künstliches neuronales Netz (KNN)

Autor und Herausgeber

Alexander Jung*, Department of Computer Science, Aalto University, Espoo, Finland (ORCID 0000-0001-7538-0990)
Editors: Konstantina Olioumtsevits (ORCID 0000-0003-2697-8659), Ekkehard Schnoor (ORCID 0000-0001-8252-1499), Aalto University, Espoo, Finland
*Corresponding author: alex.jung@aalto.fi

Zusammenfassung

Ein künstliches neuronales Netz (KNN) ist ein Hypothesenraum, dessen Hypothese-Abbildungen alle auf einem einzigen gerichteten Graphen aufbauen. Jede Hypothese in diesem Raum ist eine Abbildung von den Merkmale eines Datenpunkts auf eine Vorhersage für das Label, die sich für eine bestimmte Wahl der lernbaren Modellparameter ergibt, welche den Kanten (Kantengewichte) und Knoten (Offset-Termen) des Graphen zugeordnet sind. Jeder Knoten ist ein künstliches Neuron, das eine Aktivierungsfunktion auf die gewichtete Summe seiner Eingaben zuzüglich seines Offsets anwendet. Das Verbindungsmuster, die Architektur, ist eine wichtige Entwurfsentscheidung: Ein vorwärtsgerichtetes Netz verwendet einen gerichteten azyklischen Graphen, während ein rekurrentes neuronales Netz (RNN) einen Zyklus zulässt. Abhängig von der Konnektivität umfassen KNNs tiefe Netze, Convolutional Neural Networks (CNN), rekurrente neuronale Netze und Transformer-basierte Architekturen.

Definition

Die Gesichtserkennung einer Smartphone-Kamera und die Sprachtranskription eines Sprachassistenten werden von KNNs berechnet, deren Verbindungsstärken aus Daten gelernt wurden (Goodfellow et al., 2016, Abschn. 12.2 und 12.3). Ein KNN ist ein Hypothesenraum, dessen Hypothese-Abbildungen alle auf einem einzigen gerichteten Graphen aufbauen: Seine Knoten sind die künstlichen Neuronen, elementare Recheneinheiten, und seine gewichteten Kanten legen fest, welche künstliche Neuronen welche speisen. Die Kantengewichte und die den Knoten zugewiesenen Offset-Terme bilden die lernbaren Modellparameter $\mathbf{w}$ eines KNN, und eine Wahl von $\mathbf{w}$ legt eine Hypothese-Abbildung $h^{(\mathbf{w})}$ fest, von den Merkmale eines Datenpunkts am Eingang auf eine Ausgabe, die als Vorhersage für das Label verwendet wird. Die gelernte Hypothese $\widehat{h} = h^{(\widehat{\mathbf{w}})}$ ist eine solche Wahl, die mittels Gradientenabstiegs auf dem Trainingsfehler gewonnen wird, wobei der Gradient durch Backpropagation berechnet wird (Rumelhart et al., 1986).

Jedes künstliche Neuron wendet eine Aktivierungsfunktion $\sigma(\cdot)$ auf die gewichtete Summe seiner Eingaben zuzüglich eines Offset-Terms an (Abb. 1). Die Aktivierungsfunktion kann selbst Modellparameter tragen (parametrische Aktivierungsfunktionen) (He et al., 2015), wird aber üblicherweise im Voraus fest-

gelegt, zum Beispiel die ReLU (ReLU) (Goodfellow et al., 2016, Abschn. 6.3), sodass nur der knotenweise Offset und die Kantengewichte gelernt werden.

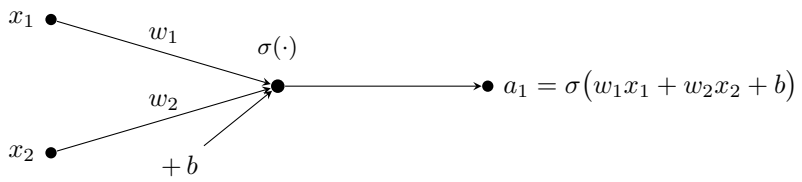

Abbildung 1: Ein einzelnes künstliches Neuron: Es wendet eine Aktivierungsfunktion $\sigma(\cdot)$ auf die gewichtete Summe $w_1x_1 + w_2x_2$ seiner Eingaben zuzüglich eines Offset-Terms b an und liefert die Ausgabe a_1. Die Kantengewichte w_1, w_2 und der Offset b sind lernbare Modellparameter

Die Ausgabe eines Neurons kann entweder als endgültige Ausgabe des KNN oder als Eingabe für andere Neuronen dienen. Der gerichtete Graph des KNN hält diese Verwendungen fest: Seine Knoten sind die künstlichen Neuronen zusammen mit den Eingabe-Merkmale, und jede gerichtete Kante leitet die Ausgabe eines Knotens an den Eingang eines anderen weiter (Abb. 2). Welche Ausgaben mit den Eingängen welches nachgelagerten Neurons verbunden sind – die Verbindungsstruktur (oder Architektur) des KNN –, ist daher ein wichtiger Entwurfsaspekt (Goodfellow et al., 2016). Ob der Graph azyklisch ist oder einen Zyklus enthält, trennt die beiden großen Familien von KNNs.

