In-Database Factorised Learning
fdbresearch.github.io

Mahmoud Abo Khamis, Hung Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich
December 2017

Logic for Data Science Seminar
Alan Turing Institute
Current Landscape for ML over DB

Factorised Learning over Normalised Data

Learning under Functional Dependencies

General Problem Formulation
Brief Outlook at Current Landscape for ML over DB (1/2)

No integration

- ML & DB distinct tools on the technology stack
- DB exports data as one table, ML imports it in own format
- **Spark/PostgreSQL + R** supports virtually any ML task
- Most ML over DB solutions operate in this space
No integration

- ML & DB distinct tools on the technology stack
- DB exports data as one table, ML imports it in own format
- **Spark/PostgreSQL + R** supports virtually any ML task
- Most ML over DB solutions operate in this space

Loose integration

- Each ML task implemented by a distinct UDF inside DB
- Same running process for DB and ML
- DB computes one table, ML works directly on it
- **MadLib** supports comprehensive library of ML UDFs
Unified programming architecture

- One framework for many ML tasks instead of one UDF per task, with possible code reuse across UDFs
- DB computes one table, ML works directly on it
- **Bismark** supports incremental gradient descent for convex programming; **up to 100% overhead over specialized UDFs**
Unified programming architecture

• One framework for many ML tasks instead of one UDF per task, with possible code reuse across UDFs
• DB computes one table, ML works directly on it
• Bismark supports incremental gradient descent for convex programming; up to 100% overhead over specialized UDFs

Tight integration ⇒ In-Database Analytics

• One evaluation plan for both DB and ML workload; opportunity to push parts of ML tasks past DB joins
• Morpheus + Hamlet supports GLM and naïve Bayes
• Our approach supports PR/FM, decision trees, ...
In-Database Analytics

- Move the analytics, not the data
  - Avoid expensive data export/import
  - Exploit database technologies
  - Exploit the relational structure (schema, query, dependencies)
  - Build better models using larger datasets and faster

- Cast analytics code as join-aggregate queries
  - Many similar queries that massively share computation
  - Fixpoint computation needed for model convergence
In-database vs. Out-of-database Analytics

- Feature extraction query
- DB
- Materialized output
- ML tool
- $\theta^*$
- Query Engine
- Optimised join-aggregate queries
- Model reformulation
- Gradient-descent Trainer
- Model
## Does It Pay Off in Practice?

<table>
<thead>
<tr>
<th>Retailer dataset (records)</th>
<th>excerpt (17M)</th>
<th>full (86M)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Linear regression</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Features (cont+categ)</td>
<td>33 + 55</td>
<td>33+3,653</td>
</tr>
<tr>
<td>Aggregates (cont+categ)</td>
<td>595+2,418</td>
<td>595+145k</td>
</tr>
<tr>
<td>MadLib Learn</td>
<td>1,898.35 sec</td>
<td>&gt; 24h</td>
</tr>
<tr>
<td>R Join (PSQL)</td>
<td>50.63 sec</td>
<td>–</td>
</tr>
<tr>
<td>Export/Import</td>
<td>308.83 sec</td>
<td>–</td>
</tr>
<tr>
<td>Learn</td>
<td>490.13 sec</td>
<td>–</td>
</tr>
<tr>
<td><strong>Our approach</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1core, commodity machine)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Join-Aggregate Converge (runs)</td>
<td>25.51 sec</td>
<td>380.31 sec</td>
</tr>
<tr>
<td></td>
<td>0.02 (343) sec</td>
<td>8.82 (366) sec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Polynomial regression degree 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features (cont+categ)</td>
</tr>
<tr>
<td>Aggregates (cont+categ)</td>
</tr>
<tr>
<td>MadLib Learn</td>
</tr>
<tr>
<td><strong>Our approach</strong></td>
</tr>
<tr>
<td>(1core, commodity machine)</td>
</tr>
<tr>
<td>Join-Aggregate Converge (runs)</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Talk Outline

Current Landscape for ML over DB

Factorised Learning over Normalised Data

Learning under Functional Dependencies

General Problem Formulation
Unified In-Database Analytics for Optimisation Problems

