Real-time Machine Learning

Chip Huyen (@chipro)
Founder | Real-time ML Infra Startup
UCSD - Nov '21
Becoming An MLE: Expectation

Study CS ----> Take ML courses ----> Get an MLE job
Becoming An MLE: My Journey

- Study Math
  - Travel for 3 years (didn't know what to do with my life)
  - Work as a PM (didn't want to do math)
- Study CS
  - Publish books
- Get rejected for MLE jobs (no engineering exp)
  - Blog/teach/develop OS projects
- Get an MLE job
Agenda

1. Understanding ML Production
2. Online prediction
3. Continual learning
Understanding ML Production
Production is a spectrum

Generating plots from notebooks

Deploying 1 billion requests/day with ms latency
ML in production: expectation

1. Collect data
2. Train model
3. Deploy model
ML in production: reality

1. Choose a metric to optimize
2. Collect data
3. Train model
4. Realize many labels are wrong -> relabel data
5. Train model
6. Model performs poorly on one class -> collect more data for that class
7. Train model
8. Model performs poorly on most recent data -> collect more recent data
9. Train model
10. Deploy model
11. Dream about $$$
12. Wake up at 2am to complaints that model biases against one group -> revert to older version
13. Get more data, train more, do more testing
14. Deploy model
15. Pray
16. Model performs well but revenue decreasing
17. Cry
18. Choose a different metric
19. Start over

Step 15 and 17 are essential
## ML System Failures

<table>
<thead>
<tr>
<th>ML-specific Failure Type</th>
<th>Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Poor data-model fit</td>
<td>- Poor hyperparams</td>
</tr>
<tr>
<td>- Poor hyperparams</td>
<td>- Data problems</td>
</tr>
<tr>
<td>- Data problems</td>
<td>- Bad features</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase</th>
<th>Failure Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Software</td>
</tr>
<tr>
<td></td>
<td>Development</td>
</tr>
<tr>
<td></td>
<td>Deployment</td>
</tr>
</tbody>
</table>
# ML System Failures

<table>
<thead>
<tr>
<th>Development</th>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ML-specific</strong></td>
<td></td>
</tr>
<tr>
<td>Poor data-model fit</td>
<td></td>
</tr>
<tr>
<td>Poor hyperparams</td>
<td></td>
</tr>
<tr>
<td>Data problems</td>
<td></td>
</tr>
<tr>
<td>Bad features</td>
<td></td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td></td>
</tr>
<tr>
<td>Poor implementation</td>
<td></td>
</tr>
</tbody>
</table>

## Phase

**Deployment**

- Poor data-model fit
- Poor hyperparams
- Data problems
- Bad features

- Poor implementation

**Development**

- Poor implementation
## ML System Failures

<table>
<thead>
<tr>
<th>Failure Type</th>
<th>Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ML-specific</strong></td>
<td><strong>Development</strong></td>
</tr>
<tr>
<td>Poor data-model fit</td>
<td>Poor hyperparams</td>
</tr>
<tr>
<td>Poor hyperparams</td>
<td>Data problems</td>
</tr>
<tr>
<td>Data problems</td>
<td>Bad features</td>
</tr>
<tr>
<td>Bad features</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software</th>
<th><strong>Deployment</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Poor implementation</td>
<td>Degenerate feedback loops</td>
</tr>
</tbody>
</table>
ML System Failures: data distribution shifts

- Covariate shift
- Concept drift
- Label shift
- Feature change
- Label schema change
Stakeholder objectives

**ML team**
- highest accuracy

**Product**
- fastest inference

**Sales**
- sells more ads

**Manager**
- maximizes profit
  = laying off ML teams
Stakeholder objectives

ML team
- highest accuracy

Product
- fastest inference

Sales
- sells more ads

Manager
- maximizes profit
  = laying off ML teams

Decoupling these 2 teams
# ML System Failures

## ML-specific
- Poor data-model fit
- Poor hyperparams
- Data problems
- Bad features

## Software
- Poor implementation

## Development
- Train-serving skew
- Data distribution shifts
- Edge cases
- Degenerate feedback loops
- Misalignment of ML/business

## Deployment
- Dependency failures
- Deployment failures: wrong binaries, read/write permissions
- Hardware: CPU overheat, OOM
- Network: crash, downtime
- Distributed system

## Failure Type

### Phase

- Development
- Deployment

---

15
ML System Failures

Google: 60 / 96 failures are non-ML
Myth #1: Deploying is hard
Myth #1: Deploying is hard

Deploying is easy. Deploying reliably is hard
Myth #2: You only deploy one or two ML models at a time
Myth #2: You only deploy one or two ML models at a time

Booking.com: 150+ models, Uber: thousands
Myth #3: You won’t need to update your models as much
DevOps: Pace of software delivery is accelerating

- Elite performers deploy $973x$ more frequently with $6570x$ faster lead time to deploy. ([Google DevOps Report, 2021](#))
- DevOps standard (2015)
  - Etsy deployed 50 times/day
  - Netflix 1000s times/day
  - AWS every 11.7 seconds
DevOps to MLOps: Slow vs. Fast

Only 11% of organizations can put a model into production within a week, and 64% take a month or longer.

