Relational QPs
Exploiting Symmetries for Modelling and Solving QPs

Kristian Kersting - Exploiting Symmetries for Modelling and Solving QPs
Take-away message

Statistical Machine Learning (ML) needs a crossover with data and programming abstractions

- ML high-level languages increase the number of people who can successfully build ML applications and make experts more effective
- To deal with the computational complexity, we need ways to automatically reduce the solver costs
Arms race to deeply understand data
Bottom line:
Take your data spreadsheet ...
Kristian Kersting  -  Exploiting Symmetries for Modelling and Solving QPs

... and apply machine learning

Gaussian Processes

Features

Is it really that simple?

Graphical models

CH₃

Graph Mining

H₃C

Diffusion Models

and many more …
Complex data networks abound

[Lu, Krishna, Bernstein, Fei-Fei „Visual Relationship Detection“ CVPR 2016]

Actually, most data in the world stored in relational databases
Punshline: Two trends that drive ML

1. Arms race to deeply understand data
2. Data networks of a large number of formats

It costs considerable human effort to develop, for a given dataset and task, a good ML algorithm

Crossover of ML with data & programming abstractions

- make the ML expert more effective
- increases the number of people who can successfully build ML applications
Thinking Machine Learning

Machine Learning Database
(data, weighted rules, loops and data structures)

Model Rules and Domain Knowledge
Representation Learning
DM and ML Algorithms
Graph Kernels
Diffusion Processes
Random Walks
Decision Trees
Frequent Itemsets
SVMs
Graphical Models
Topic Models
Gaussian Processes
Autoencoder
Matrix and Tensor Factorization
Reinforcement Learning
...

Features and Data Rules

External Databases

Feature Extraction
Declarative Learning Programming
Symbolic-Numerical Solver

Inference Results

Features and Data Rules

Feedback/AutoDM

This connects the CS communities

**Jim Gray** Turing Award 1998
“Automated Programming”

**Mike Stonebraker** Turing Award 2014
“One size does not fit all”

Data Mining/Machine Learning, Databases, AI, Model Checking, Software Engineering, Optimization, Knowledge Representation, Constraint Programming, Operation Research, … !
The Machine Learning Genome

The ML Genome is a dataset, a knowledge base, an ongoing effort to learn and reason about ML concepts.
A simple example

What is the problem that the first card of a randomly shuffled deck with 52 cards is an Ace?

How would a machine solve this?
One option is to treat this as an inference problem within in a graphical model, solved approximately using some mathematical program!
A simple example

- card (1,d2)
- card (1,d3)
- ... card (1,pAce)
- card (52,d2)
- card (52,d3)
- ... card (52,pAce)
A simple example

card (1,d2)
card (1,d3)
⋯
card (1,pAce)
card (52,d2)
card (52,d3)
⋯
card (52,pAce)
We do not want to write down all the rules!
Faster modelling

Let’s use **programming abstractions** such as e.g.

\[ w1: \forall p, x, y: \text{card}(P, X), \text{card}(P, Y) \Rightarrow x = y \]
\[ w2: \forall c, x, y: \text{card}(X, C), \text{card}(Y, C) \Rightarrow x = y \]

We do not want to write down all the rules!
A simple example

What about inference?

card (1, d2)  card (1, d3)  ...  card (1, pAce)

card (52, d2)  card (52, d3)  ...  card (52, pAce)
A simple example

No independencies.  Fully connected.

$2^{2704}$ states
A simple example

A machine will not solve the problem
What are we missing?

Positions and cards are exchangable but the machine is not aware of these symmetries
Let’s use programming abstractions together with symmetry- and language-aware solvers.

Positions and cards are exchangable but the machine is not aware of these symmetries.
Let’s say we want to classify publications into scientific disciplines
**Classification using LP SVMs**

[Bennett’99; Mangasarian’99; Zhou, Zhang, Jiao’02, ...]

\[ H^* = \{ \tilde{x} \mid \langle \tilde{x}, \tilde{\beta} \rangle + \beta_0 = 0 \} \]

\[ d(H_1, H_2) = \frac{2}{||\tilde{\beta}||} \]

Replace \( l_2 \)- by \( l_1 \)-, \( l_\infty \)-norm in the standard SVM prog.
Relational Data and Program Abstractions

[Kersting, Mladenov, Tokmakov AIJ’15, Mladenov, Heinrich, Kleinhans, Gonsio, Kersting DeLBP’16]

The machine compiles it into solver form.

http://www-ai.cs.uni-dortmund.de/weblab/static/RLP/html/

RELOOP: A Toolkit for Relational Convex Optimization

Embedded within Python s.t. loops and rules can be used

Logically parameterized LP variable (set of ground LP variables)

Logically parameterized LP objective

Logically parameterized LP constraint

Write down the LP-SVM in "paper form". The machine compiles it into solver form.
But wait, publications are citing each other. OMG, I have to use graph kernels!