Our target: retail-planning and forecasting applications

- **Typical databases**: weekly sales, promotions, and products
- **Training dataset**: Result of a feature extraction query
- **Task**: Train model to predict additional demand generated for a product due to promotion
- **Training algorithm**: batch gradient descent
- **ML tasks**: ridge linear regression, polynomial regression, factorisation machines; logistic regression, SVM; PCA.
Typical Retail Example

- Database $I = (R_1, R_2, R_3, R_4, R_5)$
- Feature selection query $Q$:

$$Q(sku, store, color, city, country, unitsSold) \leftarrow$$

$$R_1(sku, store, day, unitsSold), R_2(sku, color),$$
$$R_3(day, quarter), R_4(store, city), R_5(city, country).$$

- Free variables
  - Categorical (qualitative):
    $$F = \{sku, store, color, city, country\}.$$  
  - Continuous (quantitative): $unitsSold$.

- Bound variables
  - Categorical (qualitative): $B = \{day, quarter\}$
Typical Retail Example

- We learn the ridge linear regression model

\[ \langle \theta, x \rangle = \sum_{f \in F} \langle \theta_f, x_f \rangle \]

- Training dataset: \( D = Q(I) \)
- Feature vector \( x \) and response \( y = \text{unitsSold} \)

- The parameters \( \theta \) obtained by minimising the objective function:

\[
J(\theta) = \frac{1}{2|D|} \sum_{(x,y) \in D} (\langle \theta, x \rangle - y)^2 + \ell_2-\text{regulariser} \left\| \theta \right\|_2^2
\]
Side Note: One-hot Encoding of Categorical Variables

- **Continuous** variables are mapped to scalars
  - $y_{\text{unitsSold}} \in \mathbb{R}$.

- **Categorical** variables are mapped to indicator vectors
  - country has categories vietnam and england
  - country is then mapped to an indicator vector
    - $x_{\text{country}} = [x_{\text{vietnam}}, x_{\text{england}}]^T \in (\{0, 1\}^2)^T$.
  - $x_{\text{country}} = [0, 1]^T$ for a tuple with country = ‘‘england’’

This encoding leads to wide training datasets and many 0s.
We can solve $\theta^* := \arg\min_{\theta} J(\theta)$ by repeatedly updating $\theta$ in the direction of the gradient until convergence:

$$\theta := \theta - \alpha \cdot \nabla J(\theta).$$
We can solve $\theta^* := \arg \min_\theta J(\theta)$ by repeatedly updating $\theta$ in the direction of the gradient until convergence:

$$\theta := \theta - \alpha \cdot \nabla J(\theta).$$

Define the matrix $\Sigma = (\sigma_{ij})_{i,j \in |F|}$, the vector $c = (c_i)_{i \in |F|}$, and the scalar $s_Y$:

$$\sigma_{ij} = \frac{1}{|D|} \sum_{(x,y) \in D} x_i x_j^\top, \quad c_i = \frac{1}{|D|} \sum_{(x,y) \in D} y \cdot x_i, \quad s_Y = \frac{1}{|D|} \sum_{(x,y) \in D} y^2.$$
We can solve $\theta^* := \arg\min_\theta J(\theta)$ by repeatedly updating $\theta$ in the direction of the gradient until convergence:

$$\theta := \theta - \alpha \cdot \nabla J(\theta).$$

Define the matrix $\Sigma = (\sigma_{ij})_{i,j \in |F|}$, the vector $c = (c_i)_{i \in |F|}$, and the scalar $s_Y$:

$$\sigma_{ij} = \frac{1}{|D|} \sum_{(x,y) \in D} x_i x_j^\top, \quad c_i = \frac{1}{|D|} \sum_{(x,y) \in D} y \cdot x_i, \quad s_Y = \frac{1}{|D|} \sum_{(x,y) \in D} y^2.$$

Then,

$$J(\theta) = \frac{1}{2|D|} \sum_{(x,y) \in D} (\langle \theta, x \rangle - y)^2 + \frac{\lambda}{2} \|\theta\|_2^2$$

$$= \frac{1}{2} \theta^\top \Sigma \theta - \langle \theta, c \rangle + \frac{s_Y}{2} + \frac{\lambda}{2} \|\theta\|_2^2.$$
Σ, c, s_Y can be Expressed as SQL Queries

