Einführung in MLOps

17 Model Drift

Tobias Mérinat
teaching2025@fsck.ch

Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN

Department of Information Technology
Lucerne University of Applied Sciences and Arts
6343 Rotkreuz, Switzerland

14. und 15. Februar 2025

Fehler in Software ohne ML

Software ohne ML failt aufgrund von

- Fehler in der Programmlogik (Bug, falsche Requirements)
- Fehlern in Up- oder Downstream-Systemen (Bugs, Interface-Changes)
- Infrastrukturproblemen (Downtime, Fehlkonfiguration, Netzwerkproblem).

Software failt

- mit einer Exception
- still

Fehler in Software mit ML

Software mit ML kann auf eine zusätzliche Art failen:

- System läuft (gleich wie bisher) korrekt (keine der obigen Ursachen)
- aber die Performance nimmt trotzdem ab

Ursache für die Performance-Abnahme

- Grundannahme in Machine Learning: *Unseen Data* kommt aus derselben (stationären) Verteilung wie die Trainingsdaten
- Modellentwicklung:
  - fixes Trainingsset
  - korrekte Cross Validation
- Modell = Snapshot der Wirklichkeit zum Zeitpunkt des Trainings
- Die Welt ist aber nicht statisch, sie ändert sich, womit sich auch die Inputdaten für ein Modell ändern können

Model Drift / Shift / Skew

- **Model Drift**: Der Umstand, dass die Vohersagequalität eines ML- Modells im Laufe der Zeit schlechter werden kann, obwohl sich an dem Modell an sich nichts verändert.
- Synonym: *shift* oder kurz *drift*
- *Model* Drift ist ein unglücklich gewählter Begriff, da sich das Model nicht verändert, sondern dessen Inputdaten
- **Skew**: Unterschied zweier Verteilungen, speziell *Training-Serving-Skew*.
- *Skew* ist der Unterschied, *Drift* die (schleichend) zunehmende Abweichung
- Grundsätzlich: Wenn sich die Verteilung der Daten zum Zeitpunkt der Inferenz von derer zum Zeitpunkt der Trainings abweicht

Drift kann ein grosses Problem darstellen, wenn wir ML in der realen Welt einsetzen, in der Daten oft dynamisch und ständig im Wandel sind.

Grundlagen

- X sind die Inputs eines Modells, Y sind dessen Outputs
- In Supervised Learning kommen die Trainingsdaten aus der gemeinsamen Wahrscheinlichkeitsverteilung $P(X, Y)$.
- Modelliert wird $P(Y|X)$
- Für die gemeinsame Verteilung gilt
  - $P(X, Y) = P(Y|X)P(X)$
  - $P(X, Y) = P(X|Y)P(Y)$
- Dabei ist
  - $P(Y|X)$ die bedingte Wahrscheinlichkeit eines Outputs, gegeben den Input
  - $P(X)$ die Wahrscheinlichkeit der Inputs
  - $P(Y)$ die Wahrscheinlichkeit der Outputs

Lucerne University of Applied Sciences and Arts
HOCHSCHULE LUZERN

Drei Arten von Drift (Shift)

Drift wird in drei Haupt-Kategorien aufgeteilt, es können Mischformen auftreten:

- **Covariate Shift** $P(X)$ ändert, aber $P(Y|X)$ bleibt gleich
- **Label Shift** $P(Y)$ ändert, aber $P(X|Y)$ bleibt gleich
- **Concept drift** $P(Y|X)$ ändert, aber $P(X)$ bleibt gleich

Lucerne University of Applied Sciences and Arts
HOCHSCHULE LUZERN

Covariate Shift: $P(X)$ ändert, $P(Y|X)$ bleibt gleich

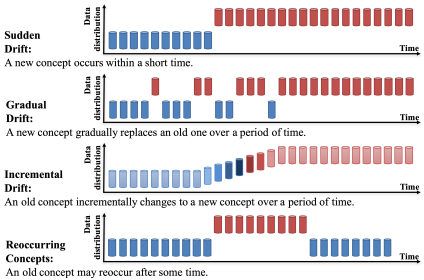
- Die Inferenz-Daten sind anders als die Trainingsdaten
- Kann während Trainingsphase eingeführt werden, z.B. durch Selection Bias
  - Oversampling eines Imbalanced Datensets
  - Active Learning
- Oft ausgelöst durch eine substantielle Veränderung der Umgebung oder der Art, wie ein Produkt verwendet wird.
  - App mit Feature *Alter* wird von einer anderen Altersgruppe verwendet, als durch die Trainingsdaten gegeben war
  - oder in einem anderen Land
  - oder durch andere Sensoren, neuere und höherauflösende Kameras, oder mit Noise im Hintergrund, während Trainingsdaten in einem isolierten Raum aufgezeichnet wurden.

