. Fitness Proportonal Selction(FPS)**, hierbei spielt die im vohrigen Schritt berechnete Fintess eine große Rolle. Die Eltern werden nach dieser Fittness proportional als Elternteil ausgewählt und zum Elternpool hinzugefügt.

- **Ranking Selektion**, diese Selktion wurde als Verbesserung der Fitness Proportonal Selection entwickelt. Dabei werden die Eltern nicht direkt nach ihrer Fittness ausgewählt sondern über die Fitness im vergleich zur gesamt Population. Sprich sie werden in einer Rangliste aufgestellt wobei das beste Individum den Rand y-1 belegt und das schlechteste den Rang 0. Dieses Ranking kann in vielen varianten vorgenommen werden, Linear oder expotenziel umgesetzt werden.

- **Tunier selektion**, in diesem Verfahren werden k induviduen der Population ausgewählt. Diese k Induviuen treten dann wie in einem Tunier gegeneinander an. Das Individum mit dem Besten Fittneswert 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 selction**, Genetic algorithm essentails s.17

- **Pluss selection**, Genetic algorithm essentails s.17

Nun wurden die Eltern ausgewählt aus der Elterngeneration kann jetzt eine neue Kindergeneration erstelt werden.

Paarung aka Breed / Variation Aus dem Elternpool/paarungspool werden nun Nachkommen(Kinder) geschaffen. Alleine 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 negative man nur die Eigenschaften der Eltern übernimmt kann es sogar dazu kommen das negative eigenschaften mit übernommen werden. Da dies natürlich nicht gewollt ist gibt es eine einfach Verbesserungs möglichkeit. Die Muation, hier wird jedes Gen noch einmal mit einer zufälligen Muation versehen welches ähnliche aber andere Lösungen hervorbringt. Nun gehen wir noch einmal genauer auf Operation Chrossover und Muation ein.

- **Crossover**, werden die Chromostränge der Kinder Individuen bestimmt. Beim Crossover gibt es mehrer 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 unabhänig von einander ausgewählt, wie im Unterenteil der Abbildung dies nennt man 2 zu sehen ist.

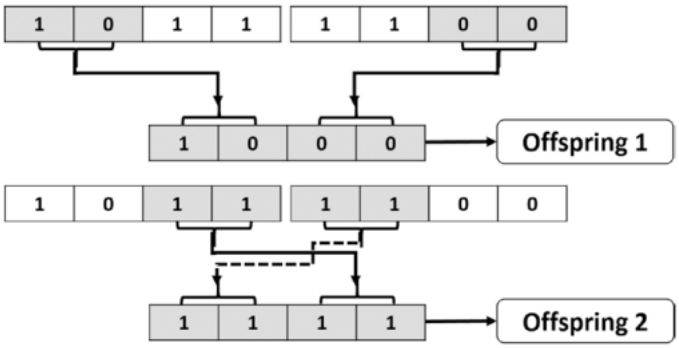

Figure 4-3. *Single-point crossover between two parents to produce two offspring*

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 Indivium 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.

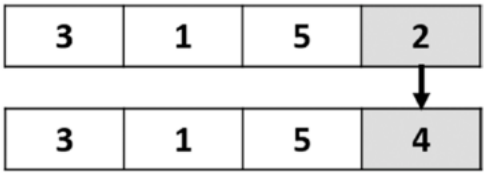

Figure 4-5. *Uniform mutation for a solution with more than two values for its genes*

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 durchegführt bis die Populationsfittnes das zuvor festgelegte Maximum erreicht. Wenn dies geschied gibt es viele Lösungen welche alle sehr ähnlich sein sollten. Aus dieser kann dann das beste Individum ausgesucht werden und als beste Lösung eingesetz werden.

2.4 Steady-State-GA

Dieser Ansatz unterscheidet sich von dem Standart GA vorallem in der überlappenden Population. Denn hier wir nicht die gesamte Population nach einer generation ausgetauscht sondern nur das schlechteste Indivdum. Auch hierbei gibt es die genannten Elternselectionen, Crossover und Mutation als Auswahl.

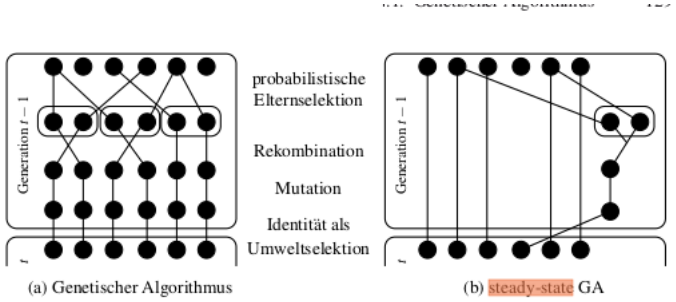

Bild 4.1. Der unterschiedliche Ablauf des (a) GA und des (b) steady state GA wird jeweils mit einem Ablaufschema verdeutlicht.

