# Deep machines that know when they do not know

and how to exploite symmetries for modelling and solving quadratic programs

Kristian Kersting

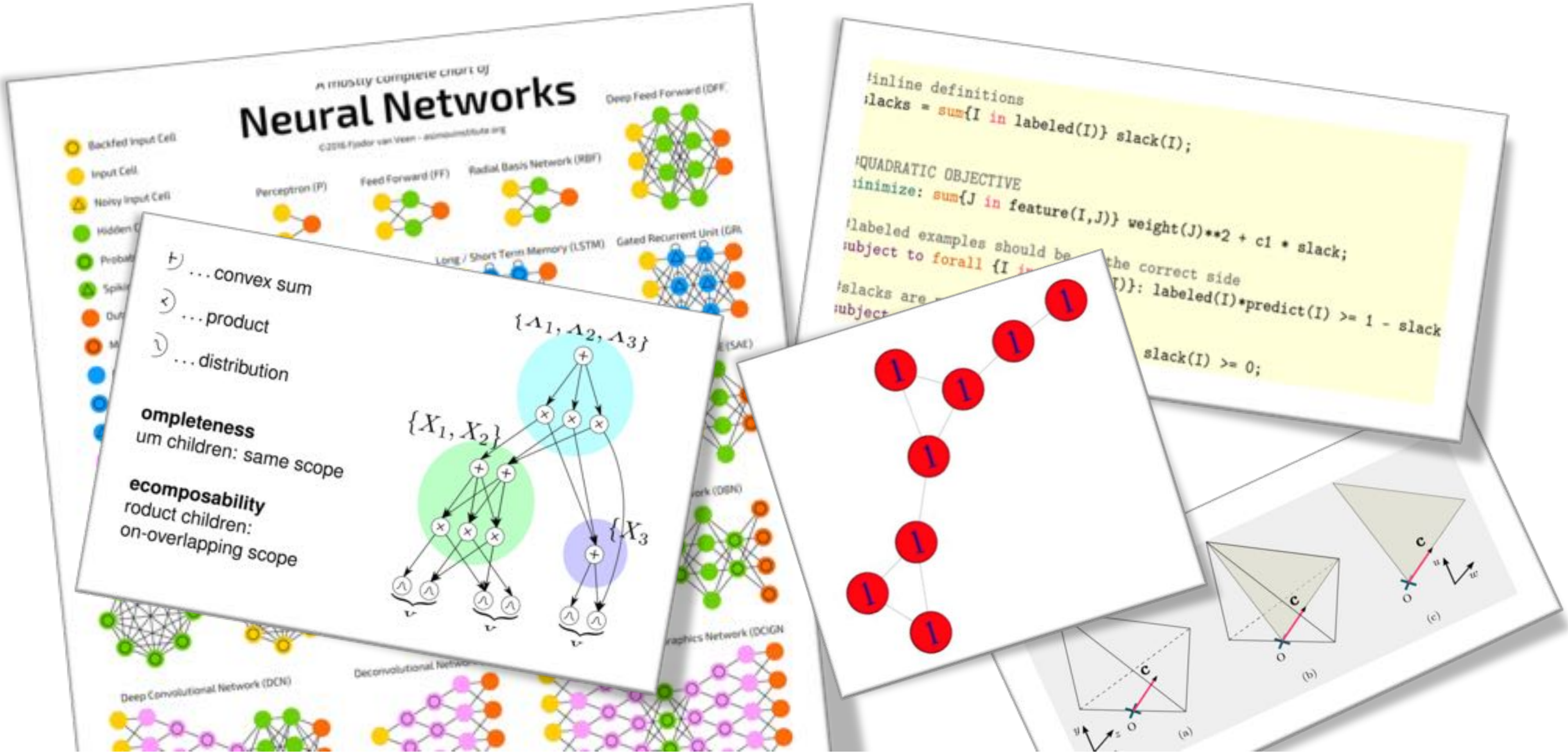
TECHNISCHE UNIVERSITÄT DARMSTADT

Centre for Cognitive Science

Fachbereich Informatik

ELLIS European Laboratory for Learning and Intelligent Systems

CLAIRE CONFEDERATION OF LABORATORIES FOR ARTIFICIAL INTELLIGENCE RESEARCH IN EUROPE



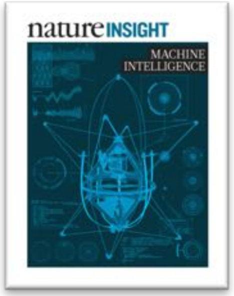
# AI and ML have a strong impact


Data are now ubiquitous; there is great value from understanding this data, building models and making predictions