SQL queries for $\sigma_{ij} = \frac{1}{|D|} \sum_{(x,y) \in D} x_i x_j^\top$ (w/o factor $\frac{1}{|D|}$):

- $x_i, x_j$ continuous $\Rightarrow$ no group-by variable
  
  $\text{SELECT} \ \sum(x_i * x_j) \ \text{FROM} \ D$

- $x_i$ categorical, $x_j$ continuous $\Rightarrow$ one group-by variable
  
  $\text{SELECT} \ x_i, \ \sum(x_j) \ \text{FROM} \ D \ \text{GROUP BY} \ x_i$

- $x_i, x_j$ categorical $\Rightarrow$ two group-by variables
  
  $\text{SELECT} \ x_i, x_j, \ \sum(1) \ \text{FROM} \ D \ \text{GROUP BY} \ x_i, x_j$

where $D$ is the natural join of tables $R_1$ to $R_5$ in our example.

This query encoding avoids drawbacks of one-hot encoding.
Σ, c, s_Y can be Expressed as SQL Queries

SQL queries for $\sigma_{ij} = \frac{1}{|D|} \sum_{(x,y) \in D} x_i x_j^\top$ (w/o factor $\frac{1}{|D|}$):

- $x_i, x_j$ continuous $\Rightarrow$ no group-by variable

  SELECT SUM (x_i * x_j) FROM D;

where $D$ is the natural join of tables $R_1$ to $R_5$ in our example.
\[ \Sigma, c, s_Y \text{ can be Expressed as SQL Queries} \]

SQL queries for \( \sigma_{ij} = \frac{1}{|D|} \sum_{(x,y) \in D} x_i x_j^\top \) (w/o factor \( \frac{1}{|D|} \)):

- \( x_i, x_j \) continuous \( \Rightarrow \) no group-by variable
  
  \[
  \text{SELECT SUM} (x_i \times x_j) \text{ FROM } D;
  \]

- \( x_i \) categorical, \( x_j \) continuous \( \Rightarrow \) one group-by variable
  
  \[
  \text{SELECT } x_i, \text{ SUM}(x_j) \text{ FROM } D \text{ GROUP BY } x_i;
  \]

where \( D \) is the natural join of tables \( R_1 \) to \( R_5 \) in our example.
\( \Sigma \), \( c \), \( s_Y \) can be Expressed as SQL Queries

SQL queries for \( \sigma_{ij} = \frac{1}{|D|} \sum_{(x,y) \in D} x_i x_j^\top \) (w/o factor \( \frac{1}{|D|} \)):

- \( x_i, x_j \) continuous \( \Rightarrow \) no group-by variable
  
  \[
  \text{SELECT} \ \text{SUM} \ (x_i * x_j) \ \text{FROM} \ D ;
  \]

- \( x_i \) categorical, \( x_j \) continuous \( \Rightarrow \) one group-by variable
  
  \[
  \text{SELECT} \ x_i , \ \text{SUM}(x_j) \ \text{FROM} \ D \ \text{GROUP BY} \ x_i ;
  \]

- \( x_i, x_j \) categorical \( \Rightarrow \) two group-by variables
  
  \[
  \text{SELECT} \ x_i , \ x_j , \ \text{SUM}(1) \ \text{FROM} \ D \ \text{GROUP BY} \ x_i , \ x_j ;
  \]

where \( D \) is the natural join of tables \( R_1 \) to \( R_5 \) in our example.
\( \sum \), \( c \), \( s_Y \) can be Expressed as SQL Queries

SQL queries for \( \sigma_{ij} = \frac{1}{|D|} \sum_{(x,y) \in D} x_i x_j^\top \) (w/o factor \( \frac{1}{|D|} \)):

- \( x_i, x_j \) continuous \( \Rightarrow \) no group-by variable
  
  \[
  \text{SELECT \ SUM (} x_i \ * \ x_j \text{) FROM } D ;
  \]

- \( x_i \) categorical, \( x_j \) continuous \( \Rightarrow \) one group-by variable
  
  \[
  \text{SELECT } x_i , \ \text{SUM}(x_j) \text{ FROM } D \ \text{GROUP BY } x_i ;
  \]

- \( x_i, x_j \) categorical \( \Rightarrow \) two group-by variables
  
  \[
  \text{SELECT } x_i , \ x_j , \ \text{SUM}(1) \text{ FROM } D \ \text{GROUP BY } x_i , \ x_j ;
  \]

where \( D \) is the natural join of tables \( R_1 \) to \( R_5 \) in our example.

