A Relational Framework for Classifier Engineering

Benny Kimelfeld
Technion, Israel

Christopher Ré
Stanford University
Background

• ML application constantly increasing
  – e.g., by 2020 >50% Intel servers will run ML (D. Bryant, Intel SVP)

• Rising interest in DB research for ML
  – e.g., query optimization for feature selection / evaluation [Zhang+14, Kumar+15,16], ML on factorized DB [Schleich+16]
  – DEEM workshop on *Data Management for End-to-End ML*
  – Dagstuhl Presp. Workshop 16151: *Research Directions for PDM*

• Feature Engineering (FE) critical for quality
  – Yet heavy resource consumer in ML development
  – Tooling and principles [Guyon+06 book]
  – Standard practice; here to stay!
    ▪ *Deep Learning* avoids FE; applicable in certain areas / domains w/ massive training data available
Classic ML Classification Flow

Raw data flows into a database (DB). The data undergoes feature engineering, which is largely programming; the “artistic” part. Features are then classified by a classifier (model) into two classes: fraudulent or honest. Common classes typically reused include linear, log, kernel, decision tree, etc. These classes are learned or tuned from examples.
Classic ML Classification Flow

Our scope

Our focus

Traditional ML theory

Raw data

DB

Q₁ → +1
Q₂ → +1
Q₃ → -1
Q₄ → +1
Q₅ → -1

Classifier (model)

fraudulent

honest

Features

Largely programming; the “artistic” part

Feature engineering

Common classes typically reused: linear, log, kernel, decision tree, etc.

Learned (tuned/adjusted) from examples

Feature engineering

Largely programming; the “artistic” part

Common classes typically reused: linear, log, kernel, decision tree, etc.

Learned (tuned/adjusted) from examples
Framework Goal

• DB “understands” how entities become features
  – Relational structure, constraints, queries

• Can be used for assisting FE?
  – Estimate feature quality?
  – Suggest new features?
  – Test for suitability of a feature language?
  – Detect engineering faults?
  – Implication of underlying languages on computational complexity?
  – Benefit from decades of DB theory?

• Setup for attacking questions

• Step towards DB theory for ML engineering
Outline

- Formal Setup
- Computational Problems
- Complexity Results
- Directions
• ML task: **binary classification**
  – Learn a mapping \( \text{entity} \rightarrow +1/-1 \)

• Boolean features
  – Simplifies the framework
  – Common in practice
    ▪ e.g., *binning / bucketing*

• Hence, a **classifier** has the form

\[
C : \{+1,-1\}^n \rightarrow \{+1,-1\}
\]
<table>
<thead>
<tr>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TXN</th>
<th>card</th>
<th>country</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>US</td>
<td>GA</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>US</td>
<td>NY</td>
</tr>
<tr>
<td>3</td>
<td>101</td>
<td>BR</td>
<td>RJ</td>
</tr>
<tr>
<td>4</td>
<td>102</td>
<td>US</td>
<td>CA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>ssn</th>
<th>country</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>200</td>
<td>US</td>
<td>GA</td>
</tr>
<tr>
<td>101</td>
<td>201</td>
<td>US</td>
<td>NY</td>
</tr>
<tr>
<td>102</td>
<td>202</td>
<td>BR</td>
<td>RJ</td>
</tr>
</tbody>
</table>
(txn in owner's state) $Q_1(x) \leftarrow \text{TxnInfo}(x, n, c, s), \text{Card}(n, c, s)$
(txn in owner's country) $Q_2(x) \leftarrow \text{TxnInfo}(x, n, c, s), \text{Card}(n, d, s)$
(txn in NY) $Q_3(x) \leftarrow \text{TxnInfo}(x, n, c, 'NY')$
Formal Setup

• Entity schema: \((S, \eta)\)
  – \(S\) is a relational schema (signature, constraints)
  – \(\eta\) is a unary relation in \(S\), representing entities

• An instance \(I\) of \(S\) defines:
  – An entity set \(\eta^I\) (the \(\eta\) relation of \(I\))
  – Information on the entities (all other relations)

• Feature query: unary query \(Q\) over \(S\)

• Statistic: series \(\Pi = (Q_1, ..., Q_n)\) of feature queries

• Each \(e \in \eta^I\) has a feature vector \(\Pi(e) = (f_1, ..., f_n)\)

\[ f_i = \begin{cases} 
+1 & \text{if } e \in Q_i(I) \\
-1 & \text{if } e \notin Q_i(I) 
\end{cases} \]
\[ \mathbf{S} \]

<table>
<thead>
<tr>
<th>Txn</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TxnInfo</th>
<th>Card</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXN</td>
<td>card</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>101</td>
</tr>
<tr>
<td>4</td>
<td>102</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>ssn</th>
<th>country</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>200</td>
<td>US</td>
<td>GA</td>
</tr>
<tr>
<td>101</td>
<td>201</td>
<td>US</td>
<td>NY</td>
</tr>
<tr>
<td>102</td>
<td>202</td>
<td>BR</td>
<td>RJ</td>
</tr>
</tbody>
</table>