However, there are not enough data scientists, statisticians, machine learning and AI experts

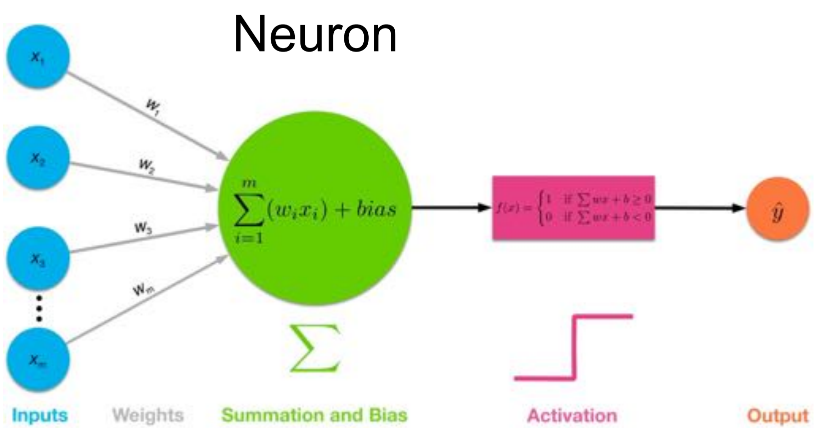
Provide the foundations, algorithms, and tools to develop systems that ease or even automate AI model discovery from data as much as possible

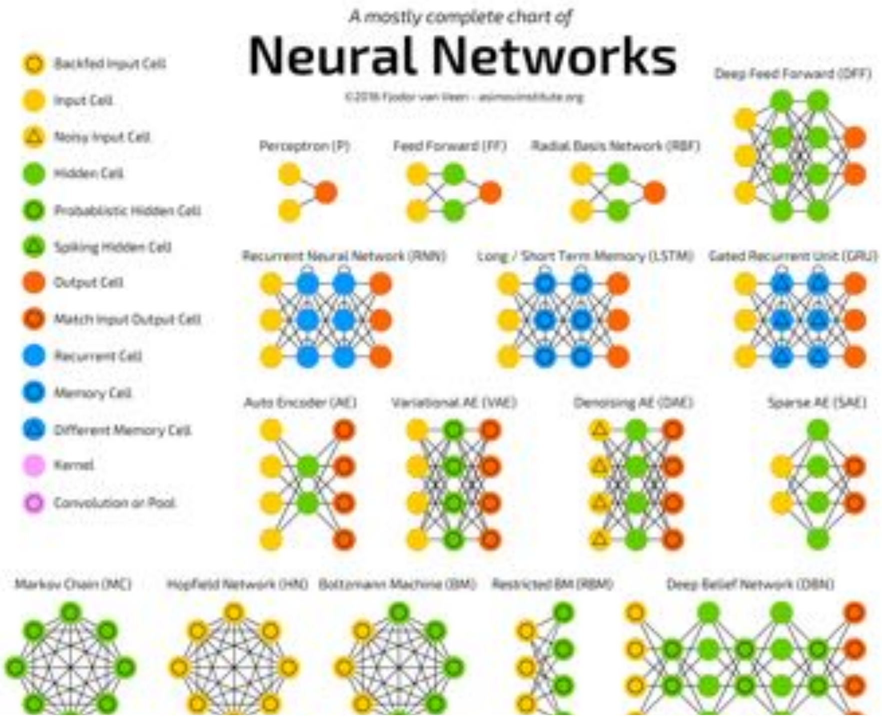
# Deep Neural Networks




Potentially much more powerful than shallow architectures, represent computations

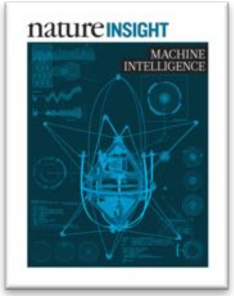
[LeCun, Bengio, Hinton Nature 521, 436–444, 2015]



