Pipeline für maschinelles Lernen (ML-Pipeline)

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

Eine Pipeline für maschinelles Lernen (ML-Pipeline) ist die Zusammensetzung der Funktionen, durch die ein ML-System (ML-System) rohe Daten in Vorhersagen umwandelt. Ihre Stufen sind typischerweise die Daten preprocessing, Repräsentationslernen oder handentworfene Merkmal-Extraktion, das Modell Training und die inference auf neuen Datenpunkte. Jede Stufe ist eine Funktion, deren Ausgabe der Input der nächsten ist, sodass die gesamte Pipeline eine einzige Abbildung von rohen Daten zu Vorhersagen darstellt und während des Training und im Einsatz identisch angewendet wird. Das Anpassen von Stufen wie der Merkmal-Skalierung innerhalb der Pipeline, nur auf dem Trainingsdatensatz, verhindert Data Leakage vom Validierungsdatensatz oder Testdatensatz.

Definition

Die Wetterstation Krems an der Donau meldet täglich acht Messwerte an den Datenhub der GeoSphere Austria. Diesen Datenstrom in eine Vorhersage der morgigen Temperatur um 06:00 (UTC) umzuwandeln, erfordert vier Schritte: Bereinigung der Aufzeichnungen, Extraktion von Merkmale, Training eines Modell und Berechnung der Vorhersage. Die Abfolge solcher Schritte ist eine ML-Pipeline: eine Komposition (d.h. Verkettung) mehrerer Funktionen innerhalb eines ML-System (ML-System). Die einzelnen Funktionen umfassen die Daten preprocessing, Repräsentationslernen oder handentworfene Merkmal-Extraktion, das Modell Training und die inference auf neuen Datenpunkte. Durch deren Kombination wandelt ein ML-System rohe Daten in Vorhersagen um (Hapke and Nelson, 2020) (Abb. 1).

Mathematisch ausgedrückt ist eine Pipeline aus drei Stufen die Verkettung

$$h = h^{(3)} \circ h^{(2)} \circ h^{(1)}, \tag{1}$$

wobei $h^{(1)}$ die Merkmalsabbildung ist, $h^{(2)}$ die auf die resultierenden Merkmale angewendete Hypothese und $h^{(3)}$ die Nachbearbeitung ihres Ausgabe. Von den vier oben genannten Schritten bilden die Bereinigung und Merkmal-Extraktion $h^{(1)}$, die trainierte Hypothese ist $h^{(2)}$ (das Training ist die Art und Weise, wie sie ausgewählt wird, kein Faktor von h), und die Nachbearbeitung der Vorhersage ist $h^{(3)}$. Jeder Faktor wird aus einem eigenen Hypothesenraum gewählt, $h^{(j)} \in$

$\mathcal{H}^{(j)}$ für $j = 1, 2, 3$. Die Verkettungen (1) sind selbst ein Hypothesenraum,

$$\mathcal{H} = \left\{ h^{(3)} \circ h^{(2)} \circ h^{(1)} \ : \ h^{(j)} \in \mathcal{H}^{(j)}, \ j = 1, 2, 3 \right\},$$

sodass das Training der Pipeline empirische Risikominimierung (ERM) über $\mathcal{H}$ ist und nicht über eine einzelne Stufe.

Jede Stufe ist eine Funktion, deren Ausgabe der Input der nächsten ist, sodass die Pipeline eine einzige Abbildung von rohen Daten zu Vorhersagen darstellt, die während des Training und im Einsatz identisch angewendet wird. Stufen mit eigenen Parameter, wie die Merkmal-Skalierung, werden nur auf dem Trainingsdatensatz angepasst. Das Anpassen dieser Stufen an alle verfügbaren Daten lässt Informationen vom Validierungsdatensatz oder dem Testdatensatz in die gelernte Hypothese sickern, sodass der Validierungsfehler oder der Fehler auf dem Testdatensatz das Risiko unterschätzt (siehe Data Leakage) (Hastie et al., 2009, Sect. 7.10.2). Softwarebibliotheken bündeln daher die Stufen zu einem Objekt, das als Ganzes auf dem Trainingsdatensatz trainiert wird (Pedregosa et al., 2011). Die Pipeline ist der rechnerische Teil eines ML-System, das zusätzlich die Daten-Speicherung, den Einsatz und das Monitoring umfasst.

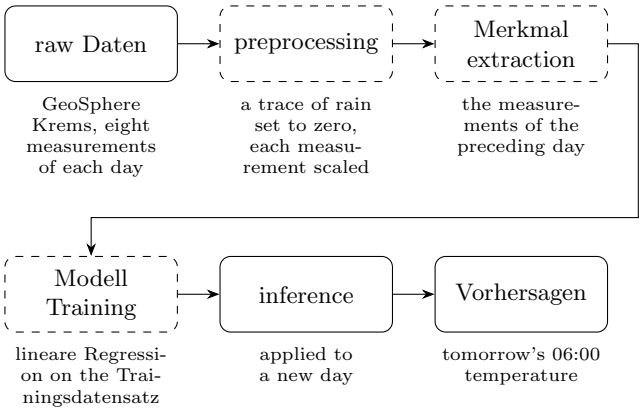


Abbildung 1: Eine ML-Pipeline als Verkettung von Funktionen (1), wobei der Ausgabe jeder Stufe der Input der nächsten ist, mit der im Text genannten Krems-Vorhersage unter jeder Stufe. Die drei mit gestrichelter Linie gezeichneten Stufen tragen eigene Parameter und werden nur auf dem Trainingsdatensatz angepasst, was Data Leakage vom Validierungsdatensatz oder Testdatensatz in die gelernte Hypothese verhindert.

Siehe auch: Funktion, ML-System, preprocessing, Repräsentationslernen, inference, Data Leakage, Trainingsdatensatz, Hypothesenraum, ERM.

Querverweise

Literatur

1. Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12(85), pp. 2825–2830.

2. Hapke and Nelson (2020). Building Machine Learning Pipelines: Automating Model Life Cycles with TensorFlow. O'Reilly Media, Sebastopol, CA, USA.

3. Hastie et al. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed., Springer Science+Business Media, New York, NY, USA.