Karlsruher Institut für Technologie

Fakultät für Informatik
Institut für Anthropomatik und Robotik
Lehrstuhl für Intelligente Sensor-Aktor-Systeme (ISAS)
Prof. Dr.-Ing. Uwe D. Hanebeck

Masterarbeit

Ableitung von Bewegungsmodellen für Anwendungen in der Schüttgutsortierung mittels Machine Learning

Tobias Hornberger

31. Dezember 2020

Referent: Prof. Dr.-Ing. Uwe D. Hanebeck

Betreuer: Dipl.-Inform. Florian Pfaff

Zusammenfassung

Abstract.

Eidesstattliche Erklärung

Hiermit erkläre ich, die vorliegende Masterarbeit selbstständig angefertigt zu haben. Die verwendeten Quellen sind im Text gekennzeichnet und im Literaturverzeichnis aufgeführt.

Karlsruhe, 31. Dezember 2020

Tobias Hornberger

Contents

List of Figures

List of Algorithms

List of Examples

List of Review Material

Notation

Conventions

x	Scalar
$\boldsymbol{x}$	Random variable
$\hat{x}$	Mean of random variable $\boldsymbol{x}$.
$\underline{x}$	Column vector
$\underline{\boldsymbol{x}}$	Random vector
$\hat{\underline{x}}$	Mean of random vector $\underline{\boldsymbol{x}}$.
$\mathbf{A}$	Matrix
$(.)_k$	Quantity at time step k.
$\mathbb{R}$	Set of real numbers.
$\sim$	Distribution operator. E.g., $\boldsymbol{x} \sim \mathcal{U}$ means $\boldsymbol{x}$ is distributed according to $\mathcal{U}$.
$\blacksquare$	End of example.
$\square$	End of proof.

Abbreviations

KF	Kalman Filter
LRKF	Linear Regression Kalman Filter
RMSE	Root Mean Square Error

CHAPTER 1

Grundlagen

1.1 Das Kalman-Filter

Als Kalman-Filter bezeichnet man ein mathematisches Verfahren mit dem Messfehler in realen Messwerten reduziert werden können und nicht messbare Systemgrößen geschätzt werden können.

[vergangene, aktuelle und zukünftige Systemzustände schätzen] [Einschränkung Linearität (Extended Kalman) und Gauß rauschen]

Der Zustand des Systems zum Zeitschritt t wird als y_t und die Messung im Zeitschritt t als z_t bezeichnet.

$$y_t = Ay_{t-1} + w, w \sim N(0, Q)$$

$$z_t = Hy_t + v, v \sim N(0, R)$$

Dabei ist A die Zustandsübergangsmatrix, die den Übergang von einem Zustand in den nächsten beschreibt. H ist die Messmatrix, die beschreibt wie Messungen aus dem Zustand entstehen und Q und R sind die Kovarianzmatrizen des Systemrauschens beziehungsweise des Messrauschens.

Das Kalman-Filter funktioniert mittels abwechselnd ausgeführter *predict* und *update* Schritte.

$$\hat{y}_t' = A\hat{y}_{t-1}'$$

$$\hat{P}_t' = A\hat{P}_{t-1}'A^T + Q$$

1.2 Neuronale Netze

Die Grundsteine des Feldes wurde 1943 von Warren McCulloch und Walter Pitts gelegt, die in ihrem Paper ein Neuronenmodell vorschlugen, mit dem sich logische arithmetische Funktionen berechnen lassen. Infolge dessen gab verschiedene Forschungsbestrebungen in dem Feld, wie [Examples: TODO!].

[Viele der Begrifflichkeiten, die wir heute noch verwenden wurden 1956 auf der Dartmouth Conference festlegt.]

Nachdem jedoch Marvin Minsky und Seymour Papert zeigten, dass einzelne Perzeptrons nicht in der Lage sind linear nicht separierbare Probleme zu lösen sank das Interesse an dem Feld.