Feature queries:
- \( Q_1(x) \leftarrow \text{TxnInfo}(x, n, c, s), \text{Card}(n, c, s) \)
- \( Q_2(x) \leftarrow \text{TxnInfo}(x, n, c, s), \text{Card}(n, d, s) \)
- \( Q_3(x) \leftarrow \text{TxnInfo}(x, n, c, \text{'NY'}) \)

Statistic: \( \mathbf{\Pi} = (Q_1, Q_2, Q_3) \)
Training

- Let \((S, \eta)\) be an entity schema.
- A \textit{training instance} is a pair \((I, \lambda)\) where
  - \(I\) is an instance over \(S\)
  - \(\lambda: \eta^I \rightarrow \{+1, -1\}\) is a labeling function.
- \((I, \lambda) + \text{statistic } \Pi\) define the training collection
  \[
  T = \{ \langle \Pi(e), \lambda(e) \rangle \mid e \in \eta^I \}\]
- Training finds a classifier from a \textit{hypothesis class} \(H\) by minimizing a \textit{risk function} over \(T\).
<table>
<thead>
<tr>
<th>id</th>
<th>λ(e)</th>
<th>id</th>
<th>ssn</th>
<th>country</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>100</td>
<td>200</td>
<td>US</td>
<td>GA</td>
</tr>
<tr>
<td>2</td>
<td>+1</td>
<td>101</td>
<td>201</td>
<td>US</td>
<td>NY</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>102</td>
<td>202</td>
<td>BR</td>
<td>RJ</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>102</td>
<td>202</td>
<td>BR</td>
<td>RJ</td>
</tr>
</tbody>
</table>

Q₁(x) ← TxnInfo(x, n, c, s), Card(n, c, s)
Q₂(x) ← TxnInfo(x, n, c, s), Card(n, d, s)
Q₃(x) ← TxnInfo(x, n, c, 'NY')

Π = (Q₁, Q₂, Q₃)
QL ∈ \{ Q_1(x) \leftarrow \text{TxnInfo}(x, n, c, s), \text{Card}(n, c, s), \quad \text{Q}_2(x) \leftarrow \text{TxnInfo}(x, n, c, s), \text{Card}(n, d, s), \quad \text{Q}_3(x) \leftarrow \text{TxnInfo}(x, n, c, 'NY') \}
\quad \Pi = (Q_1, Q_2, Q_3) 
\epsilon H
Outline

- Formal Setup
- Computational Problems
- Complexity Results
- Directions
Can we fit feature queries & classifier of chosen families?
Problem 1: Separability

The naïve “noise-free” training from ML textbooks:

*Is full separation possible?*

**(H,QL)-separability**

Given a training instance \((I,\lambda)\) over a schema \((S,\eta)\), is there any statistic \(\Pi\) in \(QL\) such that \((I,\lambda)\) can be perfectly realized by a classifier in \(H\)?
<table>
<thead>
<tr>
<th>Txn</th>
<th>id</th>
<th>λ(e)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TXN</th>
<th>card</th>
<th>country</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>US</td>
<td>GA</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>US</td>
<td>NY</td>
</tr>
<tr>
<td>3</td>
<td>101</td>
<td>BR</td>
<td>RJ</td>
</tr>
<tr>
<td>4</td>
<td>102</td>
<td>US</td>
<td>CA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>ssn</th>
<th>country</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>200</td>
<td>US</td>
<td>GA</td>
</tr>
<tr>
<td>101</td>
<td>201</td>
<td>US</td>
<td>NY</td>
</tr>
<tr>
<td>102</td>
<td>202</td>
<td>BR</td>
<td>RJ</td>
</tr>
</tbody>
</table>

Q₁(\(x\)) ← TxnInfo(\(x, n, c, s\)), Card(\(n, c, s\))
Q₂(\(x\)) ← TxnInfo(\(x, n, c, s\)), Card(\(n, d, s\))
Q₃(\(x\)) ← TxnInfo(\(x, n, c, 'NY'\))

Redundancy in features?
Redundancy / Identifiability

- Linear column dependence in the feature matrix often means redundant features
  - e.g., linear/logistic classification/regression

- ML libraries often require full column rank
  - For their optimization solution to be “identifiable”
  - c.f. “Theory of Point Estimation” \cite{LehmannCasella83}
<table>
<thead>
<tr>
<th>TXN in owner’s US state</th>
<th>TXN in different US state</th>
<th>TXN in East Coast</th>
<th>TXN US but not East Coast</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>TXN in owner’s US state</td>
<td>TXN in different US state</td>
<td>TXN in East Coast</td>
<td>TXN US but not East Coast</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

sum = sum
Problem 2: Identifiability

QL-identifiability

Given a statistic $\pi$ in QL over entity schema $(S, \eta)$, is there any instance $I$ with a column-independent feature matrix?