REALLY?
Relational Data and Program Abstractions

[Kersting, Mladenov, Tokmakov AIJ’15, Mladenov, Heinrich, Kleinhans, Gonsio, Kersting DeLBP’16]

Lifted LP-SVM

Collective constraints

Logical query defines scope of abstract constraint

Citing papers share topics

No kernel, the structure is expressed within the constraints!

```
var pred/1;  # predicted label for unlabeled instances
var slack/1; # the slacks
var coslack/2; # slack between neighboring instances
var weight/1; # the slope of the hyperplane
var b/0;      # the intercept of the hyperplane
var r/0;      # margin

slack = sum{label(I)} slack(I);
coslack = sum{cite(I1,I2),label(I1),query(I2)} slack(I1,I2)

minimize: -r + C(1) * slack + C(2) * coslack;

subject to forall {I in label(I)}: pred(I) = innerProd(I) + b;
# related instances should have the same labels.
subject to forall {I1, I2 in cite(I1,I2), label(I1), query(I2)}:
  label(I1) * pred(I2) + slack(I1, I2) >= r;
  # the symmetric case
subject to forall {I1, I2 in cite(I1,I2), label(I2), query(I1)}:
  label(I2) * pred(I1) + slack(I1, I2) >= r;

subject to forall {I in label(I)}:
  label(I)*(innerProd(I) + b) + slack(I) >= r;
# weights are between -1 and 1
subject to forall {J in attribute(_, J)}: -1 <= weight(J) <= 1;
subject to: r >= 0;    # the margin is positive
subject to forall {I in label(I)}: slack(I) >= 0;  # slacks are positive
```
OK, we have now a high-level, declarative language for mathematical programming.

HOW CAN THE MACHINE NOW HELP TO REDUCE THE SOLVER COSTS?
Lifted Mathematical Programming
Exploiting computational symmetries

[Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]
Lifted Mathematical Programming
Exploiting computational symmetries

\[ \max_{[x,y,z]^T \in \mathbb{R}^3} \quad 0x + 0y + 1z \]
\[ \text{s.t.} \quad \begin{bmatrix}
1 & 1 & 1 \\
-1 & 0 & 0 \\
0 & -1 & 0 \\
1 & 1 & -1 \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
\end{bmatrix} \leq \begin{bmatrix}
1 \\
0 \\
0 \\
-1 \\
\end{bmatrix} \]

View the mathematical program as a colored graph

Reduce the MP by running Weisfeiler-Lehman on the MP-Graph
**Weisfeiler-Lehman (WL) aka “naive vertex classification”**

Basic subroutine for GI testing
Computes LP-relaxations of GA-ILP, fractional automorphisms
**Quasi-linear** running time $O((n+m)\log(n))$ when using asynchronous updates [Berkholz, Bonsma, Grohe ESA´13]

**Part of graph tool SAUCY** [See e.g. Darga, Sakallah, Markov DAC´08]

Has lead to highly performant graph kernels [Shervashidze, Schweitzer, van Leeuwen, Mehlhorn, Borgwardt JMLR 12:2539-2561 ’11]

Can be extended to weighted graphs/real-valued matrices [Grohe, Kersting, Mladenov, Selman ESA´14]

Actually a Frank-Wolfe optimizer and can be viewed as recursive spectral clustering [Kersting, Mladenov, Garnett, Grohe AAAI´14]
Compression: Coloring the graph

[Kersting, Ahmadi, Natarajan UAI’09; Ahmadi, Kersting, Mladenov, Natarajan MLJ’13, Mladenov, Ahmadi, Kersting AISTATS’12, Grohe, Kersting, Mladenov, Selman ESA’14, Kersting, Mladenov, Tokmatov AIJ’17]

Color nodes initially with the same color, say red

Color factors distinctively according to their equivalences. For instance, assuming \( f_1 \) and \( f_2 \) to be identical and B appears at the second position within both, say blue
**Compression: Pass colors around**

