Encoder

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 Encoder ist eine Abbildung, die den ursprünglichen Merkmalsvektor eines Datenpunkts auf einen kürzeren Vektor schickt, einen Code aus d' Zahlen mit d' kleiner als der Anzahl d der ursprünglichen Merkmale. Sein Gegenstück ist der Decoder, der einen Code wieder auf einen Vektor der ursprünglichen Länge abbildet, und beide werden gemeinsam beurteilt: Der auf den Code eines Datenpunkts angewandte Decoder soll näherungsweise den Merkmalsvektor zurückliefern, von dem der Encoder ausgegangen ist. Der im Mittel verbleibende Verlust, der Rekonstruktionsfehler, ist das, was das Training minimiert. Die Hauptkomponentenanalyse (PCA) verwendet eine lineare Abbildung als Encoder, eine Matrix, deren Zeilen die zu den größten Eigenwerte gehörenden Eigenvektoren der Stichproben-Kovarianzmatrix sind. Ein Autoencoder lernt den Encoder gemeinsam mit dem Decoder, sodass er ein tiefes Netz statt einer lineare Abbildung sein kann. Ein großes Sprachmodell (LLM) beginnt mit dem Token-zu-Vektor-Encoder, der jedes Token eines Vokabulars auf eine Zeile einer Einbettungs-Matrix abbildet. Die Codelänge entscheidet darüber, was der Decoder wiederherstellen kann, und der anstelle der ursprünglichen Merkmalsvektoren aufbewahrte Code macht einen Encoder zu einer Merkmalstransformation für eine spätere Lernaufgabe. Was ein kürzerer Code verwirft, ist für jeden Decoder verloren, sodass ein auf gute Rekonstruktion gelernter Code dieser Lernaufgabe nicht gut dienen muss.

Definition

Die Wetterstation in Krems an der Donau (Österreich) zeichnet für jeden Tag acht Zahlen auf: die tiefste, die höchste und die mittlere Lufttemperatur, den Niederschlag, die Sonnenscheindauer, die Luftfeuchtigkeit, den Luftdruck und die Windgeschwindigkeit. Zwei aus den acht berechnete Zahlen genügen, um einen Tag im Januar von einem Tag im Juli zu unterscheiden, und eine zweite Abbildung stellt die acht aus den zweien mit kleinem Fehler wieder her. Die Abbildung, welche die zwei Zahlen liefert, ist ein Encoder. Er schickt den ursprünglichen Merkmalsvektor $\mathbf{x} \in \mathbb{R}^d$ eines Datenpunkts auf einen kürzeren Vektor $\mathbf{z} \in \mathbb{R}^{d'}$, seinen *Code*,

$$\mathbf{\Phi} : \mathbb{R}^d \to \mathbb{R}^{d'} : \mathbf{x} \mapsto \mathbf{z} := \mathbf{\Phi}(\mathbf{x}),$$

mit einer Codelänge d' kleiner als der Anzahl d der ursprünglichen Merkmale. Sein Gegenstück ist der Decoder $\mathbf{\Phi}^* : \mathbb{R}^{d'} \to \mathbb{R}^d$, der einen Code wieder auf einen Vektor mit d Einträgen abbildet (Goodfellow et al., 2016, Ch. 14).

Encoder unterscheiden sich im Hypothesenraum, aus dem sie gewählt werden. Die Hauptkomponentenanalyse (PCA) wählt eine lineare Abbildung: Der Encoder ist eine Matrix $\mathbf{W} \in \mathbb{R}^{d' \times d}$ mit $\mathbf{z} = \mathbf{W}\mathbf{x}$, deren Zeilen die zu den d' größten Eigenwerte gehörenden Eigenvektoren der Stichproben-Kovarianzmatrix der Merkmalsvektoren sind, und der Decoder ist die Transponierte $\mathbf{W}^\top$ (Bishop, 2006, Sect. 12.1.2; Hotelling, 1933). Ein Autoencoder wählt den Encoder aus einem Hypothesenraum aus tiefe Netze und lernt ihn gemeinsam mit dem Decoder, sodass keine der beiden Abbildungen linear sein muss (Goodfellow et al., 2016, Sect. 14.1). Ein großes Sprachmodell (LLM) beginnt mit dem Token-zu-Vektor-Schritt, einem Encoder auf Text. Jedes Token eines Vokabulars $\mathcal{V}$ trägt eine Zeile einer Einbettungs-Matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d'}$. Das Auswählen der Zeile des Tokens mit Index v ist die lineare Abbildung $\mathbf{z} = \mathbf{E}^\top \mathbf{e}^{(v)}$, angewandt auf den One-Hot-Vektor $\mathbf{e}^{(v)} \in \mathbb{R}^{|\mathcal{V}|}$ dieses Tokens (Vaswani et al., 2017, Sect. 3.4).