This query encoding avoids drawbacks of one-hot encoding.
How To Compute Efficiently These Join-Aggregate Queries?
## Factorised Query Computation by Example

<table>
<thead>
<tr>
<th>Orders (O for short)</th>
<th>Dish (D for short)</th>
<th>Items (I for short)</th>
</tr>
</thead>
<tbody>
<tr>
<td>customer</td>
<td>day</td>
<td>dish</td>
</tr>
<tr>
<td>Elise</td>
<td>Monday</td>
<td>burger</td>
</tr>
<tr>
<td>Elise</td>
<td>Friday</td>
<td>burger</td>
</tr>
<tr>
<td>Steve</td>
<td>Friday</td>
<td>hotdog</td>
</tr>
<tr>
<td>Joe</td>
<td>Friday</td>
<td>hotdog</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Consider the natural join of the above relations:

\[ O(\text{customer}, \text{day, dish}), D(\text{dish, item}), I(\text{item, price}) \]
Factorised Query Computation by Example

<table>
<thead>
<tr>
<th>customer</th>
<th>day</th>
<th>dish</th>
<th>item</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elise</td>
<td>Monday</td>
<td>burger</td>
<td>patty</td>
<td>6</td>
</tr>
<tr>
<td>Elise</td>
<td>Monday</td>
<td>burger</td>
<td>onion</td>
<td>2</td>
</tr>
<tr>
<td>Elise</td>
<td>Monday</td>
<td>burger</td>
<td>bun</td>
<td>2</td>
</tr>
<tr>
<td>Elise</td>
<td>Friday</td>
<td>burger</td>
<td>patty</td>
<td>6</td>
</tr>
<tr>
<td>Elise</td>
<td>Friday</td>
<td>burger</td>
<td>onion</td>
<td>2</td>
</tr>
<tr>
<td>Elise</td>
<td>Friday</td>
<td>burger</td>
<td>bun</td>
<td>2</td>
</tr>
</tbody>
</table>

... ...

An algebraic encoding uses product (\(\times\)), union (\(\cup\)), and values:

\[
\begin{align*}
Elise \times Monday \times burger \times patty & \times 6 \cup \\
Elise \times Monday \times burger \times onion & \times 2 \cup \\
Elise \times Monday \times burger \times bun & \times 2 \cup \\
Elise \times Friday \times burger \times patty & \times 6 \cup \\
Elise \times Friday \times burger \times onion & \times 2 \cup \\
Elise \times Friday \times burger \times bun & \times 2 \cup \ldots
\end{align*}
\]
There are several algebraically equivalent factorised joins defined by distributivity of product over union and their commutativity.
Observation:

- price is under item, which is under dish, but only depends on item,
- .. so the same price appears under an item regardless of the dish.

Idea: Cache price for a specific item and avoid repetition!
Same Data, Different Factorisation

day
  └── customer
      └── dish
          └── item
              │   │   │   │
              patty bun onion
              |     |     |
              x     x     x
              U     U     U
              6     2     2

Monday
  └── Elise
      └── burger
          └── patty
              └── bun
                  └── onion
                      └── x
                          U
                          6

Friday
  └── Elise
      └── burger
          └── patty
              └── bun
                  └── onion
                      └── x
                          U
                          6

Elise
  └── burger
      └── patty
          └── bun
              └── onion
                  └── x
                      U
                      6

Joe
  └── hotdog
      └── bun
          └── onion
              └── sausage
                  └── x
                      U
                      U
                      2 2 4

Steve
  └── hotdog
      └── bun
          └── onion
              └── sausage
                  └── x
                      U
                      U
                      2 2 4
Grounding Variable Orders to Factorised Joins