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

2.5 Künstliche Neuronale Netze

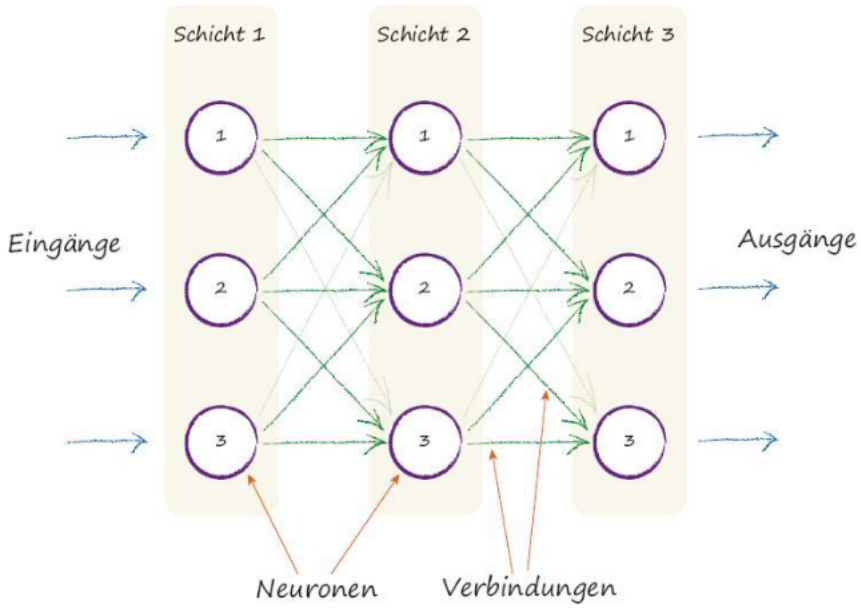

Abbildung 5: Künstliches Neuronales Netz mit drei Schichten je drei Neuronen [1]

Hier zu sehen ist ein Künstliches Neuronales Netz mit drei Schichten 5. Dies wurde dem natürlichen Vorbild der neuronalen Netze im Gehirn nach empfunden. Die Kreise nennt man Neuronen oder auch Perseptron, mehrere Neuronen zusammen ergeben eine Schicht oder auch Layer genannt. Die Verbindungen repräsentieren die Gewichte, über diese kann einem Netz verschiedene Zusammenhänge von Input und Output antrainiert bzw. angelernt werden. Zum Training werden viele Daten benötigt, aus welchen das Netz „Lernt“. Dafür ist es wichtig, viele aufbereitete Daten zu besitzen, denn diese Netze brauchen viele Trainingsiterationen, bis das gewünschte Ergebnis zustande kommt. Ein Neuron besteht aus Eingängen, Gewichten und einer Aktivierungfunktion sowie einem Ausgang. Die Vernetzung mehrerer Neuronenschichten lässt ein Neuronales Netz entstehen.

2.6 Aufbau eines Neurons

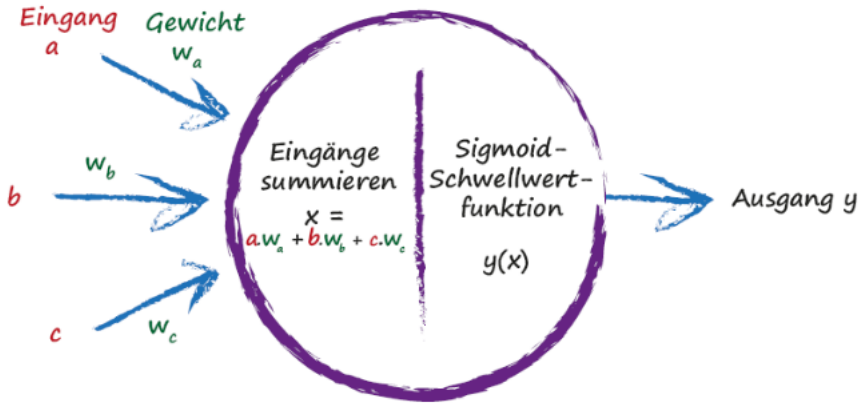

Abbildung 6: Aufbau eines Neurons [1]

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 Offset 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 nen Schicht Neronen verbundne sein oder direkt als Ausgang gesehen werden. Über welchen man anhand von xxxVariabelenwerten/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 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.

2.8 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 duch 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 Aktualisierund der Parameter verwendet wird hier könnte Beispielweise den RMSProp-Optimierer, der die trägheit des Gradientenabstiegsverfahren berücksichtet.

Seite 83 - Deep Learning chollet

2.9 Hyperparameter

Als Hyperparameter werden, in Bezug auf KNN's, meist die Anfangsbedingungen bezeichnet. Somit handelt es sich um die Learnrate (eng. Learningrate), der Abdeckunggrad(eng. Dropout), die verlustfunktion oder auch der Optimizer. In selten fällen kann

man die Modelachitektur auch als Hyperparameter bezeichen. Für diese Hyperparameter gelten keine universellen Werte, sondern müssen je nach Daten und Funktion(oder KNN), speziell angepasst und verändert werden. Deshalb gibt es nur einige Regeln und grobe abschätzungen in welchem grenzen sich diese Hyperparameter befinden.

2.10 Zusammenfassung

3 Stand der Technik

3.1 Einleitung

3.2 Forschung

3.3 Anwendungen

3.4 Zusammenfassung

4 Konzept

4.1 Einleitung

4.2 Anforderungsanalyse

4.3 Genetischer Algorithmus

4.3.1 Eltern auswahl

Ranking Selction mit 50 übernahme als Eltern. Zusätzlich werden die besten 4 Eltern ganz übernommen. das die Generationen sich nicht zu sehr ähnlichsind 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 Induviduen 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.

4.4 Zusammenfassung

5 Implementierung

5.1 Einleitung

5.2 Systemaufbau

5.3 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.