1.2.1 Perzeptron

Die kleinste Einheit eines neuronalen Netzes ist das Perzeptron. Es ist eine Art künstliches Neuron, dass eine Reihe an Eingaben entgegen nimmt und einen einzelnen Wert o ausgibt. Die einzelnen Eingaben x_i haben jeweils eine Gewichtung w_i. Es existiert ein sogenannter Schwellwert oder *bias*, der normalerweise durch eine zusätzliche Eingabe x_{m+1} mit dem Wert $+1$ und dem dazugehörigen Gewicht w_{m+1} modelliert wird. Den Ausgabewert y erhält man dadurch, dass man die gewichteten Eingaben aufsummiert und in die Aktivierungsfunktion des Perzeptrons gibt. Ein Überblick über verschiedene Aktivierungsfunktionen ist unter 1.2.2 zu finden.

Mathematisch ist die Ausgabe eines Perzeptrons also wie folgt definiert:

$$y = \phi(\sum_{i=0}^{m} w_i x_i)$$

Beim Lernen werden die Gewichte der einzelnen Eingaben so an gepasst, dass die gewünschte Ausgabe erreicht wird. Ein einzelnes Perzeptron mit zwei Eingängen kann zur Darstellung der logischen Operatoren AND, OR und NOT genutzt werden

Letztendlich ist ein solches Perzeptron jedoch nur ein linearer Klassifikator und kann somit zum Beispiel den XOR Operator nicht auflösen. Um solche, nicht linearseparierbare Probleme zu lösen müssen mehrere Schichten an Neuronen kombiniert werden.

1.2.2 Aktivierungsfunktionen

Es gibt verschiedene Aktivierungsfunktionen, die für den Einsatz in neuronalen Netzen in Frage kommen. Sie sind von essenzieller Wichtigkeit, da ohne eine Nicht-Linearität das Netz in eine einfache Regression kollabiert.

Eine Aktivierungsfunktion sollte leicht abzuleiten sein, da dies im Rahmen des Backpropagation Algorithmus häufig geschieht und sonst beträchtlicher Rechenaufwand entsteht.

Einige häufig verwendete Aktivierungsfunktionen sollen hier vorgestellt werden. Jede dieser Funktionen stellt eine Nicht-Linearität dar und nimmt eine einzelne Zahl, wendet eine bestimmte, festgelegte mathematische Operation auf diese an und gibt das Ergebnis zurück.

Sigmoid-Funktion

$$f(x) = \frac{1}{1+e^x} = \frac{e^x}{e^{x+1}}$$

$$f'(x) = f(x) * (1 - f(x))$$

Die mathematische Form der Sigmoid Aktivierungsfunktion ist in Abbildung 1.1 zu sehen. Sie bildet die reellen Zahlen $\mathbb{R}$ auf das Intervall $(0, 1)$ ab. Für betragsmäßig größer werdende negative Zahlen nähert sich der Rückgabewert 0 an, ebenso wie für größer werdende positive Zahlen sich der Rückgabewert an 1 annähert.

Die Sigmoid Funktion ist eine historisch häufig genutze Funktion, da sie das Verhalten eines natürlichen Neurons, der biologischen Motivation für künstliche Neuronen, gut nachbildet: komplette Inaktivität eines Neurons bei Ausgabe 0 bis zum feuern mit maximaler Frequenz bei Ausgabe 1.

In der Praxis jedoch haben sich einige Nachteile der Sigmoid Funktion gezeigt, weshalb sie quasi nicht mehr genutzt wird. Der gewichtigste von diesen ist, dass ihre Ableitung bei großen Beträgen beinah 0 ist. Dies führt dazu, dass während der Ausführung des Backpropagation-Algorithmus beinah keine Änderungen passieren und dementsprechend das Netz sehr langsam lernt.