Was einen Code nützlich macht, ist, dass der Decoder den Datenpunkt daraus wiederherstellen kann. Der Rekonstruktionsfehler eines Encoders auf einem Datensatz $\mathcal{D} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ des Stichprobengrößes m ist der mittlere Verlust zwischen einem Merkmalsvektor und dem, was der Decoder für dessen Code zurückliefert,

$$\frac{1}{m} \sum_{r=1}^{m} L\left(\mathbf{x}^{(r)}, \mathbf{\Phi}^*\big(\mathbf{\Phi}\big(\mathbf{x}^{(r)}\big)\big)\right). \tag{1}$$

Eine gängige Verlustfunktion ist der quadratischer Verlust $\|\mathbf{x} - \mathbf{\Phi}^*(\mathbf{\Phi}(\mathbf{x}))\|_2^2$, und die Minimierung des Rekonstruktionsfehlers (1) über einem Hypothesenraum von Encodern und einem zweiten von Decoder ist empirische Risikominimierung (ERM). Es gehen keine Labels ein: Was der Decoder reproduzieren muss, ist der Datenpunkt selbst, sodass das Training eines Encoders eine Form des selbstüberwachtes Lernens ist.

Jeder der acht Messwerte eines Tages in Krems wird auf einen Stichprobenmittel von null und eine Stichprobenvarianz von eins skaliert, sodass sie vergleichbares Gewicht tragen. Über die 366 Tage des Jahres 2024 belässt der PCA-Encoder mit $d' = 2$ dann einen Rekonstruktionsfehler von 2.91 von der gesamten Varianz 8.00, und die acht rekonstruierten Messwerte des 1. Januar 2024 weichen von den skalierten Originalwerten um höchstens 0.80 ab. Abb. 1(a) trägt 21 der Tage an ihren beiden Codezahlen ein, geteilt am Median der Höchsttemperatur, und Abb. 1(b) vergleicht den Rekonstruktionsfehler dieses Encoders mit dem, was ein zufälliger Encoder mit orthonormalen Zeilen bei wachsendem Code belässt.

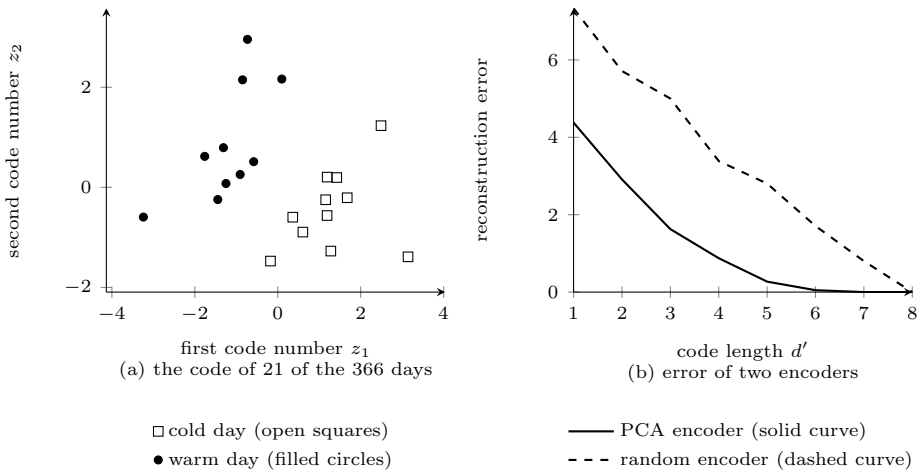


Abbildung 1: Der PCA-Encoder der 366 Tage des Jahres 2024 in Krems, wobei jeder Tag ein Datenpunkt mit acht skalierten Messwerten ist. (a) Die zwei Codezahlen jedes 18 Tages, insgesamt 21, eingezeichnet als kalter Tag (offene Quadrate) oder warmer Tag (ausgefüllte Kreise), je nachdem, ob die Höchsttemperatur unter oder über ihrem Median liegt. (b) Der Rekonstruktionsfehler in Abhängigkeit von der Codelänge d' für den PCA-Encoder (durchgezogene Kurve) und für einen zufälligen Encoder mit orthonormalen Zeilen (gestrichelte Kurve). Daten erzeugt mit `pythondemos/encoder.py`