Machine learning Platform in Weibo (WML) —— CTR model iteration

After successive iterations, Weibo machine learning platform (WML), can support over 1000B parameters, 1m QPS, and iteration cycle around 10 minutes now.

Left image from Algorithmia | Right image: Machine learning with Flink in Weibo (Qian Yu, QCon 2019)
Accelerating ML Delivery

How often SHOULD I update my models?

How often CAN I update my models?
ML + DevOps =
Myth #4: ML can magically your business overnight
Myth #4: ML can magically your business overnight

Magically: possible
Overnight: no
Efficiency improves with maturity

Model deployment timeline and ML maturity

- Getting developed models into production
- Early adopters (models in production 1-2 years)
- Mid-stage adopters (models in production 2-4 years)
- Sophisticated (models in production 5+ years)

0-7 days  8-30 days  31-90 days  91-365 days  1+ years
ML engineering is more engineering than ML

MLEs might spend most of their time:
● wrangling data
● understanding data
● setting up infrastructure
● deploying models
instead of training ML models
Two levels of real-time ML

- Online prediction
- Online continual learning
Online prediction
You’ve developed a model, now what?
Online prediction

Predictions are generated and returned as soon as* requests arrive
Latency matters

- 100ms delay can hurt conversion rates by 7% (Akamai study, 2017)
- 30% increase in latency costs 0.5% conversion rate (Booking.com, 2019)
- 53% will leave a page that takes >3s to load (Google, 2016)
ML evolution

Inference latency

Model size

Time

Bigger, better, slower
Batch prediction

- Generate predictions offline
- Store them somewhere (e.g. SQL tables)
- Pull out pre-computed predictions when needed
Batch predictions work great in many cases

Everything is a Recommendation

Recommendations are driven by machine learning algorithms

Over 80% of what members watch comes from our recommendations

Balancing Discovery and Continuation in Recommendations (Hossein Taghavi, 2017)
Batch predictions: problems

- Needs to know the queries in advance
- Can’t adapt to changing interests
Batch predictions: problems

- Needs to know the queries in advance
- Can’t adapt to changing interests
Online prediction: solution

- Fast inference
  - model that can make predictions with latency acceptable to users

- Real-time pipeline
  - a pipeline that can process data, input it into model, and return a prediction in real-time
Real-time pipeline: ride-sharing example

To detect whether a transaction is fraud, need features from:

- this transaction
- user’s recent transactions (e.g. 7 days)
- credit card’s recent transactions
- recent in-app frauds
- etc.
Real-time pipeline: ride-sharing example

To detect whether a transaction is fraud, need features from:

- this transaction
- user’s recent transactions (e.g. 7 days)
- credit card’s recent transactions
- recent in-app frauds
- etc.

How to quickly access these features?
Real-time Transport

Permanent storage (S3)

Events > 7 days

Discard

10:02 - Picks location
10:03 - Books trip
10:04 - Adds credit card

10:06 - Cancels trip
10:04 - Contacts driver

Incoming events

In-memory storage
Real-time Transport

972 companies reportedly use Kafka in their tech stacks, including Uber, Spotify, and Shopify.

233 companies reportedly use Amazon Kinesis in their tech stacks, including Amazon, Instacart, and Lyft.
Online prediction: static & dynamic features

<table>
<thead>
<tr>
<th>Static data</th>
<th>Streaming data</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV, PARQUET, etc.</td>
<td>Kafka, Kinesis, Pulsar, etc.</td>
</tr>
<tr>
<td>Bounded: know when a job finishes</td>
<td>Unbounded: never finish</td>
</tr>
<tr>
<td>Static features:</td>
<td>Dynamic features</td>
</tr>
<tr>
<td>● age, gender, job, city, income</td>
<td>● locations in the last 10 minutes</td>
</tr>
<tr>
<td>● when account was created</td>
<td>● recent activities</td>
</tr>
<tr>
<td>Can be processed in batch</td>
<td>Processed as events arrive</td>
</tr>
<tr>
<td>● e.g. MapReduce, Spark</td>
<td>● e.g. Apache Flink, Samza, Spark Streaming</td>
</tr>
</tbody>
</table>
One model, two pipelines

**Inference**
- Streaming data
- Stream processing
- Features

**Training**
- Static data
- Batch processing
- Features
One model, two pipelines

⚠⚠ A common source of errors in production ⚠⚠
One model, two pipelines: example

Machine learning with Flink in Weibo (Qian Yu, QCon 2019)
Stream & batch processing