TanH

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$f'(x) = 1 - f(x)^2$$

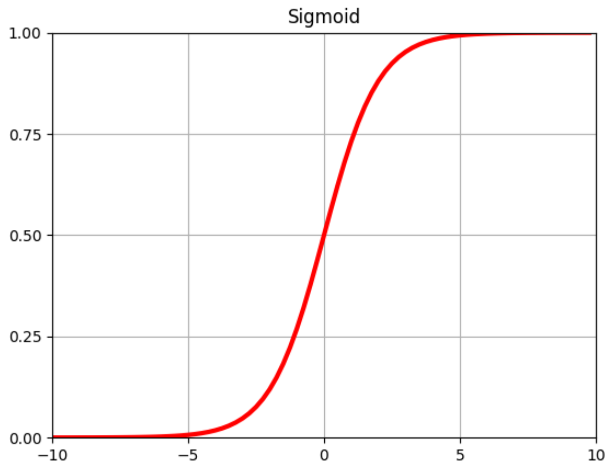


Figure 1.1: Plot der Sigmoid Funktion

Die tanh Aktivierungsfunktion ist in Abbildung 1.2 dargestellt. Im Gegensatz zur Sigmoid Funktion bildet sie die reellen Zahlen $\mathbb{R}$ auf das Intervall $(-1, 1)$ ab. Weil sie zentriert um den Nullpunkt ist, wird sie bei realen Anwendungen der Sigmoid Funktion vorgezogen. Das Saturationsproblem der Sigmoid Funktion besteht jedoch immer noch.

ReLU

$$f(x) = max(0, x)$$

$$f'(x) = \begin{cases} 0 & \text{, falls } x < 0 \\ 1 & \text{, falls } x > 0 \end{cases}$$

Abbildung 1.3 zeigt den Plot einer *Rectified Linear Unit*, oder kurz ReLU. Die Aktivierung von ReLus ist ein einfacher Schwellwert, der weit weniger rechenintensiv ist, als die aufwendigen Exponenzialfunktionen von Sigmoid und tanh. In der Praxis hat sich gezeigt zudem gezeigt, dass ReLus deutlich schneller konvergieren als Sigmoid- oder tanh-Neuronen. Krizhevsky et al. haben in ihrem Paper[5] einen Geschwindigkeitsgewinn um Faktor 6 feststellen können. Ein Problem, das mit

test123

1.3 Related Work

Notizen:

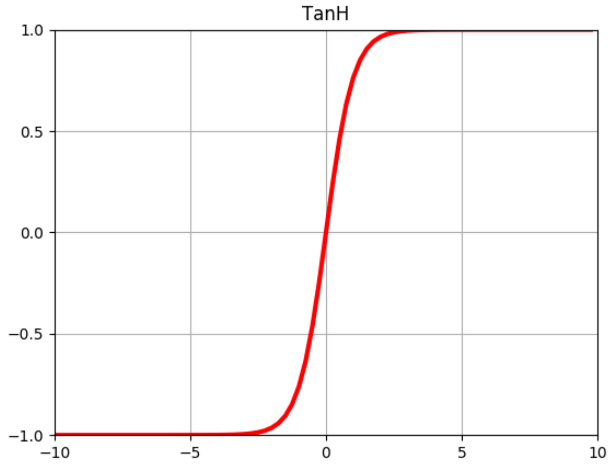


Figure 1.2: Plot der Tanh Funktion

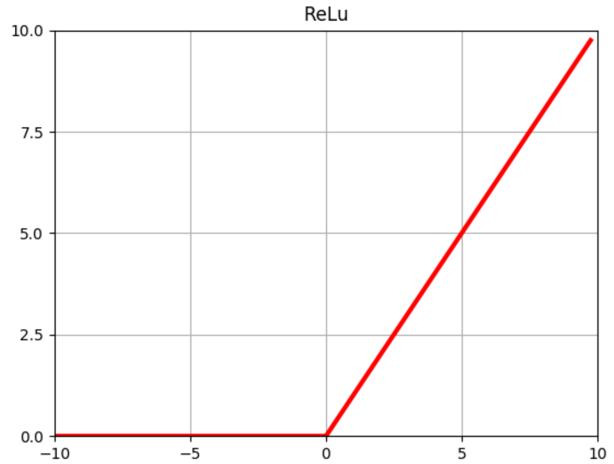
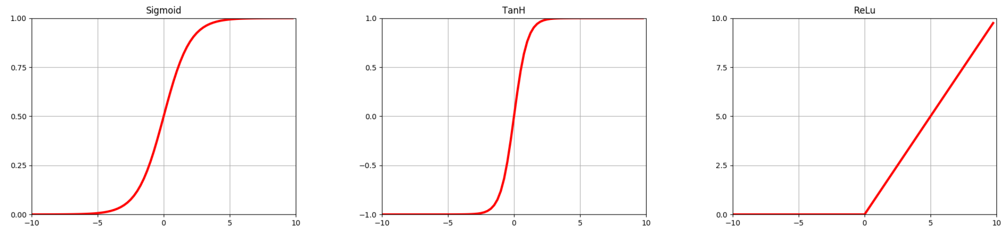