Two variants:

- **Linear** independence (arises in, e.g., least-square minimization)
- **Affine** independence (arises in, e.g., entropy minimization)
<table>
<thead>
<tr>
<th>id</th>
<th>λ(e)</th>
<th>TXN</th>
<th>card</th>
<th>country</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>100</td>
<td>US</td>
<td>GA</td>
</tr>
<tr>
<td>2</td>
<td>+1</td>
<td>2</td>
<td>100</td>
<td>US</td>
<td>NY</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>3</td>
<td>101</td>
<td>BR</td>
<td>RJ</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>4</td>
<td>102</td>
<td>US</td>
<td>CA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>ssn</th>
<th>country</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>200</td>
<td>US</td>
<td>GA</td>
</tr>
<tr>
<td>101</td>
<td>201</td>
<td>US</td>
<td>NY</td>
</tr>
<tr>
<td>102</td>
<td>202</td>
<td>BR</td>
<td>RJ</td>
</tr>
</tbody>
</table>

 txn in owner's state) Q₁(x) ← TxnInfo(x, n, c, s), Card(n, c, s)
 (txn in owner's country) Q₂(x) ← TxnInfo(x, n, c, s), Card(n, d, s)
 (txn in NY) Q₃(x) ← TxnInfo(x, n, c, 'NY')

How much training to learn over our features?
Vapnik-Chervonenkis (VC) Dimension

- *What is the max #entities that can be shattered*?
  - That is, perfectly classified on every possible labeling?

- Complexity measure for learnability
  - (not the only one)

- Estimate training amount to avoid overfitting
Problem 3: Dimensionality

(H, QL)-dimensionality

Given a statistic $\Pi$ in $QL$ over an entity schema $(S, \eta)$, what is the max $m$ such that some instance with $m$ entities can be shattered by $H$?
Outline

- Formal Setup
- Computational Problems
- Complexity Results
- Directions
Scope of Results

• Complexity analysis in a specific setting:
  – Hypothesis class $H = \text{Lin}$: linear classifiers
  – Query language $QL = \text{CQ}$: conjunctive queries
    ▪ Without constants
  – No schema constraints

• Mostly intractable complexity classes (expected)

• Baseline & justification for future assumptions

• Next, a few highlights
(Lin,CQ)-Separability

Given a training instance \((I,\lambda)\) over a schema \((S,\eta)\), is there any statistic \(\Pi\) in CQ such that \((I,\lambda)\) can be perfectly realized by a classifier in Lin?

- Every training instance is separable, unless entities with different labels are indistinguishable by CQs
  - That is, there are \(e\) and \(e'\) with \(\lambda(e)\neq\lambda(e')\) and endomorphism that maps \(e\) and \(e'\) and vice versa
  - Relationship to CQ-query-by-example
    - [Willard10, tenCateDalmau15, BarcelóRomero16]
  - \(\text{coNP}\)-complete

- Avoiding self joins \(\rightarrow\) harder: \(\Sigma^P_2\)-complete!
CQ-Identifiability

Given a statistic $\Pi$ in CQ over entity schema $(S, \eta)$, is there any instance $I$ with a column-independent feature matrix?

- The following are equivalent if CQs are connected:
  - $\Pi$ is linearly identifiable
  - $\Pi$ is affinely identifiable
  - $\Pi$ is non-redundant (no equivalent feature queries)

- Pairwise equivalences break if:
  - CQs can be disconnected
  - CQs can have negation

- Generalized characterization for disconnected CQs
- coNP-complete
(Lin,CQ)-Dimensionality

Given a statistic $\Pi$ in CQ over entity schema $(S, \eta)$, what is the max $m$ such that some instance with $m$ entities can be shattered by Lin?

- For connected CQs VC dim w.r.t. $\Pi$ is $d+1$
  
  $d = \#$(equivalence classes among CQs in $\Pi$)
  
  - In particular, containment among CQs does not reduce the VC dimension compared to vanilla linear classification

- Can go down if we allow:
  
  - Disconnected CQs
  - Negation
Outline

- Formal Setup
- Computational Problems
- Complexity Results
- Directions
Directions for Future Research

• Schema constraints

• Generalized features / tasks
  – Numeric, aggregate, multi-label, regression

• Realistic variants of separability
  – Approximate/noisy, incremental

• Restrict model complexity
  – Small/shallow feature queries, low statistic dimension

• Connection to prob. DBs (statistical guarantees?)

• Context of text analysis
  – Doc. spanners [Fagin+2014], DeepDive [Shin+2015]

• …
Summary

- Framework for classifier engineering over DBs
  - Entity schema, feature query, statistic, training instance
- Goal: DB smartness (schema, constraints, queries) to aid feature engineering
- Illustrated on several computational problems
  - Separability, dimensionality, identifiability
  - Preliminary results for linear classifiers and CQs
- Plethora of problems / directions to pursue

Thank you! Questions?