- Batch is a special case of streaming
Microservices ft. REST APIs
REST APIs: request-driven

Server has to listen for the request to register
Inter-service communication

Rider management

Need driver availability & price to show riders

Driver management

Need ride demand & price to incentivize drivers

Price optimization

Need ride demand & driver availability to set price
Request-driven: problems

- Complex inter-service communication
- How to map data transformations through the entire system?
Event-driven

- All services publish to a stream
- All services subscribe to this stream to get info they need
Event-driven

- Data flows through this stream, can monitor data transformations through the entire system
Continual Learning
Model’s performance degrades in production

- Data distribution shifts
  - Sudden
  - Cyclic
  - Gradual
From Monitoring to Continual Learning

- Monitoring: detect changing data distributions
- Continual learning: continually adapt models to changing data distributions
Continual learning

- Very few companies actually update models with each incoming sample
  - Catastrophic forgetting
  - Can get unnecessarily expensive*
- Update models with micro-batches
Learning schedule != evaluating schedule

- Evaluated after a certain period of time
  - Offline evaluation (sanity check)
  - Online evaluation: canary analysis, A/B testing, bandits
Iteration cycle: minutes

- Alibaba: Singles Day sale
- Weibo
- Tiktok
- SheIN
Iteration cycle: US

Only 11% of organizations can put a model into production within a week, and 64% take a month or longer.

- 1 day or less: 1%
- 1 day-1 week: 10%
- 1 week-1 month: 25%
- 1 month-1 quarter: 24%
- 1-2 quarters: 18%
- 2-3 quarters: 12%
- 3 quarters-1 year: 6%
- More than 1 year: 3%
How often to retrain your model is just a knob to turn
Continual learning: use cases

- Rare events
  - Christmas/Black Friday/Prime Day shopping
  - Total Landscaping

- Continuous cold start (in-session adaptation)
  - New users
  - New devices
  - Users not logged in
  - Users rarely logged in
Continual learning is especially good for

- Natural labels: e.g. user click -> good prediction
- Short feedback loops
- Examples:
  - RecSys
  - Ranking
  - Ads CTR prediction
  - eDiscovery
Quantify the value of data freshness

1. How much model’s performance changes if switch from retraining monthly to weekly to daily to hourly?
   a. FB: CTR loss can be reduced ~1% going from training weekly to daily

Practical Lessons from Predicting Clicks on Ads at Facebook (2014)
Quantify the value of data freshness

1. How much model’s performance changes if switch from retraining monthly to weekly to daily to hourly?
2. How would retention change if you can do in-session adaptation?
3. Model iteration vs. data iteration?
Quantify the value of fast iteration

1. How many more experiments can you run if model changes can be deployed automatically ASAP?
Quantify cloud bill savings

Going from monthly training to daily training gives 45x cost savings and +20% metrics increase

Online Learning for Recommendations at Grubhub (Alex Egg, 2021)
Barriers to streaming-first infrastructure

1. Companies don’t see the benefits of streaming
   a. Systems not at scale
   b. Batch prediction works fine*
   c. Online prediction would work better but never done that before so don’t know
Barriers to streaming-first infrastructure

1. Companies don’t see the benefits of streaming
2. High initial investment on infrastructure

Snowflake Streaming: Now Hiring! Help design and build the future of big data and stream processing

Confluent IPO: Remaking The Massive Database Industry

Streaming database platform provider Materialize lands $60M
Bet on the Future

1. 1000s incremental improvements on batch system vs. one big jump to streaming
2. New technologies: best chance for a big metric win 🚀🚀🚀
Thank you!
Talk to us about how we can help 👋

@chipro
chip@huyenchip.com
General tips

1. Job search and interview preparation are lifelong processes.
2. The best time to interview is when you don’t need a job.
3. Start looking for jobs 3-6 months before.
4. Build up your portfolio and publish them.
5. Get people to refer you.
6. Look up your interviewers. Review their work.
7. Have your friends to give you mock interviews.
8. Don’t pretend that you know something when you don’t.
9. Don’t criticize your previous or current employers.
10. Don’t talk about your age, marital status, religion, political affiliation.
11. Have competing offers.
12. Don’t sweat it. If you tank an interview, move on.
ML in production: reality

1. Choose a metric to optimize
2. Collect data
3. Train model
4. Realize many labels are wrong -> relabel data
5. Train model
6. Model performs poorly on one class -> collect more data for that class
7. Train model
8. Model performs poorly on most recent data -> collect more recent data
9. Train model
10. Deploy model
11. Dream about $$$
12. Wake up at 2am to complaints that model biases against one group
13. Get more data, train more, do more testing
14. Deploy model
15. Pray
16. Model performs well but revenue decreasing
17. Cry
18. Choose a different metric
19. Start over