Continuous Live Monitoring of Machine Learning Models with Delayed Label Feedback

Patrick Baier
Lorand Dali

Zalando Payments

June 2018
OUTLINE

Who we are and what we do
Why we should monitor
Prediction Monitoring
Our implementation
WHO WE ARE AND WHAT WE DO
WHO WE ARE

Patrick Baier
- Data Scientist at Zalando (~ 3.5 years)
- PhD in Computer Science from Uni Stuttgart

Lorand Dali
- Data Scientist at Zalando (~ 1.5 years)
- Diploma in Computer Science from the Technical University of Cluj Napoca
Detect and prevent payment fraud
WHAT WE DO

Detect and prevent payment fraud
MODEL TRAINING

- Amazon S3
- Apache Spark
- ML Model (LR, RF, GBT, ...)

Use bullet points to summarize information rather than writing long paragraphs in the text box.
- REST service
- Scala Play service with Spark bindings
- Response time: <1 second
OUR TECH STACK

- Scala
- Amazon Web Services
- GitHub
- Spark
- Docker
- EXASOL
- R
- Python
WHY WE SHOULD MONITOR
Let’s deploy a model for fraud detection in an online shop!

Steps we take:

1. Collect training data.
2. Train a model.
3. Deploy it to production.
COLLECT DATA

Go through the systems and collect data for training
<table>
<thead>
<tr>
<th>#</th>
<th>Feature-1</th>
<th>Time-to-order [s]</th>
<th>...</th>
<th>Feature-N</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>300</td>
<td></td>
<td>1</td>
<td>not-fraud</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td></td>
<td>0</td>
<td>fraud</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>120</td>
<td></td>
<td>0</td>
<td>not-fraud</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>200</td>
<td></td>
<td>1</td>
<td>not-fraud</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>250</td>
<td></td>
<td>0</td>
<td>fraud</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
FEATURE DISTRIBUTION

Distribution of feature in training data
Once we are live, we get features $x$ sent over by a different microservice in real-time.
MONITORING

ML Model

- X
- $p_{\text{fraud}}$
- CPU usage
- Memory usage
- Latency
- ...

Please write the title in all capital letters.
Some weeks later, people are angry:
“We fail to detect fraud, our business is ruined!”

What happened?
INVESTIGATION

ML Model

CPU usage
Memory usage
Latency

$\mathbf{X} \rightarrow \mathbf{p}_{\text{fraud}}$
INVESTIGATION

**ML Model**

- Time-to-order
  - 300000
  - 5000
  - 120000
  - ...

- $p_{\text{fraud}}$
INVESTIGATION

ML Model

- Time-to-order:
  - 300000
  - 5000
  - 120000
  - ...

\( p_{\text{fraud}} \)

Mean shifted from 200 to 200000!
Please write the title in all capital letters. Use bullet points to summarize information rather than writing long paragraphs in the text box.

**INVESTIGATION**

- **ML Model**

  → All our predictions are corrupt!

<table>
<thead>
<tr>
<th>Time-to-order</th>
</tr>
</thead>
<tbody>
<tr>
<td>300000</td>
</tr>
<tr>
<td>5000</td>
</tr>
<tr>
<td>120000</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

The feature is sent to us in the unit of milliseconds (not in seconds)!
PROBLEMS

1. We lost a lot of money.
2. We did not detect it in time.
3. We could have detected it in time and provided a fix.
CONCLUSIONS

We need to make sure that the distributions of input features are (always) the same as in training.
PREDICTION
MONITORING
FAILING FEATURES

Monitor failing input features:

<table>
<thead>
<tr>
<th>feature name</th>
<th>fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>feature one</td>
<td>0.903</td>
</tr>
<tr>
<td>feature two</td>
<td>0.004</td>
</tr>
<tr>
<td>feature three</td>
<td>0.004</td>
</tr>
<tr>
<td>feature four</td>
<td>0.004</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
LIVE MONITORING

Compare feature distributions and output probability:

- Feature distribution on test data
- Feature distribution on live data

Quality Monitor
LIVE MONITORING
LIVE MONITORING

Compare distributions with KS-distance:

<table>
<thead>
<tr>
<th>feature name</th>
<th>this vs previous</th>
<th>this vs test</th>
<th>previous vs test</th>
</tr>
</thead>
<tbody>
<tr>
<td>feature one</td>
<td>0.0000008</td>
<td>0.928806</td>
<td>0.928798</td>
</tr>
<tr>
<td>feature two</td>
<td>0.0009117</td>
<td>0.019504</td>
<td>0.020416</td>
</tr>
<tr>
<td>feature three</td>
<td>0.1075305</td>
<td>0.316970</td>
<td>0.313337</td>
</tr>
<tr>
<td>feature four</td>
<td>0.943896</td>
<td>0.943655</td>
<td>0.045654</td>
</tr>
<tr>
<td>prediction</td>
<td>6.606939e-02</td>
<td>0.255182</td>
<td>0.277325</td>
</tr>
</tbody>
</table>
How big should the window size for data aggregation be?

- **$t = 1h$**
  - Fast detection of anomalies
  - Suffers from short term seasonalities

- **$t = 12h$**
  - Deals with short term seasonalities
  - Slow detection of anomalies
EXECUTION SCHEDULE

How often should we analyze?

- every hour?
- $t = 12h$
- $t = 12h$
- $t = 12h$

go live
EXECUTION SCHEDULE

How often should we analyze?

More often:
- Detect anomalies more quickly
- High complexity, higher costs

Less often:
- Less complex, lower costs
- Delay of anomaly detection

every hour?
LIVE MONITORING

possible discoveries

- technical problems,
- seasonalities,
- change of behaviour,
- fraud wave,
- fraud patterns,
- deviation from expectations.
IMPLEMENTATION
DISTANCE BETWEEN TWO DISTRIBUTIONS

\[ d = \frac{\int |f_1(x) - f_2(x)| \, dx}{\int \max(f_1(x), f_2(x)) \, dx} \]

Sum and normalize to $[0, 1]$
WE USE THE CDF

CDF

PERCENTILES

PDF

HISTOGRAM
import com.tdunning.math.stats.TDigest
import org.apache.spark.rdd.RDD

def create(numbers: Seq[Double]): TDigest = {
  val digest: TDigest = TDigest.createDigest(100)
  numbers.foreach(x => digest.add(x))
  digest
}

def create(numbers: RDD[Double]): TDigest = {
  val empty: TDigest = TDigest.createDigest(100)

  numbers.treeAggregate(empty)(
    seqOp = (acc: TDigest, x: Double) => {
      acc.add(x)
      acc
    },
    combOp = (digest1: TDigest, digest2: TDigest) => {
      digest1.add(digest2)
      digest1
    }
  )
}
PREDICTION SERVING
PREDICTION SERVING

- Request
- \( p_{\text{fraud}} \)
- REST
- Play framework
- Prediction engine
- ML model
- SQS
- S3
- (Lumberjack) Process
func process(sqsClient sqsiface.SQSAPI, dumpSize int,
    interrupt <- chan bool, upload func([]*sqs.Message)) {

    buffer := make([]*sqs.Message, 0, dumpSize)
    timer := time.NewTimer(maxFetchingTime)

    for {
        select {
            case <-interrupt:
                return
            case <-timer.C:
                upload(buffer)
            default:
                for _, message := range receiveMessages(sqsClient) {
                    buffer = append(buffer, message)
                }
                if len(buffer) == dumpSize {
                    upload(buffer)
                }
        }
    }
}
PUTTING IT TOGETHER IN AWS DATA PIPELINE

- process logs
- group by models
- get failed features
- create tdigests
- create histograms
- feature distances

Reports

Alerts
FINAL NOTES

• if you have a ML system deployed in production, then you have to monitor it somehow
• monitoring is especially important if performance feedback is delayed
• start simple and non-intrusive, keep the reports flexible
• automate as much as possible
• to measure how far you are with monitoring, go through the questions in this paper from Google: "What’s your ML Test Score? A rubric for ML production systems"
THANK YOU!

Patrick Baier & Lorand Dali

https://tech.zalando.com/blog/scalable-fraud-detection-fashion-platform