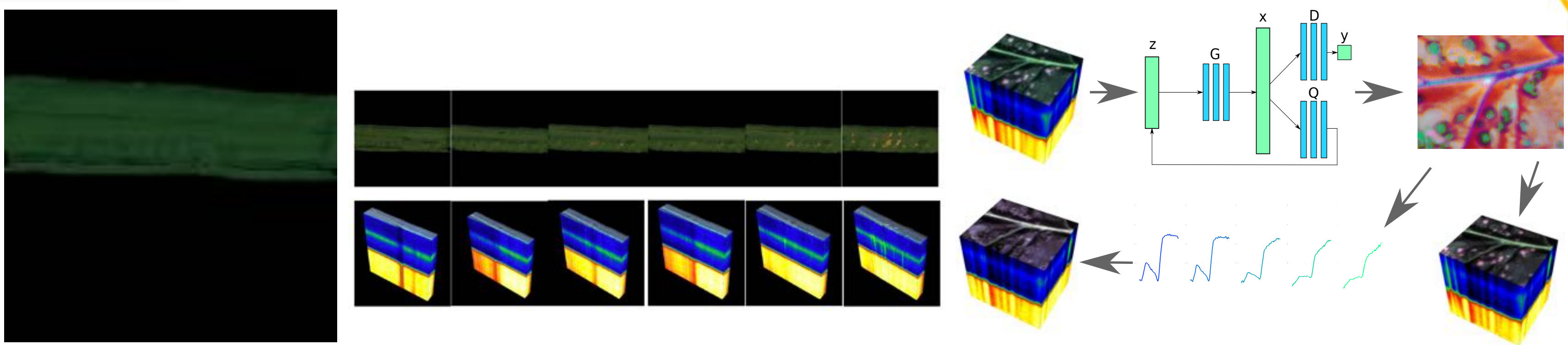

**Differentiable Programming**

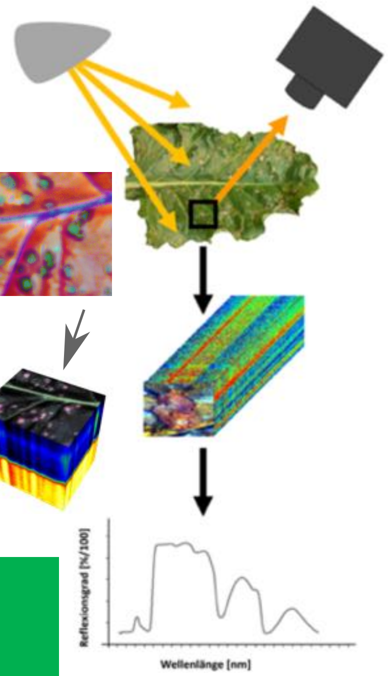
# Deep Neural Networks




Potentially much more powerful than shallow architectures, represent computations

[LeCun, Bengio, Hinton Nature 521, 436–444, 2015]



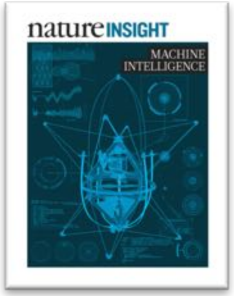

**They "develop intuition" about complicated biological processes and generate scientific data**

[Schramowski, Brugger, Mahlein, Kersting 2019]

DePhenSe

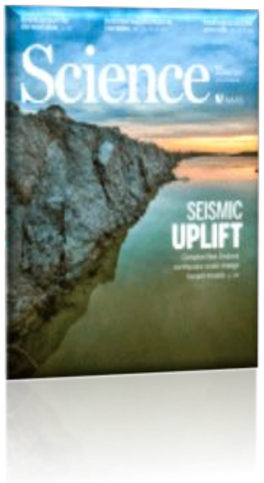
Bundesanstalt für Landwirtschaft und Ernährung

# Deep Neural Networks




Potentially much more powerful than shallow architectures, represent computations

[LeCun, Bengio, Hinton Nature 521, 436–444, 2015]




SHARE

1.02k

0

REPORTS | PSYCHOLOGY

## Semantics derived automatically from language corpora contain human-like biases

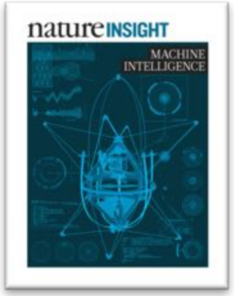
Aylin Caliskan[1,*], Joanna J. Bryson[1,2,*], Arvind Narayanan[1,*]

+ See all authors and affiliations

Science 14 Apr 2017:
Vol. 356, Issue 6334, pp. 183-186
DOI: 10.1126/science.aal4230

