Difficulties of data preparation for production ML.

Dmitry Zhukov
Tartu, 2018
I am

- Optics engineer
- Work at TransferWise
- Was doing fraud detection
Content

- Technical Debt
- Categorical Features
What is a technical debt?
- The term is coined by Ward Cunningham in 1992.
- Like incurring fiscal debt, there are often sound strategic reasons to take on technical debt.
- Machine Learning has a high interest rate.
Data processing complexity.
ML problems classification.

- “Spatial” data-processing complexity
- “Temporal” data-processing complexity
- Data processing latency
Complexity of the data processing pipeline that is induced by the number of unique upstream dependencies.

“Spatial” data-processing complexity.
“Spatial” data-processing complexity.

- Number of sensors in the car
- Number of micro-services in the company
“Temporal” data-processing complexity.

Complexity of the data processing pipeline that is induced by the number of observations that has to be pre-aggregated before being passed to the ML algorithm.
“Temporal” data-processing complexity.

- Single observation
  - Detecting a cat in a single image
  - If document is fake
- Thousands and millions of historic observations
  - Trading
  - Fraud detection
Data processing latency.

The latency that has to be provided when doing prediction in production setup.
Data processing latency.

- Predict once and forget
  - DNA Research
  - Growth forecasts
- Sub-second latency
  - Trading
  - Fraud detection
  - Self-driving cars
  - Face detection
There are many services in TransferWise that the data has to be collected from - spatial data-processing complexity.
Each user has long history of transfers, moreover banks that users transfer money from have even longer history - temporal data-processing complexity.
Fraud/Not fraud decision has to be made in seconds or even less - prediction latency.
But what this has to do with the machine learning?
ONE DOES NOT SIMPLY EMBRACE

THE COMPLEXITY OF THE MANIFOLD
The difference is in the scale.
No good tools to prepare the data $\rightarrow$ Technical difficulties that lead to debt
Technical debt in production ML
Types of technical debt

- Entanglement
- Different code paths in production and training
- Glue code
Entanglement (coupling)

Derives from:

- Spatial data-processing complexity
Loose coupling

Traditional code
Tight coupling
Even worse :(  

[Diagram showing a network with 'ML' at the center]
Modularisation struggles

Current

Desired
Different code paths in production and training

Derives from:
- Temporal data-processing complexity
- Prediction latency
There is no single tool that fulfills all the requirements
Services

Production

Collect Data
Process Data
Extract Features
Predict

Training / Testing

Collect Data
Process Data
Extract Features
Train
Test
Services

Collect Data

Process Data

Extract Features

Predict

Log

Extract Features

Train

Test

Production

Training / Testing

http://

Java Camel

Apache

Camel

php

Spark
Glue code

Derives from:
- ML ecosystem fragmentation
Schema management turned to be a nightmare
Pipeline Jungle

Derives from:

• Spatial data-processing complexity
How to pay back?
Log all inputs and outputs

- Reuse data processing from production
- Replay
- Traceability
Event sourcing

- Easier to add a new feature
- Replayable data-processing
- Fixing bugs
One code for production and training setups

- Guarantees no discrepancies
- No need to test in production
Treat all data as a stream

- No duplicate effort
- Avoid data leaks
- Use same stream processing framework in production and training
Integrated streaming and machine learning

- No glue code
- Ease of development
Apache Spark?
No... the problem waits for it’s heroes!
Categorical Features
Features

- Continuous
- Categorical
Categorical features

Cardinality

- Low
- High
- Ultra-high
Can be handled by one-hot encoding or decision trees

Unseen values almost do not appear

Examples:
- Browser: Chrome, Firefox, IE
- Payment method: bank transfer, credit card

Low cardinality
High cardinality

- More than Low but still orders of magnitude less than the number of observations
- Appear quite often but retraining the model is enough to catch up
- Examples:
  - Country code: EE, GB, DE
  - Card BIN (first 6 digits): 652456, 451235
Ultra-high cardinality

- The same order as the number of observations
- Unseen values appear almost with every new observation
- Examples:
  - IP: 241.52.63.77, 5.25.8.3
  - Card number: 6524 5624 5252 7774, 4512 3568 7825 8372
Different cardinalities require different approaches

<table>
<thead>
<tr>
<th>Method</th>
<th>Low</th>
<th>High</th>
<th>Ultra-high</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-hot encoding</td>
<td>Good</td>
<td>Bad</td>
<td>Useless</td>
</tr>
<tr>
<td>Target encoding</td>
<td>Bad</td>
<td>Good</td>
<td>Ok</td>
</tr>
<tr>
<td>Model specific</td>
<td>Good</td>
<td>Ok</td>
<td>-</td>
</tr>
<tr>
<td>Graphical models</td>
<td>Bad</td>
<td>Ok</td>
<td>Good</td>
</tr>
</tbody>
</table>
One-hot encoding

- Easy
- The best for low cardinality
- Gets very bad as cardinality grows
- No data loss
Target encoding
aka label averaging, ratio feature

- Tricky to implement right
- Suffers from data leaks
- Difficult to scale - the source for temporal data-processing complexity
- Loses data
Almost always better than one-hot
- Controllable data loss
- Doesn’t get to ultra-high anyway
- Examples: decision trees, neural net embeddings

Model specific
Graphical models

- The only ultimate answer to ultra-high cardinality
- Super hard to implement
Thank you!

Dmitry Zhukov
dzhukov@transferwise.com