[Kersting, Ahmadi, Natarajan UAI’09; Ahmadi, Kersting, Mladenov, Natarajan MLJ’13, Mladenov, Ahmadi, Kersting AISTATS’12, Grohe, Kersting, Mladenov, Selman ESA’14, Kersting, Mladenov, Tokmatov AIJ’17]

1. Each factor collects the colors of its neighboring nodes
Compression: Pass colors around

[Kersting, Ahmadi, Natarajan UAI’09; Ahmadi, Kersting, Mladenov, Natarajan MLJ’13, Mladenov, Ahmadi, Kersting AISTATS’12, Grohe, Kersting, Mladenov, Selman ESA’14, Kersting, Mladenov, Tokmatov AIJ’17]

1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs“ its color signature with its own color
Compression: Pass colors around

1. Each factor collects the colors of its neighboring nodes
2. Each factor "signs" its color signature with its own color
3. Each node collects the signatures of its neighboring factors
1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs“ its color signature with its own color
3. Each node collects the signatures of its neighboring factors
4. Nodes are recolored according to the collected signatures
Compression: Pass colors around

1. Each factor collects the colors of its neighboring nodes
2. Each factor “signs“ its color signature with its own color
3. Each node collects the signatures of its neighboring factors
4. Nodes are recolored according to the collected signatures
5. If no new color is created stop, otherwise go back to 1

[Kersting, Ahmadi, Natarajan UAI’09; Ahmadi, Kersting, Mladenov, Natarajan MLJ’13, Mladenov, Ahmadi, Kersting AISTATS’12, Grohe, Kersting, Mladenov, Selman ESA’14, Kersting, Mladenov, Tokmatov AIJ’17]
Lifted Mathematical Programming
Exploiting computational symmetries

[Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]

Weisfeiler-Lehman in quasi-linear time
automatically compressed

Run Solver

Big Model

Small Model

Run Solver
Collective Classification
Cora (most common vs. rest)

The more observed the more lifting
Faster end-to-end even in the light of Gurobi’s fast pre-solving heuristics
As also noted by Stephen Boyd

**Dense vs. sparse is not enough, solvers need to be aware of symmetries**
Feasible region of LP and the objective vectors

Span of the fractional auto-morphism of the LP

Projections of the feasible region onto the span of the fractional auto-morphism

Why does this work?
Compute Equitable Partition (EP) of the LP using WL

\[ \mathcal{P} = \{ P_1, \ldots, P_p; Q_1, \ldots, Q_q \} \]

Partition of LP variables
Partition of LP constraints

Intuitively, we group together variables resp. constraints that interact in the very same way in the LP.
Fractional Automorphisms of LPs

The EP induces a fractional automorphism of the coefficient matrix $A$

$$X_QA = AX_P$$

where $X_Q$ and $X_P$ are doubly-stochastic matrixes (relaxed form of automorphism)

$$(X_P)_{ij} = \begin{cases} 
1/|P| & \text{if both vertices } i, j \text{ are in the same } P, \\
0 & \text{otherwise.}
\end{cases}$$

$$(X_Q)_{ij} = \begin{cases} 
1/|Q| & \text{if both vertices } i, j \text{ are in the same } Q, \\
0 & \text{otherwise}
\end{cases}$$
Fractional Automorphisms Preserve Solutions

If $\mathbf{x}$ is feasible, then $\mathbf{X}_\rho \mathbf{x}$ is feasible, too.

By induction, one can show that left-multiplying with a double-stochastic matrix preserves directions of inequalities; they are averagers. Hence,

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \Rightarrow \mathbf{X}_Q \mathbf{A}\mathbf{x} \leq \mathbf{X}_Q \mathbf{b} \iff \mathbf{A}\mathbf{X}_P \mathbf{x} \leq \mathbf{b}$$
Fractional Automorphisms Preserve Solutions

If $\mathbf{x}^*$ is optimal, then $\mathbf{X}_p \mathbf{x}^*$ is optimal, too.

Since by construction $\mathbf{c}^T \mathbf{X}_P = \mathbf{c}^T$ and hence

$$
\mathbf{c}^T (\mathbf{X}_P \mathbf{x}) = \mathbf{c}^T \mathbf{x}
$$
What have we established so far?

Instead of considering the original LP

$$(A, b, c)$$

It is sufficient to consider

$$(A X_P, b, X_P^T c)$$

i.e. we “average” parts of the polytope.

