Filter

overfitting (105 min)

Advanced ML models
- Neural networks (75 min)
- Embeddings (45 min)
- Large language models (LLMs) (45 min)

Real-world ML
- Production ML systems (70 min)
  - Introduction (2 min)
  - Static vs. dynamic training (10 min)
  - Static vs. dynamic inference (10 min)
  - When to transform data? (3 min)
  - Deployment testing (5 min)
  - Monitoring pipelines (15 min)
  - Questions to ask (10 min)
  - Test your knowledge (15 min)
  - What's next
- Automated machine learning (30 min)
- Fairness (110 min)

Production ML systems: When to transform data?

Raw data must be feature engineered (transformed). When should you transform data? Broadly speaking, you can perform feature engineering during either of the following two periods:

- *Before* training the model.
- *While* training the model.

Transforming data before training

In this approach, you follow two steps:

1. Write code or use specialized tools to transform the raw data.
2. Store the transformed data somewhere that the model can ingest, such as on disk.

Advantages

- The system transforms raw data only once.
- The system can analyze the entire dataset to determine the best transformation strategy.

Disadvantages

- You must recreate the transformations at prediction time. Beware of **training-serving skew**!

Training-serving skew is more dangerous when your system performs dynamic (online) inference. On a system that uses dynamic inference, the software that transforms the raw dataset usually differs from the software that serves predictions, which can cause training-serving skew. In contrast, systems that use static (offline) inference can sometimes use the same software.

Transforming data while training

In this approach, the transformation is part of the model code. The model ingests raw data and transforms it.

Advantages

- You can still use the same raw data files if you change the transformations.
- You're ensured the same transformations at training and prediction time.

Disadvantages

- Complicated transforms can increase model latency.
- Transformations occur for each and every batch.

Transforming the data per batch can be tricky. For example, suppose you want to use **Z-score normalization** to transform raw numerical data. Z-score normalization requires the mean and standard deviation of the feature. However, transformations per batch mean you'll only have access to *one batch of data*, not the full dataset. So, if the batches are highly variant, a Z-score of, say, -2.5 in one batch won't have the same meaning as -2.5 in another batch. As a workaround, your system can precompute the mean and standard deviation across the entire dataset and then use them as constants in the model.

Key terms:

- Training-serving skew
- Z-score normalization

Last updated 2024-10-09 UTC.