

Erklärung

Hiermit erkläre ich, dass ich diese schriftliche Abschlussarbeit selbstständig verfasst habe, keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe.

Tübingen, den 14.06.2017

(Steffen Schnürer)

Inhaltsverzeichnis

1	Einführung	1
2	Implementation	2
2.1	Das Bouncing Ball Szenario	2
2.2	Unser LSTM, Parameter und Testfälle	3
2.3	Das Training des Bouncing Verhaltens	5
2.4	Umfragen zum Thema	5
2.5	Feedback	5

1 Einführung

Bishier noch bisschen leer...

2 Implementation

Nun da die Grundlagen gelegt sind können wir anfangen zu untersuchen wie ein LSTM Netz mit Events umgeht. Wir betrachten dazu das Bouncing Ball Szenario, da es klare Events hat und sich dadurch auch gut segmentieren lässt. Zur Implementation wurde JANNNLab (Otte et al. 2013), ein Java Framework für Neuronale Netze verwendet [16]. Dieses Kapitel soll die im Zuge dieser Arbeit verwendeten Techniken erklären und aufzeigen wie die im nächsten Kapitel beschriebenen Ergebnisse erreicht wurden, sowie welche Probleme sich dabei in den Weg gestellt haben.

2.1 Das Bouncing Ball Szenario

Das Bouncing Ball Szenario beschreibt einen Ball, der gleichförmig durch die Ebene gleitet bis er an entsprechenden Begrenzungen abprallt. Wir haben die 1- und die 2-dimensionale Version verwendet.

Im 1D Fall haben wir einen Ball ohne Ausdehnung (also eigentlich ein Punkt aber diese Unterscheidung ist für unsere Zwecke irrelevant) mit einer Position x_b und einer konstanten Geschwindigkeit v . Dieser springt zwischen den Grenzen $x_1 = -1$ und $x_2 = 1$ hin und her, also immer wenn die Position des Balls eine der Grenzen annimmt, wird die Geschwindigkeit invertiert $v := -v$. Also genau so wie man es sich vorstellt. Der 1D Fall unterteilt sich also in 2 Ereignisse, die "Bewegung nach links" und die "Bewegung nach rechts", wenn man das Bouncen zwischen -1 und 1 als Bewegung auf einer horizontalen Linie nimmt.

Der 2D Fall funktioniert analog, der Ball hat eine Position (x_b/y_b) , eine konstante Geschwindigkeit (v_x/v_y) , welcher nun zwischen den 4 Grenzen $x_1 = -1$ und $x_2 = 1$, $y_1 = -1$ und $y_2 = 1$ hin und her springt. Nimmt nun eine Koordinate des Balls eine der Grenzen an wird die entsprechende Koordinate invertiert, also $(v_x/v_y) = (-v_x/v_y)$ beziehungsweise $(v_x/-v_y)$. Der 2D Fall unterteilt sich also in 4 Ereignisse, die "Bewegung in Richtung oben-rechts", etc. wenn man das Bouncen zwischen dem Quadrat von $(-1/-1)$ und $(1/1)$ im 2-dimensionalen als Bewegung in einer stehenden Ebene nimmt.

Das 1D-Szenario habe ich hauptsächlich verwendet um mich einzuarbeiten und zurechtzufinden, da es unter anderem mit weniger Neuronen erlernt werden kann und so die Aktivierungen der einzelnen Gates auch übersichtlicher sind. Der Grund warum das 1D-Szenario auch für diese Ausarbeitung wichtig ist, ist das unter Hinzunahme der Zeitachse aussagekräftige Diagramme erstellt werden können, anhand derer einige Trainingseffekte und Verläufe veranschaulicht werden können. Entsprechende Analysen konnten vom 2D-Fall auch gemacht werden, wenn man sah wie sich eine Linie oder ein Ball mit Verzögerung entsprechend über den Bildschirm bewegt, die resultierenden Bilder sind jedoch eher unübersichtlich. Das 2D Szenario hingegen ist das für unsere Untersuchungen interessantere, da es mehr Events gibt, die auch

in komplexeren Beziehungen zueinander stehen. In Abbildung 2.1 sieht man jeweils 2 Beispiele, zu dem genauen Aufbau des dazu benutzten LSTM Netzes kommen wir in der folgenden Sektion.