But why is this dimensionality reduction?
Dimensionality Reduction

The doubly-stochastic matrix $X_P$ can be written as

$$X_P = BB^T$$

$$B_{iP} = \begin{cases} \frac{1}{\sqrt{|P|}} & \text{if vertex } i \text{ belongs to part } P, \\ 0 & \text{otherwise.} \end{cases}$$

Since the column space of $B$ is equivalent to the span of $X_P$, it is actually sufficient to consider only

$$(AB_P, b, B_P^Tc)$$

This is of reduced size, and actually we can also drop any constraint that becomes identical.
WL induces a Fractional Automorphism of the LP
Approximate probabilistic inference closely connected to LPs

\[ \hat{x} \in \arg \max_{x \in \mathcal{X}^N} \left\{ \sum_{s \in V} \theta_s(x_s) + \sum_{(s,t) \in E} \theta_{st}(x_s, x_t) \right\} \]
Lifted Optimization

**Attention:** For special-purpose solvers such as message-passing (coordinate descent, ) for probabilistic inference we may have to reparameterize the lifted model.

[Mladenov, Globerson, Kersting UAI 2014; Mladenov, Kersting UAI 2015]
Lifted probabilistic inference = Inference in a smaller, reparameterized model

[MLadenov, Globerson, Kersting UAI 2014; MLadenov, Kersting UAI 2015]
Holds also for Convex QPs

\[ x^* = \arg \min_{x \in \mathcal{D}} J(x) \]
\[ J(x) = x^T Q x + c^T x \]
\[ \mathcal{D} = \{ x : Ax \leq b \} \]

Mladenov, Kleinhans, Kersting AAAI´17

On par with state-of-the-art by just four lines of code

Papers that cite each other should be on the same side of the hyperplane
A geometric interpretation

Mladenov, Kleinhans, Kersting AAAI’17

For QPs, a fractional automorphism is a rotation and scaling (of the semidefinite factors B of the Gram matrix) fractional automorphism

Relaxed by scaling
No symmetry-based ML?

Indeed, one may argue that the (rotational) automorphism group of most Euclidean datasets consists of the identity transformation alone: symmetries of a given dataset B can easily be destroyed by slightly perturbing the body.

No, we can have approximate fractional automorphisms (for SVMs)

Mladenov, Kleinhans, Kersting AAAI’17

Whitening + K-means of sorted distance vectors
This provides a symmetry argument for known data reduction methods used for SVMs

Mladenov, Kleinhans, Kersting AAAI´17
Approximately Lifted SVM:
Cluster data points via K-means using sorted distance vectors.
Solve SVM on cluster representatives only

PAC-style generalization bound:
the approximately lifted SVM will very likely have a small expected error rate if it has a small empirical loss over the original dataset.

Symmetry-based Data Programming: fractional autom. of label-preserving data transformations

Same should work for deep networks

380x faster
And, there are other “-02”, “-03”, … flags, e.g. symbolic-numerical interior point solvers

\[
\begin{align*}
\text{Formulae parse trees} & \quad \rightarrow \quad \text{Algebraic Decision Diagrams} \\
\begin{pmatrix} 5 \\ 4 \end{pmatrix} & \quad + \quad \begin{pmatrix} 3 \end{pmatrix} \\
\end{align*}
\]

Matrix Free Optimization

All this opens the general machine learning toolbox for declarative machines: feature selection, least-squares regression, label propagation, ranking, collaborative filtering, community detection, deep learning, …
Relations and (fractional) automorphisms are a natural foundation for

**SYMMETRY-BASED ML AND DATA PROGRAMMING**

[GENS, DOMINGOS NIPS 2014; RATNER ET AL. NIPS 2016]

- Learning (rich) representations is a central problem of machine learning
- (Fractional) symmetry / group theory provide a natural foundation for learning representations
- Symmetries = “unimportant” variants of data (graphs, relational structures, …)
- “Unimportant” variants programmed via declarative rules
- Let’s move beyond QPs: CSPs, SDPs, Deep Networks, …
Together with high-level languages

THINKING MACHINE LEARNING

• Shortens data science code to make ML techniques faster to write and easier to understand
• Reduces the level of expertise necessary to build ML applications
• Facilitates the construction of more sophisticated ML that incorporate rich domain knowledge and separate queries from underlying code
• Supports the construction of integrated ML machines thank think across a wide variety of domains and tool types
• Accelerates ML machines by exploiting language properties, compression, and compilation

Thanks for your attention!
Kristian Kersting - Exploiting Symmetries for Modelling and Solving QPs