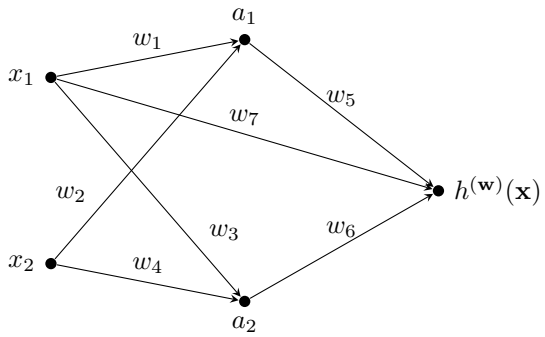

Abbildung 2: Ein KNN, dargestellt als gewichteter, azyklischer gerichteter Graph mit Knoten, die Neuronen oder Merkmale eines Datenpunkts entsprechen. Merkmalsknoten x_1, x_2 haben eine feste, durch den Merkmalswert gegebene Ausgabe und keinen Eingang. Jeder der Neuronenknoten a_1, a_2 wendet eine Aktivierungsfunktion auf die gewichtete Summe seiner Eingaben an (jeweils wie in Abb. 1, deren Offset-Terme hier weggelassen sind). Die gewichteten gerichteten Kanten, deren Pfeilspitzen die Richtung des Signalflusses markieren, geben an, wie Knoten-Ausgaben als Eingaben für nachgelagerte Neuronen verwendet werden. Die Kantengewichte sind anpassbare Modellparameter und dienen zur Skalierung der Eingaben in die Neuronen. Die Vorhersage $h^{(\mathbf{w})}(\mathbf{x})$ ist die endgültige Ausgabe des KNN, berechnet aus den Eingaben x_1, a_1 und a_2

Der Graph in Abb. 2 ist azyklisch, sodass dieses KNN vorwärtsgerichtet (engl. *feedforward*) ist (Goodfellow et al., 2016, Kap. 6): Das Signal läuft von den Eingängen zur Ausgabe, ohne zu einem Knoten zurückzukehren. Ein weit verbreiteter Typ von KNNs sind tiefe Netze, bei denen Neuronen aufeinanderfolgende Schichten bilden (Abb. 3). In einem tiefen Netz sind die Ausgaben der Neuronen in einer gegebenen Schicht typischerweise nur mit den Eingängen der Neuronen in einer darauffolgenden Schicht verbunden. Im Gegensatz dazu kann im allgemeinen gerichteten azyklischen Graphen aus Abb. 2 eine Ausgabe jedes beliebige nachgelagerte Neuron speisen, und dort ist x_1 direkt mit der Vorhersage verbunden.

Jede Schicht l wendet dann eine Merkmalstransformation $\mathbf{\Phi}^{(l)}$ an, deren Komponenten von den Neuronen dieser Schicht berechnet werden (jeweils wie in Abb. 1), und das tiefe Netz berechnet die Verkettung dieser Schichtweisen Abbildungen. Eine Kurzschluss- oder Skip-Verbindung verbindet die Ausgaben von Neuronen einer Schicht direkt mit den Eingängen von Neuronen in einer nicht unmittelbar darauffolgenden Schicht (Goodfellow et al., 2016; He et al., 2016); Abb. 3 zeigt eine solche als gestrichelte Kante. Ohne diese Kante liefert das Netz aus Abb. 3 die Vorhersage

$$h^{(\mathbf{w})}(\mathbf{x}) = \mathbf{\Phi}^{(3)}\Big(\mathbf{\Phi}^{(2)}\big(\mathbf{\Phi}^{(1)}(\mathbf{x})\big)\Big).$$

Hier ist die Eingabe der Merkmalsvektor $\mathbf{x} = (x_1, x_2, x_3)^\top$. Die beiden ver-

deckten Schichten berechnen $\mathbf{\Phi}^{(1)}, \mathbf{\Phi}^{(2)} : \mathbb{R}^3 \to \mathbb{R}^3$, und die Ausgabe-Schicht berechnet $\mathbf{\Phi}^{(3)} : \mathbb{R}^3 \to \mathbb{R}$.