Label Shift: $P(Y)$ ändert, $P(X|Y)$ bleibt gleich

- Auch als *prior shift* oder *prior probability shift* oder *target shift* bezeichnet
- Verteilung der Zielvariablen ändert sich, aber für ein gegebenes Label bleibt die Input-Verteilung gleich
- Covariate Shift and Label Shift treten oft, aber nicht immer, gemeinsam auf

Concept Drift: $P(Y|X)$ ändert, $P(X)$ bleibt gleich

- Auch als *posterior shift* bezeichnet
- *Same input, different output*
- Saisonale Effekte spielen oft eine Rolle. Höhere Preise am Wochendende wie unter der Woche.
- Kann schleichend oder plötzlich kommen



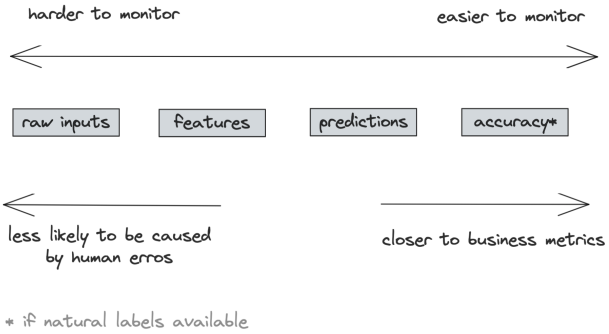
Konsequenz

- Inputdaten und Modellperformance muss überwacht werden
- Modelle müssen periodisch neu trainiert werden

Drift überwachen

Wir können vier Stufen von Drift Monitoring unterscheiden

1. Raw Data
2. Features
3. Predictions
4. Metriken



Metriken (Natural Labels)

- Um Metriken direkt zu messen, benötigen wir *Natural Labels*
- Wir speichern die Predictions aus der Inferenz und berechnen und prüfen periodisch
- Je schneller die Labels anfallen, desto scheller können wir reagieren
- Neben Natural Labels sollte aber auch jegliches anderes Feedback geloggt werden, welches als Proxy für ein Natural Label dienen könnte.
  - Clicks auf Recommendations
  - Bookmarks
  - Upvotes
  - Shares
  - Likes
  - Klick auf *skip* bzw. Abbruch (Spotify, Youtube)
  - Nutzungsdauer
  - . . .
- Entweder können natural labels abgeleitet werden oder sonst verwenden, um auftretende Veränderungen zu erkennen

Lucerne University of Applied Sciences and Arts
HOCHSCHULE LUZERN

Predictions, Features und Rohdaten (Verteilungen)

- Haben wir keine Ground Truth, vergleichen wir Verteilungen aktueller Daten mit Verteilungen von Referenzdaten
  1. Wir bilden Windows der Inference Requests
  2. berechnen die Verteilung eines Window
  3. verwenden eine Metrik, die berechnete Verteilung mit einem Referenzdatensatz zu vergleichen
  4. speichern die berechneten Metriken als Zeitreihe, um Trends zu erkennen
- Features: einfacher, weil Schema; Möglichkeit, neben Endwerten auch Zwischenstufen zu monitoren
- Rohdaten: gleiches Vorgehen, aufgrund oft fehlender Struktur jedoch etwas schwieriger
- Predictions: Fehlt ein Natural Label, kann man auch Veränderungen an der Verteilung der Predictions messen

Drift addressieren

Beeinträchtigt der Drift die Modell-Performance zu stark, muss das Modell neu trainiert werden.

- Nur mit frischen Daten trainieren (wenn notwendig labeln)
- Mit alten und neuen Daten trainieren
- Testen, welche alten Daten noch gut sind und behalten werden können
- Evaluieren und mit altem Modell vergleichen

Breit monitoren

- Generell möglichst viel monitoren, und laufend anpassen
- Brainstormen, was alles schief gehen könnte, dann entsprechende Metriken definieren
  - Operative Metriken: Memory, CPU-Auslastung, Latenz, Throughput, Server Load
  - Statistical Health der Eingangsdaten
    - Anzahl NaNs, Average Image Brightness
  - Statistical Health der Outputs
    - Verteilung, Anzahl Nulls, Anzahl True in a row
  - Benutzer-Verhalten: CTR, mit Content verbrachte Zeit, Likes, Shares, Upvotes, . . .
  - Performance auf einem speziell wichtigen Daten-Subset (als separate Metrik)

Schema Changes

- Datentypen
- Daten-Ranges (min, max oder Werte-Set)
- Regexps
- Not Null
- Key Constraints