Autoencoder

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 Autoencoder ist eine Methode des maschinellen Lernens, die einen Encoder und einen Decoder gemeinsam erlernt. Der Encoder bildet den Merkmalsvektor eines Datenpunkts auf einen neuen Merkmalsvektor ab, der als Code bezeichnet wird. Der Decoder rekonstruiert den ursprünglichen Merkmalsvektor aus dem Code. Der vom Encoder gelieferte Code ist typischerweise wesentlich kürzer als der ursprüngliche Merkmalsvektor, sodass der Code nur einen Teil der Information im Merkmalsvektor bewahren kann. Verschiedene Autoencoder verwenden unterschiedliche Hypothesenräume, aus denen der Encoder und der Decoder gewählt werden. Der Spezialfall, in dem beide lineare Abbildungen sind, lernt denselben Unterraum wie die Hauptkomponentenanalyse (PCA), und ein tiefer Autoencoder verwendet auf jeder Seite ein tiefes Netz.

Definition

Dreihundert Punkte entlang einer Kurve in der Ebene (Abb. 3) lassen sich jeweils als eine einzige Zahl speichern, aus der der Punkt nahezu exakt wiederhergestellt werden kann; ebenso lässt sich ein Bild als kurzer Code speichern, aus dem eine genaue Kopie wiederhergestellt wird. Ein Autoencoder lernt einen Encoder $\boldsymbol{\Phi}$ aus einem Hypothesenraum $\mathcal{H}$ zusammen mit einem Decoder $\boldsymbol{\Phi}^*$ aus einem zweiten Hypothesenraum $\mathcal{H}^*$ (Goodfellow et al., 2016, Kap. 14). Der Encoder bildet den Merkmalsvektor $\mathbf{x} \in \mathbb{R}^d$ eines Datenpunkts auf einen Code $\mathbf{z} = \boldsymbol{\Phi}(\mathbf{x}) \in \mathbb{R}^{d'}$ ab. Der Decoder bildet den Code zurück auf eine Rekonstruktion $\widehat{\mathbf{x}} = \boldsymbol{\Phi}^*(\mathbf{z})$ des ursprünglichen Merkmalsvektors ab (siehe Abb. 1). Der Encoder und der Decoder können mittels empirischer Risikominimierung mit einem Verlust gelernt werden, der den Rekonstruktionsfehler $\mathbf{x} - \boldsymbol{\Phi}^*(\boldsymbol{\Phi}(\mathbf{x}))$ misst, wie etwa dem quadratischen Fehlerverlust (Goodfellow et al., 2016, Abschn. 14.1),

$$\min_{\boldsymbol{\Phi}\in\mathcal{H},\, \boldsymbol{\Phi}^*\in\mathcal{H}^*} \frac{1}{m}\sum_{r=1}^{m} L\Big(\mathbf{x}^{(r)},\, \boldsymbol{\Phi}^*\big(\boldsymbol{\Phi}\big(\mathbf{x}^{(r)}\big)\big)\Big). \tag{1}$$

Der Trainingsdatensatz benötigt keine Labels: Was der Decoder reproduzieren muss, ist der Datenpunkt selbst; daher ist ein Autoencoder eine Form des selbstüberwachten Lernens, da er keine manuelle Annotierung erfordert.

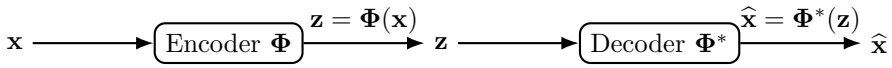


Abbildung 1: Ein Autoencoder: Der Encoder $\mathbf{\Phi}$ bildet einen Merkmalsvektor $\mathbf{x}$ auf einen Code $\mathbf{z}$ ab, und der Decoder $\mathbf{\Phi}^*$ bildet den Code zurück auf eine Rekonstruktion $\widehat{\mathbf{x}}$ ab. Beide werden gemeinsam gelernt, indem ein quantitatives Maß für den Rekonstruktionsfehler $\mathbf{x} - \mathbf{\Phi}^*(\mathbf{\Phi}(\mathbf{x}))$ minimiert wird

Die Länge d' des Codes ist typischerweise wesentlich kleiner als die Länge d des ursprünglichen Merkmalsvektors; in diesem Fall wird der Autoencoder als untervollständig (engl. *undercomplete*) bezeichnet (Goodfellow et al., 2016, Abschn. 14.1). Der Encoder implementiert dann eine Form der Kompression, die es dem Decoder dennoch erlaubt, den ursprünglichen Merkmalsvektor mit einem kleinen Rekonstruktionsfehler zu rekonstruieren (Hinton and Salakhutdinov, 2006). Die Abbildung des Encoders kann anschließend als Merkmalstransformation für Lernaufgaben verwendet werden, die auf denselben Datenpunkte basieren, was den Autoencoder zu einer Methode des Merkmalslernens und der Dimensionsreduktion macht (Bengio et al., 2013). Der Autoencoder wird selten für die Rekonstruktion selbst verwendet: Der Decoder wird verworfen und der Code behalten, sodass eine spätere Lernaufgabe eine Hypothese mit wenigen Modellparameter auf dem Code statt auf den rohen Merkmale lernt (siehe Transfer-Learning).