Our join: $O(\text{customer, day, dish}), D(\text{dish, item}), I(\text{item, price})$
can be grounded to a factorised join as follows:

$\bigcup O(\_, \_, \text{dish}), D(\text{dish, item}) \times \bigcup O(\text{customer, day, dish}), I(\text{item, price})$

This grounding follows the previous variable order.
Grounding Variable Orders to Factorised Joins

\[ \bigcup_{O(-,\text{dish}), D(\text{dish},-)} \text{dish} \]

\[ \times \]

\[ \bigcup_{O(-,\text{day},\text{dish})} \text{day} \]
\[ \times \]
\[ \bigcup_{O(\text{customer},\text{day},\text{dish})} \text{customer} \]

\[ \bigcup_{D(\text{dish},\text{item})} \text{item} \]
\[ \times \]
\[ \bigcup_{I(\text{item},\text{price})} \text{price} \]

- Relations sorted following topological order of the variable order

- Intersection of \( O \) and \( D \) on \( \text{dish} \) in time \( \tilde{O}(\min(|\pi_{\text{dish}} O|, |\pi_{\text{dish}} D|)) \)

- The remaining operations are lookups in the relations, where we first fix the \( \text{dish} \) value and then the \( \text{day} \) and \( \text{item} \) values
COUNT(*) computed in one pass over the factorisation:

- values $\mapsto 1$,
- $\cup \mapsto +$,
- $\times \mapsto \ast$. 
Factorising the Computation of Aggregates (1/2)

\[
\text{COUNT}(* \to 1,
\text{UNION} \to +,
\text{PRODUCT} \to *.
\]
SUM(dish * price) computed in one pass over the factorisation:

- Assume there is a function $f$ that turns dish into reals.
- All values except for dish & price $\mapsto 1$,
- $\cup \mapsto +$,
- $\times \mapsto *$. 
SUM(dish * price) computed in one pass over the factorisation:

- Assume there is a function $f$ that turns dish into reals.
- All values except for dish & price $\mapsto 1$,
- $\cup \mapsto +$,
- $\times \mapsto \ast$. 

$$20f(burger) + 16f(hotdog)$$
Talk Outline

Current Landscape for ML over DB

Factorised Learning over Normalised Data

Learning under Functional Dependencies

General Problem Formulation
Consider the functional dependency \( \text{city} \rightarrow \text{country} \) and

- country categories: vietnam, england
- city categories: saigon, hanoi, oxford, leeds, bristol

The one-hot encoding enforces the following identities:

- \( x_{\text{vietnam}} = x_{\text{saigon}} + x_{\text{hanoi}} \)
  country is vietnam \( \Rightarrow \) city is either saigon or hanoi
  \( x_{\text{vietnam}} = 1 \Rightarrow \) either \( x_{\text{saigon}} = 1 \) or \( x_{\text{hanoi}} = 1 \)

- \( x_{\text{england}} = x_{\text{oxford}} + x_{\text{leeds}} + x_{\text{bristol}} \)
  country is england \( \Rightarrow \) city is either oxford, leeds, or bristol
  \( x_{\text{england}} = 1 \Rightarrow \) either \( x_{\text{oxford}} = 1 \) or \( x_{\text{leeds}} = 1 \) or \( x_{\text{bristol}} = 1 \)
Model Reparameterisation using Functional Dependencies

• Identities due to one-hot encoding
  \[ x_{\text{vietnam}} = x_{\text{saigon}} + x_{\text{hanoi}} \]
  \[ x_{\text{england}} = x_{\text{oxford}} + x_{\text{leeds}} + x_{\text{bristol}} \]

• Encode \( x_{\text{country}} \) as \( x_{\text{country}} = Rx_{\text{city}} \), where

\[
R = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & \text{vietnam} \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & \text{england}
\end{bmatrix}
\]

For instance, if city is saigon, i.e., \( x_{\text{city}} = [1, 0, 0, 0, 0, 0]^\top \), then country is vietnam, i.e., \( x_{\text{country}} = Rx_{\text{city}} = [1, 0]^\top \).

