BITS Pilani
Pilani | Dubai | Goa | Hyderabad

ML Monitoring - Metrics and Toolbox

Pravin Y Pawar

Adapted from “Designing Machine Learning Systems”
By Chip Huyen

ML Monitoring

- As the industry realized that many things can go wrong with an ML system,
  - many companies started investing in monitoring and observability for their ML systems in production

- Monitoring refers to the act of tracking, measuring, and logging different metrics that can help us determine when something goes wrong
- Observability means setting up our system in a way that gives us visibility into system to help us investigate what went wrong

- Monitoring is all about metrics!
  - Because ML systems are software systems, the first class of metrics need to monitor are the operational metrics
  - designed to convey the health of systems

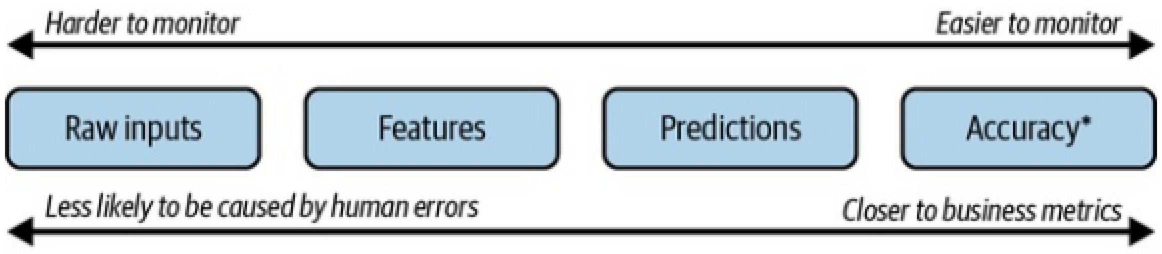
- Generally divided into three levels:
  - the network the system is run on,
  - the machine the system is run on,
  - and the application that the system runs

- Examples
  - latency; throughput; the number of prediction requests model receives in the last minute, hour, day;
  - the percentage of requests that return with a 2xx code;
  - CPU/GPU utilization; memory utilization; etc.

ML-Specific Metrics

- Within ML-specific metrics, there are generally four artifacts to monitor:
  - a model’s accuracy-related metrics,
  - predictions,
  - features,
  - and raw inputs

- Artifacts generated at four different stages of an ML system pipeline
  - The deeper into the pipeline an artifact is, the more transformations it has gone through,
    - which makes a change in that artifact more likely to be caused by errors in one of those transformations
  - However, the more transformations an artifact has gone through, the more structured it’s become
    - closer it is to the metrics actually care about, which makes it easier to monitor



Monitoring accuracy-related metrics

- Accuracy-related metrics are the most direct metrics to help you decide whether a model's performance has degraded

- If system receives any type of user feedback for the predictions it makes
  - click, hide, purchase, upvote, downvote, favorite, bookmark, share, etc.
  - should definitely log and track it

- Some feedback can be used to infer natural labels,
  - which can then be used to calculate model's accuracy-related metrics

- Feedback can be used to detect changes in ML model's performance
  - For example, when building a system to recommend to users what videos to watch next on YouTube,
  - want to track not only whether the users click on a recommended video (click-through rate),
  - but also the duration of time users spend on that video and whether they complete watching it (completion rate)
  - If, over time, the clickthrough rate remains the same but the completion rate drops, it might mean that recommender system is getting worse

- Possible to engineer system to collect users' feedback
  - For example, Google Translate has the option for users to upvote or downvote a translation

Monitoring predictions

- Prediction is the most common artifact to monitor
    - If it’s a regression task, each prediction is a continuous value (e.g., the predicted price of a house)
    - If it’s a classification task, each prediction is a discrete value corresponding to the predicted category

- Each prediction is usually just a number (low dimension), predictions are easy to visualize
    - summary statistics are straightforward to compute and interpret

- Can monitor predictions for distribution shifts as they are low dimensional
    - Easier to compute two-sample tests to detect whether the prediction distribution has shifted
    - Prediction distribution shifts are also a proxy for input distribution shifts

- Can also monitor predictions for anything odd happening
    - such as predicting an unusual number of False in a row

Monitoring features

- ML monitoring solutions in the industry focus on tracking changes in features, both
- the features that a model uses as inputs
  - the intermediate transformations from raw inputs into final features

- Feature monitoring is appealing because
  - compared to raw input data, features are well structured following a predefined schema

- The first step of feature monitoring is feature validation:
  - ensuring that features follow an expected schema

- Things can be check for a given feature:
  - If the min, max, or median values of a feature are within an acceptable range
  - If the values of a feature satisfy a regular expression format
  - If all the values of a feature belong to a predefined set
  - If the values of a feature are always greater than the values of another feature