Verschiedene Autoencoder-Methoden ergeben sich durch unterschiedliche Wahlen für die Hypothesenräume $\mathcal{H}$, $\mathcal{H}^*$ und die Verlustfunktion in der empirischen Risikominimierung des Rekonstruktionsfehlers (1). Der Spezialfall linearer Abbildungen für den Encoder und den Decoder, gepaart mit dem quadratischen Fehlerverlust, lernt denselben Unterraum wie die Hauptkomponentenanalyse (PCA). Bei jedem Minimierer der empirischen Risikominimierung des Rekonstruktionsfehlers (1) ist die Verkettung $\mathbf{\Phi}^* \circ \mathbf{\Phi}$ die Projektion auf die lineare Hülle der ersten d' Hauptkomponenten (Bishop, 2006, Abschn. 12.4.2). Der Code selbst kann aus den Koordinaten bezüglich einer beliebigen Basis dieses Unterraums bestehen, nicht nur bezüglich der orthonormalen Basis, die die PCA liefert. Für manche Anwendungen sind nichtlineare Abbildungen für Encoder und Decoder, die durch künstliche neuronale Netze dargestellt werden, nützlicher, und ein tiefer Autoencoder verwendet auf jeder Seite ein tiefes Netz (Abb. 2). Auf der Kurve aus Abb. 3 hinterlassen ein Code aus einer Zahl und eine verdeckte Schicht auf jeder Seite einen durchschnittlichen quadratischer Verlust von 0.003, gegenüber 0.457 für die beste lineare Abbildung mit demselben Code. Der Gewinn liegt nicht in den künstlichen neuronalen Netzen an sich, sondern in der Form der Rekonstruktion: Lineare Abbildungen mit einem Code aus einer Zahl können die Rekonstruktionen nur auf einer Geraden platzieren, und diese Datenpunkte liegen nicht auf einer solchen.

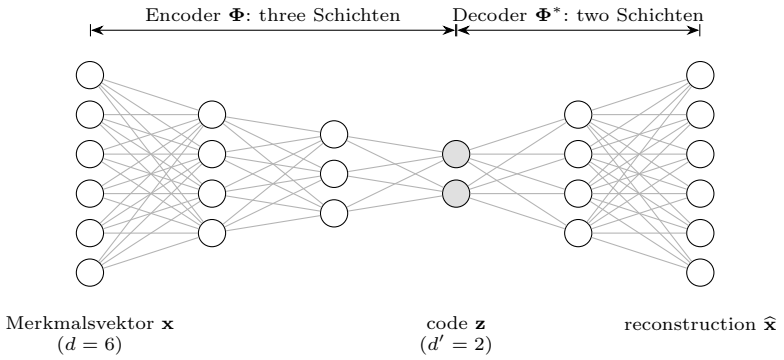


Abbildung 2: Ein tiefer Autoencoder. Der Encoder $\mathbf{\Phi}$ ist ein tiefes Netz aus drei Schichten, das den Merkmalsvektor $\mathbf{x}$, hier $d = 6$ Zahlen, auf einen Code $\mathbf{z}$ aus $d' = 2$ Zahlen (schattiert) abbildet. Der Decoder $\mathbf{\Phi}^*$ stellt den Merkmalsvektor aus diesem Code mit zwei weiteren Schichten wieder her. Der Code ist die schmalste Schicht, sodass alles, was die Rekonstruktion verwendet, durch zwei Zahlen laufen muss

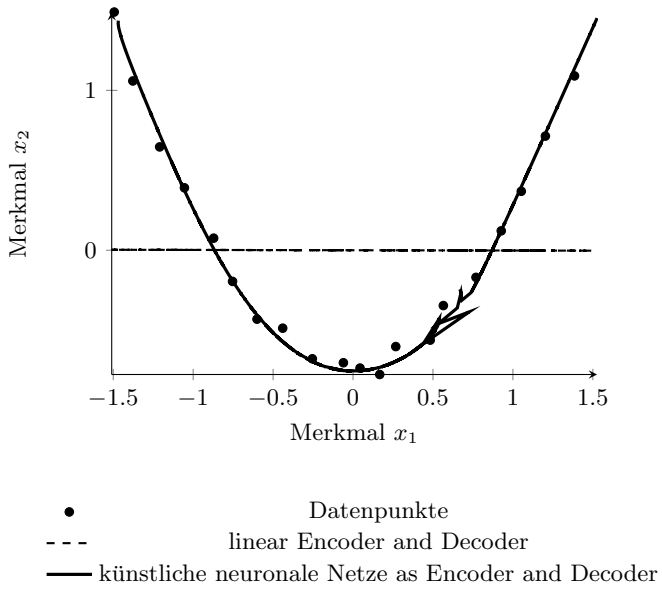


Abbildung 3: Dreihundert Datenpunkte entlang einer Kurve, komprimiert auf einen Code aus einer Zahl und wiederhergestellt; die Punkte zeigen 20 davon, jeden 15-ten. Lineare Abbildungen platzieren jede Rekonstruktion auf einer Geraden (gestrichelte Linie), wobei keine Wahl der Geraden nahe an diese Datenpunkte herankommt; künstliche neuronale Netze folgen der Kurve (durchgezogene Kurve), abgesehen von einer kleinen Schleife nahe $x_1 = 0.6$, die ein Defekt des gelernten Decoders ist, und hinterlassen einen 235-mal kleineren durchschnittlichen quadratischer Verlust. Daten erzeugt mit `pythondemos/autoencoder.py`

Siehe auch: Encoder, Decoder, Repräsentationslernen, Merkmalstransformation, Dimensionsreduktion, PCA, ERM, quadratischer Verlust, selbstüberwachtes Lernen, Transfer-Learning, KNN, tiefes Netz.

Querverweise

Literatur

1. Bengio et al. (2013). Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell., 35(8), pp. 1798–1828.

2. Bishop (2006). Pattern Recognition and Machine Learning. Springer Science+Business Media, New York, NY, USA.

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

4. Hinton and Salakhutdinov (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), pp. 504–507.