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
1 \\
0 \\
0 \\
0
\end{bmatrix}
= \begin{bmatrix}
1 \\
0
\end{bmatrix}
\]
• Functional dependency: city → country
• $x_{\text{country}} = Rx_{\text{city}}$
• Replace all occurrences of $x_{\text{country}}$ by $Rx_{\text{city}}$:

$$
\sum_{f \in F - \{\text{city}, \text{country}\}} \langle \theta_f, x_f \rangle + \langle \theta_{\text{country}}, x_{\text{country}} \rangle + \langle \theta_{\text{city}}, x_{\text{city}} \rangle
$$

$$
= \sum_{f \in F - \{\text{city}, \text{country}\}} \langle \theta_f, x_f \rangle + \langle \theta_{\text{country}}, Rx_{\text{city}} \rangle + \langle \theta_{\text{city}}, x_{\text{city}} \rangle
$$

$$
= \sum_{f \in F - \{\text{city}, \text{country}\}} \langle \theta_f, x_f \rangle + \langle R^T \theta_{\text{country}} + \theta_{\text{city}}, x_{\text{city}} \rangle_{\gamma_{\text{city}}}
$$
Model Reparameterisation using Functional Dependencies

- Functional dependency: $\text{city} \rightarrow \text{country}$
- $x_{\text{country}} = Rx_{\text{city}}$
- Replace all occurrences of $x_{\text{country}}$ by $Rx_{\text{city}}$:

$$\sum_{f \in F - \{\text{city}, \text{country}\}} \langle \theta_f, x_f \rangle + \langle \theta_{\text{country}}, x_{\text{country}} \rangle + \langle \theta_{\text{city}}, x_{\text{city}} \rangle$$

$$\sum_{f \in F - \{\text{city}, \text{country}\}} \langle \theta_f, x_f \rangle + \langle \theta_{\text{country}}, Rx_{\text{city}} \rangle + \langle \theta_{\text{city}}, x_{\text{city}} \rangle$$

$$\sum_{f \in F - \{\text{city}, \text{country}\}} \langle \theta_f, x_f \rangle + \left( R^T \theta_{\text{country}} + \theta_{\text{city}}, x_{\text{city}} \right)$$

- We avoid the computation of the aggregates over $x_{\text{country}}$.
- We reparameterise and ignore parameters $\theta_{\text{country}}$.
- What about the penalty term in the loss function?
• Functional dependency: \( \text{city} \rightarrow \text{country} \)

• \( x_{\text{country}} = Rx_{\text{city}} \quad \gamma_{\text{city}} = R^\top \theta_{\text{country}} + \theta_{\text{city}} \)

• Rewrite the penalty term

\[
\| \theta \|^2 = \sum_{j \neq \text{city}} \| \theta_j \|^2 + \| \gamma_{\text{city}} - R^\top \theta_{\text{country}} \|^2 + \| \theta_{\text{country}} \|^2
\]

• Optimise out \( \theta_{\text{country}} \) by expressing it in terms of \( \gamma_{\text{city}} \):

\[
\theta_{\text{country}} = (I_{\text{country}} + RR^\top)^{-1} R \gamma_{\text{city}} = R(I_{\text{city}} + R^\top R)^{-1} \gamma_{\text{city}}
\]

• The penalty term becomes

\[
\| \theta \|^2 = \sum_{j \neq \text{city}} \| \theta_j \|^2 + \left\langle (I_{\text{city}} + R^\top R)^{-1} \gamma_{\text{city}}, \gamma_{\text{city}} \right\rangle
\]
Talk Outline

Current Landscape for ML over DB

Factorised Learning over Normalised Data

Learning under Functional Dependencies

General Problem Formulation
General Problem Formulation

We want to solve $\theta^* := \arg \min_\theta J(\theta)$, where

$$J(\theta) := \sum_{(x,y) \in D} \mathcal{L} (\langle g(\theta), h(x) \rangle, y) + \Omega(\theta).$$

- $\theta = (\theta_1, \ldots, \theta_p) \in \mathbb{R}^p$ are parameters
- functions $g : \mathbb{R}^p \to \mathbb{R}^m$ and $h : \mathbb{R}^n \to \mathbb{R}^m$
  - $g = (g_j)_{j \in [m]}$ is a vector of multivariate polynomials
  - $h = (h_j)_{j \in [m]}$ is a vector of multivariate monomials
- $\mathcal{L}$ is a loss function, $\Omega$ is the regulariser
- $D$ is the training dataset with features $x$ and response $y$.