2.2 Unser LSTM, Parameter und Testfälle

Für unser LSTM-Netz haben sich folgende Parameter als sinnvoll erwiesen:

Aktivierungsfunktionen:

Zur Ansteuerung der Gates der LSTM-Zellen im Hiddenlayer werden Neuronen mit *Tangens hyperbolicus* als Aktivierungsfunktion verwendet. Für das Outputlayer wird zur Aktivierung die lineare Funktion verwendet, da die Position des Balles gleichverteilt Werte zwischen -1 und 1 annimmt, soll unser Netzoutput dasselbe tun. Anfangs war für das Outputlayer per Defaultoption die *sigmoid*-Funktion zur Aktivierung eingestellt, deren Bild aber nur von 0 bis 1 reicht. Das hatte natürlich nicht funktioniert und für Verwirrung gesorgt, warum denn nur die Hälfte des Problems erlernt wird.

Trainingsoptimierung:

Als Optimierungsmethode für das Training wurde die Adaptive Moment Estimation Methode (ADAM), also eine adaptive Momentumsschätzung, verwendet. Diese baut die übliche Lernmethode mit Lernrate und Momentumrate dahingehend aus, dass diese durchgehend angepasst werden, um noch schneller zu genaueren Ergebnissen zu kommen. Hierzu wurden die Standard Parameter verwendet, also $\beta_1 = 0.9$, $\beta_2 = 0.99$ und $\epsilon = 10^{-8}$. Wobei β_1 und β_2 die Abklingraten des Einflusses der vergangenen Momente auf den jetzigen Zeitschritt sind. Sind sie Nahe 1, klingt der Einfluss langsamer ab. ϵ ist ein glättender Term, der Division durch 0 verhindert. [17] Es wurde für Debugging-Zwecke an diesen Werten herumgespielt, es wurden aber nie sichtlich bessere Ergebnisse erzielt.

Netzgröße:

Die passende Netzgröße ist fallabhängig, in Abbildung 2.1 wurden für die 1D Fälle 1-2-1 Netze verwendet (also 1 Inputneuron, 1 Hiddenlayer mit 2 Neuronen und 1 Outputneuron), für die 2D Fälle 2-4-2 Netze. An anderer Stelle werden andere Netzgrößen verwendet, dann wird das an entsprechender Stelle auch nochmal aufgegriffen.

Außerdem:

Alle Layer sind mit Bias-Unit, um mit möglichst wenig Neuronen möglichst viel Funktionalität zu erreichen, damit deren Analyse wiederum möglichst ausgiebig verläuft. Peepholes [13] wurden getestet, hatten in diesem Beispiel keinen sichtbaren Effekt auf das Training und wurden dann weggelassen.

Außerdem werden verschiedene Testfälle mit verschiedenen Intentionen betrachtet, die ich an dieser Stelle vorgreifend definieren möchte. Es wurde mit fast allen Kombinationen experimentiert:

