Exposé

Optimierung neuronaler Netze durch memetische Algorithmen

[redacted]

3. November 2019

1 Theoretischer Hintergrund

1.1 Künstliche neuronale Netze

Künstliche neuronale Netze (KNN) sind Berechnungssysteme, die biologischen neuronalen Netzen wie dem Gehirn nachempfunden sind. Sie können anhand von gegebenen Daten lernen, eine bestimmte Aufgabe zu erfüllen und diese Fähigkeit dann auf unbekannte Daten übertragen. Sie sind aus mehreren miteinander verbundenen Knotenpunkten aufgebaut, die künstliche Neuronen genannt werden. Die Verbindungen zwischen den Neuronen können Signale an andere Neuronen weiterleiten und sind vergleichbar mit biologischen Synapsen. Wenn ein Neuron ein Signal erhält, verarbeitet es dieses und sendet dann ein neues Signal aus(?).

Das Ziel beim Training eines solchen Netzes ist es, die zugrundeliegende Struktur eines Datensatzes zu erkennen und diese auf unbekannte Daten zu übertragen. Wird ein Netz zu kurz trainiert, oder sind zu wenig Daten vorhanden kann die Struktur nicht erkannt werden und es entsteht ein systematischer Fehler, auch Bias genannt (Underfitting). Wird das Netz zu lange anhand der selben Daten trainiert, beginnt es die Zufälligen Fehler der Daten Mitzulernen (Overfitting) - die Varianz der Vorhersagen nimmt zu. Diese Austauschbeziehung zwischen Bias und Varianz ist das grundlegende Problem des maschinellen Lernens (Quelle Bias-Variance-Tradeoff).

In der Praxis können die Neuronen eines KNNs reelle Werte speichern, die Aktivierung genannt werden. Neuronale Netze sind oft in Schichten strukturiert (siehe Abb. ??). Es gibt eine Eingabeschicht, deren Werte von außen festgelegt werden. Von der Eingabeschicht werden die Werte über Verbindungen zur nächsten Schicht weitergeleitet. Die Verbindungen haben in der Regel eine Wichtung, die ihrer Stärke entspricht und mit der das Signal multipliziert wird. Jedes Neuron in der nächsten Schicht bildet nun die Summe aller Eingabewerte plus einen festen Wert, der Bias genannt wird. Die Aktivierung des Neurons berechnet sich dann als nicht-lineare Funktion der Summe. Diese Funktion heißt Aktivierungsfunktion ϕ des Neurons. Die Aktivierung eines Neurons der ersten Zwischenschicht wird also mit folgender Formel berechnet

$$a_j^1 = \phi\left(\sum_i a_i^0 w_{ij}^1 + b_j^1\right) \tag{1}$$

Hierbei ist a_j^1 die Aktivierung des j-ten Neurons der ersten Zwischenschicht, b_j^1 dessen Bias, a_i^0 die Aktivierung des i-ten Neurons der Eingabeschicht und w_{ij}^1 die Wichtung der Verbindung dieser Neuronen. Fasst man die Aktivierungen einer Schicht zu einem Vektor und die Wichtungen zwischen zwei Schichten zu einer Matrix zusammen, dann lässt sich Gleichung (1) wie folgt schreiben (Die Indexe zur Kennzeichnung der Schicht stehen jetzt unten):

$$\vec{a}_1 = \phi\left(W_1 \vec{a}_0 + \vec{b}_1\right). \tag{2}$$

Der Bias wird dabei meistens durch ein dauerfeuerndes Neuron realisiert, dessen Stärke

durch die jeweilige Wichtung bestimmt wird. Nach mehreren Verarbeitungsschritten durch die vorhandenen Schichten erreichen die Signale die Ausgabeneuronen. Diese repräsentieren den Lösungsvorschlag des Netzes. Die simpelste Form des KNNs lässt keine rückläufigen Verbindungen zu und wird deshalb Feed-Forward-Netz genannt. Solche KNNs können mit einer endlichen Anzahl von Neuronen jede beliebige stetige Funktion approximieren (Universal Approximation Theorem). Manche Probleme erfordern jedoch die Vorhersage einer Zeitreihe, deren Wert von ihrer Vergangenheit abhängt. Für diesen Problemtyp sind KNNs mit rückwärtigen Verbindungen nötig (Abb. ??). Sie heißen recurrent neural networks (RNN).

In beiden Fällen werden während des Lernprozesses die Wichtungen angepasst, um die Leistungsfähigkeit des Netzes zu verbessern. Zu diesem Zweck wird eine sogenannte Fehlerfunktion L definiert, die die Abweichung zwischen den berechneten (y) und den gewünschten Werten ($\hat{y}$) beschreibt. Das Ziel ist, diese Funktion durch Anpassen der Wichtungen zu minimieren. Gängige Fehlerfunktionen sind die mittlere quadratische Abweichung (mean squared error)

$$L(W) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3}$$

und die Kreuz-Entropie (cross entropy)

$$L(W) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i). \tag{4}$$

Dabei werden jeweils mehrere Datensätze, im Idealfall der gesamte Datenbestand betrachtet, wodurch die Fehlerfunktion nur noch eine Funktion der Wichtungen ist.

1.2 Gradient descent und Backpropagation

Seit den 1950er Jahren Forscht man in der wissenschaftlichen Disziplin Neuroinformatik, die sich mit künstlichen neuronalen Netzen beschäftigt. Vor allem im letzen Jahrzehnt sind in diesem Gebiet beachtliche Erfolge zu verzeichnen. Da mich das Thema sehr fasziniert, möchte ich mich in dieser Seminararbeit ebenfalls mit künstlichen neuronalen Netzen befassen.(?)