Problems: ridge linear regression, polynomial regression, Factorisation machines; logistic regression, SVM; PCA.
Special Case: Ridge Linear Regression

Under

- square loss $\mathcal{L}$, $\ell_2$-regularisation,
- data points $\mathbf{x} = (x_0, x_1, \ldots, x_n, y)$,
- $p = n + 1$ parameters $\mathbf{\theta} = (\theta_0, \ldots, \theta_n)$,
- $x_0 = 1$ corresponds to the bias parameter $\theta_0$,
- identity functions $g$ and $h$,

we obtain the following formulation for ridge linear regression:

$$J(\mathbf{\theta}) := \frac{1}{2|D|} \sum_{(x,y) \in D} (\langle \mathbf{\theta}, \mathbf{x} \rangle - y)^2 + \frac{\lambda}{2} \|\mathbf{\theta}\|_2^2.$$
Special Case: Degree-\(d\) Polynomial Regression

Under

- square loss \(\mathcal{L}\), \(\ell_2\)-regularisation,
- data points \(\mathbf{x} = (x_0, x_1, \ldots, x_n, y)\),
- \(p = m = 1 + n + n^2 + \cdots + n^d\) parameters \(\theta = (\theta_a)\), where \(a = (a_1, \ldots, a_n)\) with non-negative integers s.t. \(\|a\|_1 \leq d\).
- the components of \(h\) are given by \(h_a(x) = \prod_{i=1}^{n} x_i^{a_i}\),
- \(g(\theta) = \theta\),

we obtain the following formulation for polynomial regression:

\[
J(\theta) := \frac{1}{2|D|} \sum_{(x,y) \in D} \left( \langle g(\theta), h(x) \rangle - y \right)^2 + \frac{\lambda}{2} \|\theta\|_2^2.
\]
Special Case: Factorisation Machines

Under

- square loss $\mathcal{L}$, $\ell_2$-regularisation,
- data points $\mathbf{x} = (x_0, x_1, \ldots, x_n, y)$,
- $p = 1 + n + r \cdot n$ parameters,
- $m = 1 + n + \binom{n}{2}$ features,

we obtain the following formulation for degree-2 rank-$r$ Factorisation machines:

$$
J(\theta) := \frac{1}{2|D|} \sum_{(x, y) \in D} \left( \sum_{i=0}^{n} \theta_i x_i + \sum_{\{i, j\} \in \binom{[n]}{2}} \theta_i^{(\ell)} \theta_j^{(\ell)} x_i x_j - y \right)^2 + \frac{\lambda}{2} \|\theta\|^2_2.
$$
Special Case: Classifiers

- Typically, the regulariser is $\frac{\lambda}{2} \|\theta\|^2_2$
- The response is binary: $y \in \{\pm 1\}$
- The loss function $L(\gamma, y)$, where $\gamma := \langle g(\theta), h(x) \rangle$ is
  - $L(\gamma, y) = \max\{1 - y\gamma, 0\}$ for support vector machines,
  - $L(\gamma, y) = \log(1 + e^{-y\gamma})$ for logistic regression,
  - $L(\gamma, y) = e^{-y\gamma}$ for Adaboost.
Zoom-in: In-database vs. Out-of-database Learning

Feature extraction

Query \( R_1 \times \ldots \times R_k \)

DB

Queries:

- \( \sigma_{11} \)
- \( \ldots \)
- \( \sigma_{ij} \)
- \( \ldots \)
- \( c_1 \)
- \( \ldots \)

ML tool

| \( D \) |

Cost \( \leq N^{faqw} \ll |D| \)

Gradient-descent

\( J(\theta) \nabla J(\theta) \)

Converged?

Yes

No

Factorised query evaluation

\( h \)

\( g \)

Optimiser

Model reformulation

Model
Thank you!