Figure 1.3: Plot der Ausgabe einer ReLUs



(a) Sigmoid Funktion **(b)** TanH Funktion **(c)** ReLU

Figure 1.4: Häufig verwendete Aktivierungsfunktionen

Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization von Huseyin et al. - Motion Model und Measurement Model mittels LSTM Netzen lernen. - Am Beispiel Pose-Estimation. - 3 Separate LSTMs, je eins für Motionmodel, System und Messrauschen. (A = Transition Matrix, H = Measurement Matrix, Q = Covariance of process noise, R = Covariance of sensor noise)

CHAPTER 1

Bibliography

[1] Pieter Abbeel et al. “Discriminative Training of Kalman Filters”. In: *In Proceedings of Robotics: Science and Systems*. 2005.

[2] Huseyin Coskun et al. “Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization”. In: *CoRR* abs/1708.01885 (2017). URL: http://arxiv.org/abs/1708.01885.

[3] Patrick Emami et al. “Machine Learning Methods for Solving Assignment Problems in Multi-Target Tracking”. In: *arXiv preprint arXiv:1802.06897* (2018).

[4] Rahul G. Krishnan, Uri Shalit, and David Sontag. “Deep Kalman Filters”. In: (Nov. 16, 2015). arXiv: 1511.05121 [cs, stat]. URL: http://arxiv.org/abs/1511.05121 (visited on 05/14/2018).

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[6] John Langford, Ruslan Salakhutdinov, and Tong Zhang. “Learning Nonlinear Dynamic Models”. In: *CoRR* abs/0905.3369 (2009). URL: http://arxiv.org/abs/0905.3369.

[7] X Rong Li and Vesselin P Jilkov. “Survey of Maneuvering Target Tracking. Part I. Dynamic Models”. In: *IEEE Transactions on aerospace and electronic systems* 39.4 (2003), pp. 1333–1364.

[8] Anton Milan et al. “Online Multi-Target Tracking Using Recurrent Neural Networks.” In: *AAAI*. 2017, pp. 4225–4232.

[9] F. Pfaff et al. “TrackSort: Predictive Tracking for Sorting Uncooperative Bulk Materials”. In: *2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. Sept. 2015, pp. 7–12. DOI: 10.1109/MFI.2015.7295737.

[10] Florian Pfaff et al. “Improving Optical Sorting of Bulk Materials Using Sophisticated Motion Models”. In: *tm-Technisches Messen* 83.2 (2016), pp. 77–84.

[11] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “Facenet: A Unified Embedding for Face Recognition and Clustering”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 815–823.

[12] Kolja Thormann, Fabian Sigges, and Marcus Baum. “Learning an Object Tracker with a Random Forest and Simulated Measurements”. In: *Proceedings of the 20th International Conference on Information Fusion (FUSION 2017)*. Xi’an, P.R. China, July 2017.

[13] L. Wang, L. Zhang, and Z. Yi. “Trajectory Predictor by Using Recurrent Neural Networks in Visual Tracking”. In: *IEEE Transactions on Cybernetics* 47.10 (Oct. 2017), pp. 3172–3183. ISSN: 2168-2267. DOI: 10.1109/TCYB.2017.2705345.

[14] Robert Wilson and Leif Finkel. “A Neural Implementation of the Kalman Filter”. In: *Advances in Neural Information Processing Systems*. 2009, pp. 2062–2070.