Die Codelänge entscheidet darüber, was erhalten bleibt. An den Tagen in Krems fällt der Rekonstruktionsfehler des PCA-Encoders von 4.38 bei einer Zahl auf 2.91 bei zweien und 0.88 bei vieren, und bei jeder Codelänge gleicht er der Summe der Eigenwerte, die der Code verwirft (Bishop, 2006, Sect. 12.1.2, Eq. (12.18)). Ein zufälliger Encoder mit orthonormalen Zeilen belässt 5.71 bei zwei Zahlen; welcher Encoder verwendet wird, zählt also neben der Länge seines Codes. Etwas zu verlieren ist nicht unvermeidlich: Der Token-zu-Vektor-Encoder verkürzt einen One-Hot-Vektor der Länge $|\mathcal{V}|$ auf d' Zahlen und verliert dennoch nichts, da verschiedene Tokens verschiedene Zeilen von $\mathbf{E}$ tragen und die nächstgelegene Zeile daher das Token zurückgewinnt. Was nach dem Training behalten wird, ist für gewöhnlich der Encoder allein: Der Decoder wird verworfen und der Code dient als die Merkmalstransformation einer späteren Lernaufgabe, was einen Encoder zu einem Verfahren des Repräsentationslernens und der Dimensionsreduktion macht (Bengio et al., 2013).

Der Code aus zwei Zahlen belässt einen Rekonstruktionsfehler von 2.91, und dieser Verlust verteilt sich nicht gleichmäßig über die acht Messwerte. Von der auf eins normierten Stichprobenvarianz jedes Messwerts behält der Code 96 Prozent der mittleren Temperatur und 3 Prozent der Windgeschwindigkeit. Was die verworfenen Eigenvektoren trugen, ist für jeden Decoder verloren, nicht nur für die Transponierte $\mathbf{W}^\top$. Von den 80 Tagespaaren, deren Codes näher als 0.1 beieinanderliegen, unterscheiden sich der 14. Februar und der 22. Novem-

ber 2024 am stärksten in der Windgeschwindigkeit, 1.0 gegenüber 4.5 Metern pro Sekunde, und der Decoder liefert für beide 1.76 Meter pro Sekunde zurück. Ob der Verlust ins Gewicht fällt, entscheidet die spätere Lernaufgabe und nicht der Rekonstruktionsfehler allein. Ein Tag gilt als windig, wenn seine mittlere Windgeschwindigkeit über dem Median der 366 Werte liegt, was auf 130 davon zutrifft. Ein mittels kleinster Quadrate gelernter linearer Klassifikator erreicht eine Genauigkeit von 98 Prozent aus allen acht skalierten Messwerten und 66 Prozent aus den zwei Codezahlen, gegenüber den 64 Prozent, die man erreicht, wenn man jeden Tag als windstill einstuft. Ein einziges neues, anhand der Labels statt aus der Stichproben-Kovarianzmatrix gewähltes Merkmal, die lineare Diskriminanzanalyse (LDA)-Richtung, erreicht ebenfalls 98 Prozent. Ein auf gute Rekonstruktion gelernter Code ist daher kein auf gute Vorhersage gelernter Code, worin der Unterschied liegt, den die LDA gegenüber der PCA zieht. Ein Encoder liegt zudem fest, sobald er gelernt ist, und der Datensatz, aus dem er gelernt wurde, entscheidet darüber, wie gut er einen späteren Tag codiert. Der PCA-Encoder der 183 Tage von Januar bis März sowie von Oktober bis Dezember belässt 1.93 auf diesen Tagen und 10.87 auf den anderen 183, gegenüber 2.48 für den Encoder des gesamten Jahres. Was der Code erkauft, ist seine Kürze: weniger Zahlen pro Datenpunkt, die gespeichert und an eine spätere Lernaufgabe übergeben werden müssen, und bei $d' = 2$ eine Darstellung des Datensatzes auf zwei Achsen.

Siehe auch: Decoder, Autoencoder, variationaler Autoencoder (VAE), Repräsentationslernen, Merkmalstransformation, Dimensionsreduktion, PCA, LDA, Einbettung, Token, latente Variable, ERM, quadratischer Verlust.

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. Hotelling (1933). Analysis of a Complex of Statistical Variables into Principal Components. Journal of Educational Psychology, 24(6), pp. 417–441.

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