Die Leistungsfähigkeit eines neuronalen Netzes wird durch viele Faktoren bestimmt, darunter vor allem der Aufbau und die Stärke der Verbindungen zwischen den Neuronen. Um diese Parameter an das Problem anzupassen, wird ein Lernalgorithmus verwendet. Bei konventionellen Lernmethoden wird die Struktur des Netzes vorgegeben und lediglich die Stärke der Verbindungen zwischen den Neuronen angepasst. Doch dabei wird eine wichtige Möglichkeit der Optimierung außen vor gelassen. Aus diesem Grund möchte ich in meiner Seminararbeit Lernalgorithmen betrachten, die auch eine Anpassung der Netzwerkarchitektur erlauben.

Besonders die sogenannten evolutionären Algorithmen (EAs) sind interessant, da sie, wie auch die künstlichen neuronalen Netze selbst, an die Natur angelehnt sind. Dies hat sich in der Vergangenheit oft als sinnvoll erwiesen. Ihr Konzept wird in einem Artikel von G.

Rudolph und H.-P. Schwefel erläutert[1]. EAs liefern bessere Ergebnisse bei der Suche nach der optimalen Lösung für das jeweilige Problem, als herkömmliche Lernalgorithmen, jedoch erfordern sie auch eine höhere Rechenleistung. Aus diesem Grund kombiniert man die globale Suchalgorithmen des EA mit lokalen Suchalgorithmen wie beispielsweise Backpropagation[2]. Diese Kombination nennt man memetischer[3] Algorithmus (MA). Dies erwies sich in viele Fällen als effizienter als die alleinige Anwendung einer der beiden Optionen [4] [5].

Es gibt viele Publikationen, die die Optimierung solcher Algorithmen für einzelne Probleme dokumentieren und auch einige, die sich mit ihren grundlegenden Eigenschaften beschäftigen. Insbesondere die Auswirkung von Parametern wie der Populationsgröße, der Mutationsrate und der Wahl des lokalen Suchverfahrens auf die Leistungsfähigkeit der betrachteten Algorithmen wurde anhand von Referenzproblemen untersucht[6]. Zwar wurden MAs auch zur Optimierung der Architektur neuronaler Netze angewendet[7], diese Forschung war jedoch nicht auf den Anstieg des Trainingsaufwands durch die Einbeziehung der Architektur fokussiert. Auch sonst scheint es keine genauen Daten dazu zu geben. Deshalb wird sich meine Seminararbeit sich mit diesem Zusammenhang beschäftigen.

Dazu werde ich zwei Versionen eines MAs implementieren. Eine, die nur die Wichtungen trainiert und eine, die zusätzlich die Struktur des Netzes verändert. Bei Letzterem wird die Struktur eines Perzeptrons, d.h. eines in Schichten aufgebauten Feed-Forward-Netzes[8], beschränkt. Zur Beurteilung des Aufwand-Nutzen-Verhältnisses, also der Zunahme der nötigen Rechenleistung im Vergleich zur Qualitätssteigerung der Lösung, werden beide Versionen auf mehrere einfache Klassifizierungsprobleme angewendet. Ein typisches Beispiel für ein solches Problem ist die Erkennung von handgeschriebenen Ziffern.

Nach Beendigung der systematischen Recherche und des Erstellens einer Bibliographie werde ich umgehend mit der Implementierung beginnen. Falls die Implementierung und die Anwendung mit der zur Verfügung stehenden Rechenleistung erfolgreich ist, kann eventuell eine weitere Stufe ergänzt werden, bei der die Struktur des Netzes nur noch durch die Anzahl der Neuronen begrenzt ist. Dies verkompliziert das Problem jedoch noch einmal stark. Sollte bis zum 20.04.2019 kein wesentlicher Fortschritt bei der Implementierung erkennbar sein, muss ich das Thema der Arbeit noch einmal überdenken. Die Formulierung der theoretischen Grundlagen und der Methoden erfolgt deshalb erst nach der Implementierung. Dieser Schritt sollt bis zum Beginn der Sommerferien am 20.06.2019 abgeschlossen sein. In den Sommerferi-

[1] https://onlinelibrary.wiley.com/doi/epdf/10.1002/phbl.19940500309

[2] https://rubikscode.net/2018/01/22/backpropagation-algorithm-in-artificial-neural-networks/

[3] Mem - ein Kunstwort, dass die kleinste Einheit kultureller Information beschreibt, angelehnt an das Wort Gen. Meme ändern sich, im Gegensatz zu Genen, auch im Laufe des Lebens

[4] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.9573&rep=rep1&type=pdf

[5] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.9142&rep=rep1&type=pdf

[6] https://sites.google.com/site/algoritmalar/performancecomparasionofmemeticalgorithms.pdf

[7] https://www.researchgate.net/profile/Hussein_Abbass/publication/225338453_A_Memetic_Pareto_Evolutionary_Approach_to_Artificial_Neural_Networks/links/0046353c38c1985768000000/A-Memetic-Pareto-Evolutionary-Approach-to-Artificial-Neural-Networks.pdf

[8] Ein neuronales Netz ohne Kreisschlüsse, in denen Signale zirkulieren können

en, also bis zum 02.08.2019, werde ich dann die Ergebnisse und deren Auswertung darstellen. Der bisherige Gliederungsentwurf ist hier noch einmal im Detail dargestellt.

2 Einleitung

3 Theoretische Grundlagen

3.1 Künstliche neuronale Netze

3.2 Evolutionäre Algorithmen

3.3 Memetische Algorithmen

4 Methoden

4.1 Die Vergleichsprobleme

4.2 Implementierung (Pseudocode und Erläuterung)

5 Ergebnisse

6 Auswertung