Objectives of today's lecture

- review data validation for machine learning
- review data validation/cleaning in databases

Section 1. Data Validation for Machine Learning

Goal: Focus on how to validate the input data fed to ML pipelines
What does this mean?

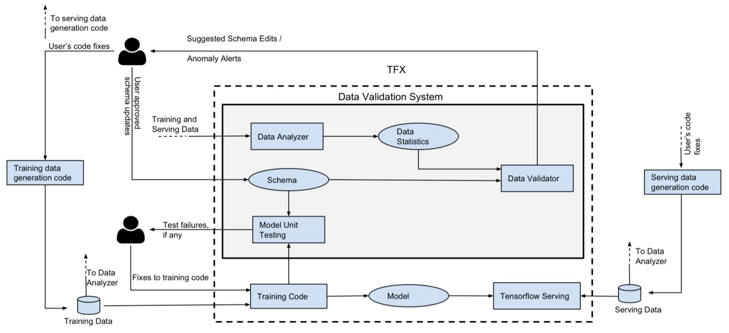
ML models in production are training daily on batches of data (remember the online deployment settings). These features can be generated from queries that operate on state data (no real-time). What if the output of the query is “-1” meaning invalid value because someone pushed this new code change?

Elements to validate:

- Training on new batches
- Serving on new batches (model evaluation)
- Evaluating the validity of the model

Requirement we need a pipeline: run in a continuous fashion with the arrival of a new batch of data triggering a new run.
Ingest the training data, validate it, send it to the training algorithm, deploy the new trained models

Section 1.1 Integrating data validation in ML pipelines



The pipeline has three main components:
- The data analyzer (computes statistics over the new data batch)
- The data validator (checks properties of the data against a schema)
- Model unit tester (looks for errors in the training code using synthetic data)

Section 1.2. Single batch validation

Core question: does this data batch contains anomalies? (Remember the definition of an anomaly?) Statistical outliers, errors based on domain-specific constraints etc.

Given the schema of a dataset we can enforce several types of constraints: domain values, domain cardinalities, domain properties (min, max values)

Testing these constraints against the batch becomes a system’s exercise.

All these are domain knowledge (we will come back to this later)

There are also statistical safeguards against anomalies, e.g., distributional skew (between serving and training data). We want across batches validation.

Section 1.3. Across batches validation

Every batch adheres to syntactic constraints (it contains valid data). We now need to ensure that the statistical properties across batches are "consistent".

We want to detect skew across batches:

- Feature skew (particular features assumes different values in training vs serving data). Example: differences in elapsed time.
- Distribution skew (we have drift between today's and yesterday's data).
- Scoring/serving skew: skewed labels from the user and feedback.

To detect distribution skew we can use different distribution distance metrics:

KL divergence: $D_{KL}(P\|Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$

Total variation distance: $\delta(P,Q) = \sup_{A \in F} (P(A) - Q(A)$ we have $\delta(P,Q) \leq \sqrt{\frac{1}{2} D_{KL}(P\|Q)}$

Wasserstein distance: $W_p(\mu,\nu) = \left(\inf E[d(X,Y)^p] \right)^{1/p}$ (minimum cost required to turn one distribution to the other)

Consideration: do engineers understand these metrics?

Distance measure: largest change in probability for any single value in the two distributions (feature specific and strong independence assumption).

Section 1.4. Model unit testing

Not validation of the incoming data but validation of the training code (can it handle the variety of data it may see?)
The training code may apply transformations over feature values (say a logarithm): do we still get a valid output?
Use the schema to generate synthetic variations of the data to test robustness.

Section 2. Formal Data Validation Models in Databases

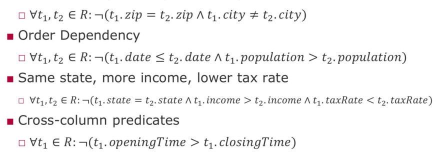
Many connections to formal models of data validation. We will rely on the following slides for this lecture:
http://pages.cs.wisc.edu/~thodrek/MLND.pdf

Section 2.1. Integrity Constraints

Integrity constraints provide a way of ensuring that changes made to the database do not result in a loss of data consistency.
- Key constraints (primary keys need to be respected)
- Constraints over relationships (1-1, 1-many) etc.
- Domain constraints

Relational Constraints:
The easiest type is that of Functional Dependencies (A constraint on the set of legal relations in a database)
They allow us to express facts about the real world we are modeling.
A -> B means that for all pairs of tuples t1 and t2 in a database relation r such that t1[A] = t2[A] it is also the case that t1[B] = t2[B] (A can be a vector)

There are other types of such relation constraints such as Conditional FDs (where we extend the implication with additional logical predicates) and the generalization of denial constraints (negated conjunction)



- Functional Dependency ZIP → City
  - $\forall t_1, t_2 \in R: \neg(t_1.zip = t_2.zip \wedge t_1.city \neq t_2.city)$
- Order Dependency
  - $\forall t_1, t_2 \in R: \neg(t_1.date \leq t_2.date \wedge t_1.population > t_2.population)$
- Same state, more income, lower tax rate
  - $\forall t_1, t_2 \in R: \neg(t_1.state = t_2.state \wedge t_1.income > t_2.income \wedge t_1.taxRate < t_2.taxRate)$
- Cross-column predicates
  - $\forall t_1 \in R: \neg(t_1.openingTime > t_1.closingTime)$

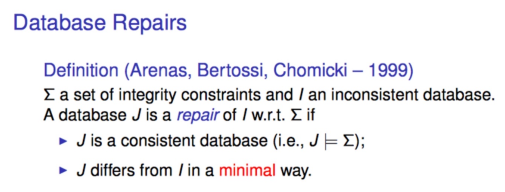
Constraint examples

What is a noisy dataset and what does data validation for a dataset mean?
Go over example in slides 13-14

main point: data validation in databases = error detection + data repairs

Section 2.2. Constraint-based repairs (Minimality)

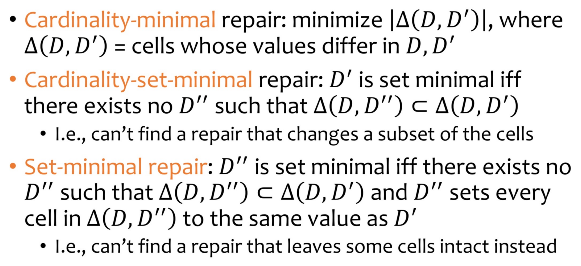
How can we formalize data repairs?



Database Repairs

Definition (Arenas, Bertossi, Chomicki – 1999)
Σ a set of integrity constraints and I an inconsistent database.
A database J is a *repair* of I w.r.t. Σ if

- J is a consistent database (i.e., $J \models \Sigma$);
- J differs from I in a minimal way.



- Cardinality-minimal repair: minimize $|\Delta(D, D')|$, where $\Delta(D, D')$ = cells whose values differ in D, D'
- Cardinality-set-minimal repair: D' is set minimal iff there exists no D'' such that $\Delta(D, D'') \subset \Delta(D, D')$
  - I.e., can't find a repair that changes a subset of the cells
- Set-minimal repair: D'' is set minimal iff there exists no D'' such that $\Delta(D, D'') \subset \Delta(D, D')$ and D'' sets every cell in $\Delta(D, D'')$ to the same value as D'
  - I.e., can't find a repair that leaves some cells intact instead

Section 2.3. Probabilistic Unclean Databases

Go over slides 29 to 41 in talk

Section 3. HoloClean

Go over slides 50 to 56 and describe the high-level ideas of probabilistic repairs in HoloClean.