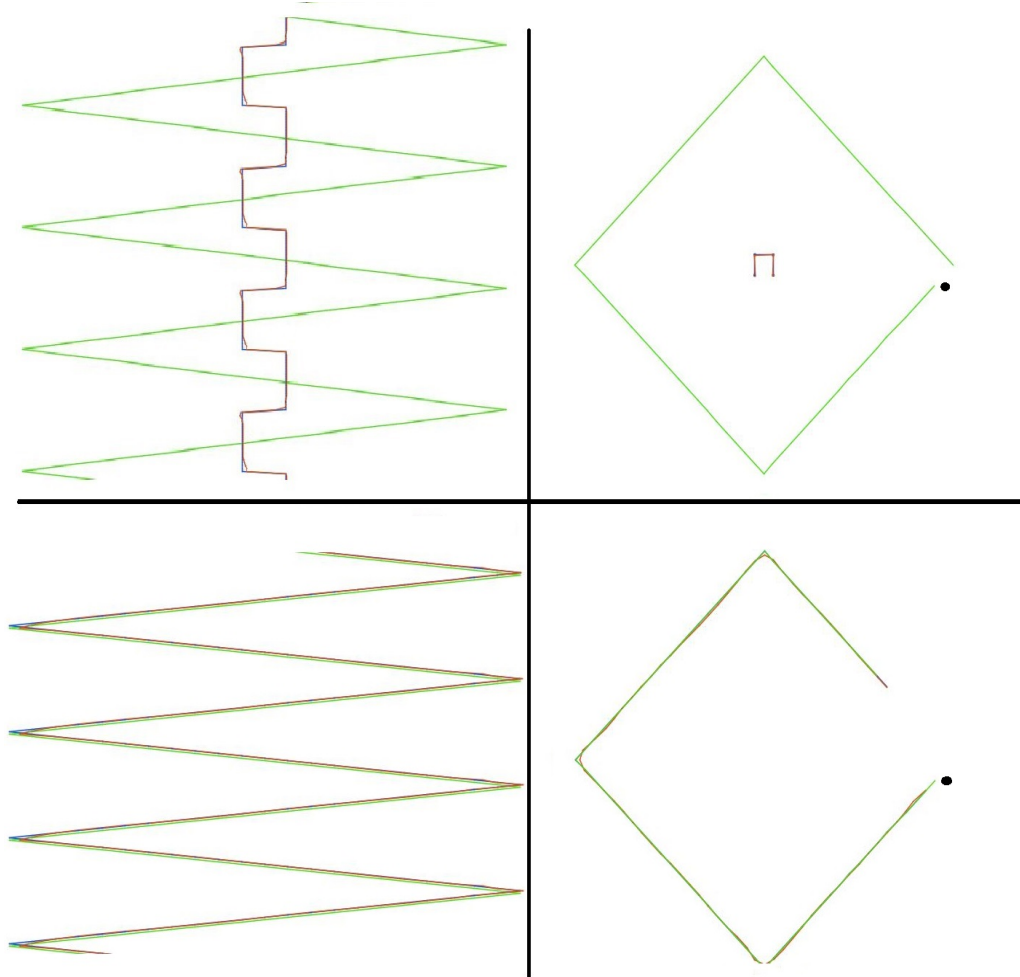


Abbildung 2.1: 4 Versionen des Bouncing Ball Szenario erlernt durch LSTM Netze. Zu sehen sind der Netinput (grün), der Netzoutput (rot) und das Trainingsziel (blau). Links sieht man 2 Beispiele des 1D Falls, zur besseren Veranschaulichung sind die Daten nach den Zeitschritten vertikal versetzt. Rechts sieht man 2 Beispiele des 2D Falls, die Abbildung zeigt jeweils den Verlauf des Balles bis kurz vor dem vollenden einer Runde, um Überlagerungen in der Abbildung zu vermeiden. Der Startpunkt ist durch den schwarzen Punkt gekennzeichnet. In den jeweils oberen Fällen wurde als Netinput die Position des Balles und als Trainingsziel die Geschwindigkeit des Balles für den nächsten Schritt. In den jeweils unteren Fällen wurde als Netinput die Position des Balles und als Trainingsziel die vorauszuberechnende nächste Position des Balles.

Verschiedene Inputs: • Position: Der Standard Input

- Noicy Position:
- Null:
- Geschwindigkeit:

Der Standart Input in das Netz ist die Position des Balles.

Verschiedene Trainingsziele:

Nächste Position: Der Standard Output.

Geschwindigkeit: Da die Geschwindigkeit bis auf Vorzeichen konstant ist, kann sich das Netz hier darauf konzentrieren, in welchem Ereignis, also Bewegungsabschnitt es sich befindet, also ob sich der Ball momentan z.B. nach links oben bewegt.

Nächstes Event: Es wurde auch getestet wie sich das Netz verhält wenn es selbst vorhersagen soll, welches Event als nächstes ansteht. War erfolgreich auch bei zufälligen Startposition und Geschwindigkeiten, was aber nicht weiter Überraschend ist, das sich das LSTM Netz auf ein lineares Problem, welche Grenze als nächstes getroffen wird, trainieren lässt. Auch aus den Aktivierungen im Netz konnte nichts interessantes gefolgert werden, weswegen das nächste Event als Trainingsziel nicht mehr aufgegriffen wird.

Startposition und Geschwindigkeit:

Feedback:

2.3 Das Training des Bouncing Verhaltens

2.4 Umfragen zum Thema

1 von 1 befragten Personen mit einem Abschluss in Bachelor of Science meinten zum Thema:

Davon weiß ich nichts. [9]

2.5 Feedback

Das Training des Bouncing Verhaltens Das interessante am Bouncing Ball Szenario ist natürlich nicht die gleichförmige Bewegung, sondern das nicht-lineare Verhalten beim Abprallen. Dieses wollen wir unserem LSTM Netzwerk möglichst genau erlernen und haben dazu einige Techniken angewendet.

Reaktives vs vorhersehendes Verhalten 1D mit 1b Zelle

RNN Annäherung vs echte nichtlinearität Random vs nicht Random, nur periodisch gelernt Feedback für random, erklären warum es nicht funktioniert hat, Feedback, Zittern erklären für Input außerhalb des eigenen Raums Schritte damit es funktioniert, feedback annäherung, hat nicht geholfen Random diskretisiert um numerisch genau definiertes Rand verhalten zu haben

Literaturverzeichnis

- [1] online.science.psu.edu/bisc004_activewd001/node/1907 (abgerufen am 10.05.2017)
- [2] de.wikipedia.org/wiki/K%C3%BCnstliches_Neuron (abgerufen am 10.05.2017)
- [3] de.wikipedia.org/wiki/Neuronales_Netz#/media/File:Neural_network.svg (abgerufen am 10.05.2017)
- [4] en.wikipedia.org/wiki/List_of_animals_by_number_of_neurons (abgerufen am 20.05.2017)
- [5] www.neuronalesnetz.de/ (abgerufen am 02.06.2017)
- [6] www.informatikseite.de/neuro/node24.php (abgerufen am 03.06.2017)
- [7] www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduct (abgerufen am 04.06.2017)
- [8] eric-yuan.me/rnn2-lstm/ (abgerufen am 14.05.2017)
- [9] www.quora.com/How-does-LSTM-help-prevent-the-vanishing-and-exploding-gradi (abgerufen am 04.06.2017)
- [10] research.googleblog.com/2016/05/chat-smarter-with-allo.html (abgerufen am 04.06.2017)
- [11] www.theinformation.com/apples-machines-can-learn-too (abgerufen am 04.06.2017)
- [12] www.allthingsdistributed.com/2016/11/amazon-ai-and-alexa-for-all-aws-apps.html (abgerufen am 04.06.2017)
- [13] colah.github.io/posts/2015-08-Understanding-LSTMs/ (abgerufen am 04.06.2017)
- [14] www.ncbi.nlm.nih.gov/pmc/articles/PMC3314399/ (abgerufen am 05.06.2017)
- [15] www.ncbi.nlm.nih.gov/pmc/articles/PMC2852534/ (abgerufen am 05.06.2017)
- [16] www.github.com/JANNLab/JANNLab (abgerufen am 04.06.2017)

- [17] www.sebastianruder.com/optimizing-gradient-descent/ (abgerufen am 04.06.2017)
- [18] (abgerufen am 04.06.2017)
- [19] (abgerufen am 04.06.2017)
- [20] (abgerufen am 04.06.2017)
- [21] (abgerufen am 04.06.2017)
- [22] (abgerufen am 04.06.2017)
- [23] (abgerufen am 04.06.2017)
- [24] (abgerufen am 04.06.2017)
- [25] (abgerufen am 04.06.2017)
- [26] (abgerufen am 04.06.2017)