- Many open source libraries that help in basic feature validation,
  - Two most common are Great Expectations and Deequ, which is by AWS

Monitoring features(2)

Four major concerns when doing feature monitoring

- A company might have hundreds of models in production, and each model uses hundreds, if not thousands, of features.
  - Even something as simple as computing summary statistics for all these features every hour can be expensive,
  - not only in terms of compute required but also memory used
  - Tracking, i.e., constantly computing, too many metrics can also slow down system
  - increase both the latency that users experience

- While tracking features is useful for debugging purposes, it's not very useful for detecting model performance degradation
  - In practice, an individual feature's minor changes might not harm the model's performance at all
  - Feature distributions shift all the time, and most of these changes are benign
  - If want to be alerted whenever a feature seems to have drifted, might soon be overwhelmed by alerts
  - realize that most of these alerts are false positives -"alert fatigue"

- Feature extraction is often done in multiple steps (such as filling missing values and standardization),
  - using multiple libraries (such as pandas, Spark),
  - on multiple services (such as BigQuery or Snowflake).
- Even if its detected a harmful change in a feature, it might be impossible to detect whether this change is
  - caused by a change in the underlying input distribution or whether it's caused by an error in one of the multiple processing steps

- The schema that features follow can change over time.
  - If don't have a way to version schemas and map each of features to its expected schema,
  - the cause of the reported alert might be due to the mismatched schema rather than a change in the data

Monitoring raw inputs

- A change in the features might be caused by problems in processing steps and not by changes in data
  - can monitor the raw inputs before they are processed
  - not be easier to monitor, as it can come from multiple sources in different formats, following multiple structures

- The way many ML workflows are set up today also makes it impossible for ML engineers to get direct access to raw input data,
  - as the raw input data is often managed by a data platform team who processes and moves the data to a location like a data warehouse,
- ML engineers can only query for data from that data warehouse where the data is already partially

processed

- Monitoring raw inputs is often a responsibility of the data platform team, not the data science or ML team

Monitoring Toolbox

- Measuring, tracking, and interpreting metrics for complex systems is a nontrivial task
  - engineers rely on a set of tools to help them do so

- Common for the industry to herald metrics, logs, and traces as the three pillars of monitoring

- Seem to be generated from the perspective of people who develop monitoring systems:
  - traces are a form of logs and metrics can be computed from logs

- Focus on the set of tools from the perspective of users of the monitoring systems:
  - logs,
  - dashboards,
  - alerts

Monitoring Toolbox(2)

Logs

- Traditional software systems rely on logs to record events produced at runtime
  - An event is anything that can be of interest to the system developers,
  - either at the time the event happens or later for debugging and analysis purposes

- Examples of events
  - a container starts,
  - the amount of memory it takes,
  - when a function is called,
  - when that function finishes running,
  - the other functions that this function calls,
  - the input and output of that function,
  - log crashes, stack traces, error codes, and more.

- The number of logs can grow very large very quickly.
  - need to query your logs for the sequence of events that caused it, a process that can feel like searching for a needle in a haystack

- Because logs have grown so large and so difficult to manage,
  - there have been many tools developed to help companies manage and analyze logs
  - The log management market is estimated to be worth USD 2.3 billion in 2021, and it’s expected to grow to USD 4.1 billion by 2026

Monitoring Toolbox(3)

Dashboards

- A picture is worth a thousand words!
  - A series of numbers might mean nothing, but visualizing them on a graph might reveal the relationships among these numbers
  - Dashboards to visualize metrics are critical for monitoring

- Another use of dashboards is to make monitoring accessible to non-engineers.
  - Monitoring isn’t just for the developers of a system, but also for non-engineering stakeholders
    - including product managers and business developers

- Dashboard rot
  - Excessive metrics on a dashboard can also be counterproductive
    - important to pick the right metrics
    - abstract out lower-level metrics to compute higher-level signals that make better sense for specific tasks

Monitoring Toolbox(4)

Alerts

- When monitoring system detects something suspicious, it’s necessary to alert the right people about it

- An alert consists of the following three components:
- An alert policy
  - describes the condition for an alert
  - might want to create an alert when a metric breaches a threshold, optionally over a certain duration
- Notification channels
  - describe who is to be notified when the condition is met
- A description of the alert
  - helps the alerted person understand what’s going on with a detailed description
  - often necessary to make the alert actionable
  - by providing mitigation instructions or a runbook, a compilation of routine procedures and operations that might help with handling the alert

- Alert fatigue is a real phenomenon!
  - can be demoralizing—nobody likes to be awakened in the middle of the night for something outside of their responsibilities
  - can be dangerous —being exposed to trivial alerts can desensitize people to critical alerts
  - important to set meaningful conditions so that only critical alerts are sent out

Thank You!

In our next session: