Software Engineering for ML/AI

Claire Le Goues  Michael Hilton

Christopher Meiklejohn
Administrivia

• Homework 2 (Code Artifacts) due today.
Learning goals

• Identify differences between traditional software development and development of ML systems.
• Understand the stages that comprise the typical ML development pipeline.
• Identify challenges that must be faced within each stage of the typical ML development pipeline.
Participation Survey

• **YES** in Zoom:
  “I’ve a taken machine learning course.”

• **NO** in Zoom:
  “I have not taken a machine learning course.”
Software Engineering and ML
Traditional Software Development

“It is easy. You just chip away the stone that doesn’t look like David.” – (probably not) Michelangelo
ML Development

• Observation
• Hypothesis
• Predict
• Test
• Reject or Refine Hypothesis
Microsoft’s view of Software Engineering for ML

- Model Requirements
- Data Collection
- Data Cleaning
- Data Labeling
- Feature Engineering
- Model Training
- Model Evaluation
- Model Deployment
- Model Monitoring
Three Fundamental Differences:

• Data discovery and management

• Customization and Reuse

• No incremental development of model itself
Case study developed by
Christian Kästner

https://ckaestne.github.io/seai/

CASE STUDY
WHAT CHALLENGES ARE THERE IN BUILDING AND DEPLOYING ML?
cash only
Qualities of Interest?
Qualities of Interest?
MACHINE LEARNING PIPELINE
Typical ML Pipeline

- **Static**
  - Get labeled data (data collection, cleaning and, labeling)
  - Identify and extract features (feature engineering)
  - Split data into training and evaluation set
  - Learn model from training data (model training)
  - Evaluate model on evaluation data (model evaluation)
  - Repeat, revising features

- **with production data**
  - Evaluate model on production data; monitor (model monitoring)
  - Select production data for retraining (model training + evaluation)
  - Update model regularly (model deployment)
# Example Data

<table>
<thead>
<tr>
<th>UserId</th>
<th>PickupLocation</th>
<th>TargetLocation</th>
<th>OrderTime</th>
<th>PickupTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>....</td>
<td>...</td>
<td>18:23</td>
<td>18:31</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example Data
Learning Data

parallel corpus

The corporation has been estimated to run more than one million pag in data centers around the world to process over one billion ser requests and about twenty-four i of user-generated data each day.

December 2012 Alexa listed as

monolingual corpus

started functioning in 1928 and established the tradition of large exhibitions and trade fairs held in Brno, and nowadays also ranks among the sights of the city. Brno is also known for hosting big motorbike and other races on the Masaryk Circuit, a tradition established in 1930 in which the Road Racing World Championship Grand Prix is one of the most prestigious races. Another notable cultural tradition is an international fireworks competition.
Feature Engineering

• Identify parameters of interest that a model may learn on
• Convert data into a useful form
• Normalize data
• Include context
• Remove misleading things
• In OCR/translation:
Features?

現金のみ

cash only
Features?
Feature Extraction

• In surge prediction:
  o Location and time of past surges
  o Events
  o Number of people traveling to an area
  o Typical demand curves in an area
  o Demand in other areas
Data Cleaning

• Removing outliers
• Normalizing data
• Missing values
• ...
Learning

• Build a predictor that best describes an outcome for the observed features
Evaluation

• Prediction accuracy on learned data vs prediction accuracy on unseen data
  o Separate learning set, not used for training

• For binary predictors: false positives vs. false negatives, precision vs. recall
• For numeric predictors: average (relative) distance between real and predicted value
• For ranking predictors: topK etc
Evaluation Data?

現金のみ

cash only
Learning and Evaluating in Production

• Beyond static data sets, **build telemetry**
• Design challenge: identify mistakes in practice

• Use sample of live data for evaluation
• Retrain models with sampled live data regularly
• Monitor performance and intervene
ML COMPONENT TRADEOFFS
Qualities of ML Components

- Accuracy
- Capabilities (e.g. classification, recommendation, clustering...)
- Amount of training data needed
- Inference latency
- Learning latency; incremental learning?
- Model size
- Explainable? Robust?
- ...
Understanding Capabilities and Tradeoffs

• Deep Neural Networks

• Decision Trees
Where should the model live?

- Glasses
- Phone
- Cloud
- OCR Component
- Translation Component
Where should the model live?

Car

Phone

Cloud

Surge Prediction
Considerations

• How much data is needed as input for the model?
• How much output data is produced by the model?
• How fast/energy consuming is model execution?
• What latency is needed for the application?
• How big is the model? How often does it need to be updated?
• Cost of operating the model? (distribution + execution)
• Opportunities for telemetry?
• What happens if users are offline?
Typical Designs

• Static intelligence in the product
  o difficult to update
  o good execution latency
  o cheap operation
  o offline operation
  o no telemetry to evaluate and improve

• Client-side intelligence
  o updates costly/slow, out of sync problems
  o complexity in clients
  o offline operation, low execution latency
Typical Designs

• Server-centric intelligence
  o latency in model execution (remote calls)
  o easy to update and experiment
  o operation cost
  o no offline operation

• Back-end cached intelligence
  o precomputed common results
  o fast execution, partial offline
  o saves bandwidth, complicated updates

• Hybrid models
Other Considerations

• Coupling of ML pipeline parts
• Coupling with other parts of the system
• Ability for different developers and analysts to collaborate
• Support online experiments
• Ability to monitor
Reactive System Design Goals

• Responsive
  o consistent, high performance

• Resilient
  o maintain responsive in the face of failure, recovery, rollback

• Elastic
  o scale with varying loads
Common Design Strategies

• Message-driven, lazy computation, functional programming
  o asynchronous, message passing style

• Replication, containment, supervision
  o replicate and coordinate isolated components, e.g. with containers

• Data streams, “infinite data”, immutable facts
  o streaming technologies, data lakes

• See “big data systems” and “cloud computing”
UPDATING MODELS
Updating Models

• Models are rarely static outside the lab
• Data drift, feedback loops, new features, new requirements
• When and how to update models?
• How to version? How to avoid mistakes?
• Latency and automation vary widely
• Heavily distributed
Update Strategy?
PLANNING FOR MISTAKES
Mistakes will happen

• No specification
• ML components detect patterns from data (real and spurious)
• Predictions are often accurate, but mistakes always possible
• Mistakes are not predicable or explainable or similar to human mistakes
• Plan for mistakes
• Telemetry to learn about mistakes?
How Models can Break

• System outage
• Model outage
  o model tested? deployment and updates reliable? file corrupt?
• Model errors
• Model degradation
  o data drift, feedback loops
Hazard Analysis

• Worst thing that can happen?
• Backup strategy? Undoable? Nontechnical compensation?
Mitigating Mistakes

• Investigating in ML
  o e.g., more training data, better data, better features, better engineers
• Less forceful experience
  o e.g., prompt rather than automate decisions, turn off
• Adjust learning parameters
  o e.g., more frequent updates, manual adjustments
• Guardrails
  o e.g., heuristics and constraints on outputs
• Override errors
  o e.g., hardcode specific results
Mistakes?

現金のみ

cash only
Mistakes?
Telemetry

• **Purpose:**
  o monitor operation
  o monitor success (accuracy)
  o improve models over time (e.g., detect new features)

• **Challenges:**
  o too much data – sample, summarization, adjustable
  o hard to measure – intended outcome not observable? proxies?
  o rare events – important but hard to capture
  o cost – significant investment must show benefit
  o privacy – abstracting data
Summary

• Machine learning in production systems is challenging
• Many tradeoffs in selecting ML components and in integrating them in larger system
• Plan for updates
• Manage mistakes, plan for telemetry