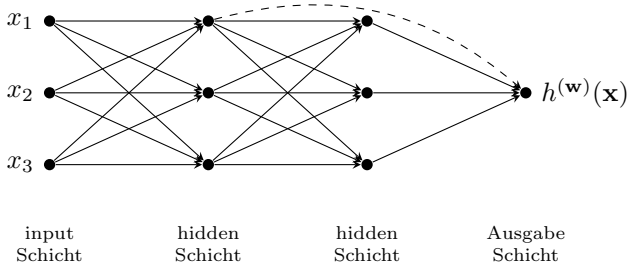

Abbildung 3: Ein tiefes Netz: Die Neuronen bilden aufeinanderfolgende Schichten, und jedes Neuron empfängt nur die Ausgaben der vorangehenden Schicht. Jede durchgezogene Kante trägt ein lernbares Kantengewicht (nicht eingezeichnet). Die gestrichelte Kante ist eine Skip-Verbindung, die zwei nicht unmittelbar aufeinanderfolgende Schichten direkt verbindet

Die bisher betrachteten Graphen sind azyklisch. Ein rekurrentes neuronales Netz (RNN) enthält stattdessen eine Rückkopplungskante (Goodfellow et al., 2016, Kap. 10), sodass sein Graph einen Zyklus aufweist und kein DAG ist (Abb. 4): Eine Selbstschleife führt die Ausgabe eines Neurons an seinen eigenen Eingang zurück. Durch diese Rückkopplung kann das Netz Informationen über die Schritte einer Folge hinweg weitertragen; rollt man die Rekurrenz über die Folge aus, mit einer Kopie jedes Neurons pro Schritt, erhält man wieder einen gerichteten azyklischen Graphen (Goodfellow et al., 2016, Abschn. 10.1).

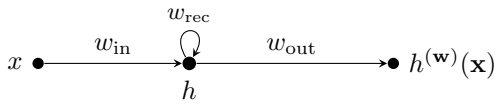

Abbildung 4: Ein einfaches RNN. Die Selbstschleife am verdeckten Neuron h (Kantengewicht w_{rec}) führt dessen Ausgabe an seinen eigenen Eingang zurück, sodass der Graph einen Zyklus enthält und kein DAG ist. Die Rückkopplung trägt Informationen über die Schritte einer Folge hinweg weiter

Die Netze der Abb. 2, 3 und 4 unterscheiden sich nur darin, welche Knoten ihre gerichteten Kanten verbinden. Dieses Verbindungsmuster gibt den gängigen Architekturen ihren Namen. Vorwärtsgerichtete Netze haben einen azyklischen Graphen, einen gerichteten azyklischen Graphen (tiefe Netze sind der geschichtete Fall), während Convolutional Neural Networks (CNN) ebenfalls vorwärtsgerichtet sind, aber Gewichte über räumliche Positionen hinweg gemeinsam nutzen. Rekurrente neuronale Netze fügen Rückkopplungsverbindungen für die Verarbeitung von Folgen hinzu, und Transformer-basierte Architekturen, die modernen große Sprachmodelle (LLM) zugrunde liegen, sind ebenfalls

vorwärtsgerichtet und nutzen Attention, um jedes Token mit jedem anderen zu verbinden (Vaswani et al., 2017, Abschn. 3). Skip-Verbindungen treten in diesen Architekturen als die Residualverbindungen tiefer Convolutional Neural Networks (He et al., 2016) und Transformer (Vaswani et al., 2017) auf. Jede Architektur legt einen anderen Hypothesenraum fest, innerhalb dessen die empirische Risikominimierung eine einzelne Hypothese $\widehat{h} = h^{(\widehat{\mathbf{w}})}$ lernt, indem sie diejenigen Modellparameter $\widehat{\mathbf{w}}$ wählt, die den durchschnittlichen Verlust auf einem Trainingsdatensatz minimieren. Die Architektur bestimmt auch die Generalisierung der gelernten Hypothese auf Datenpunkte außerhalb des Trainingsdatensatzs, woran die Wahl einer Architektur gemessen wird. Diese Wahl muss nicht von Hand getroffen werden: Die neuronale Architektursuche (engl. *neural architecture search*) lernt eine geeignete Konnektivität automatisch (Elsken et al., 2019).

Siehe auch: Label, künstliches Neuron, Aktivierungsfunktion, tiefes Netz, Schicht, CNN, RNN, Transformer.

Querverweise

Literatur

1. Elsken et al. (2019). Neural Architecture Search: A Survey. J. Mach. Learn. Res., 20(55), pp. 1–21.

2. Goodfellow et al. (2016). Deep Learning. MIT Press, Cambridge, MA, USA.

3. He et al. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: 2015 IEEE Int. Conf. Comput. Vis. (ICCV), pp. 1026–1034.

4. He et al. (2016). Deep Residual Learning for Image Recognition. In: 2016 IEEE Conf. Comput. Vis. Pattern Recognit., pp. 770–778.

5. Rumelhart et al. (1986). Learning Representations by Back-Propagating Errors. Nature, 323, pp. 533–536.

6. Vaswani et al. (2017). Attention is all you need. In: Adv. Neural Inf. Process. Syst., pp. 5998–6008.