**They “capture” stereotypes from human language**

# Deep Neural Networks

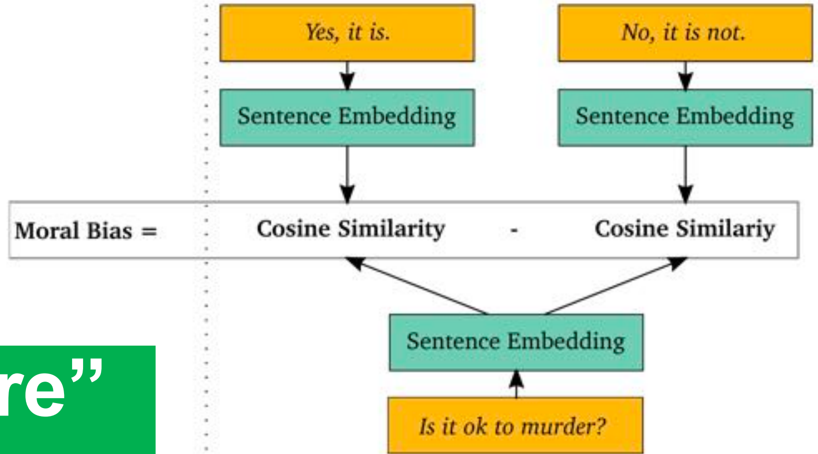



Potentially much more powerful than shallow architectures, represent computations

[LeCun, Bengio, Hinton Nature 521, 436–444, 2015]

## The Moral Choice Machine

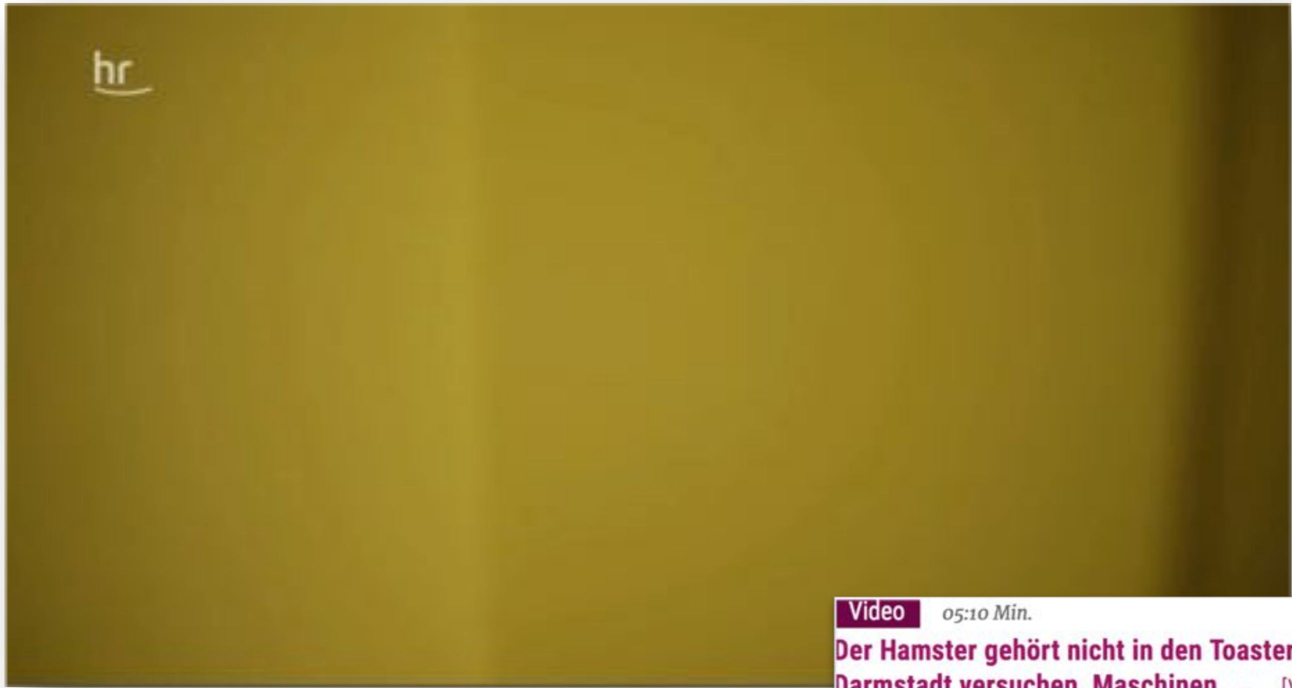
| **Dos** | WEAT | Bias | **Don'ts** | WEAT | Bias |
|---|---|---|---|---|---|
| smile | 0.116 | 0.348 | rot | -0.099 | -1.118 |
| sightsee | 0.090 | 0.281 | negative | -0.101 | -0.763 |
| cheer | 0.094 | 0.277 | harm | -0.110 | -0.730 |
| celebrate | 0.114 | 0.264 | damage | -0.105 | -0.664 |
| picnic | 0.093 | 0.260 | slander | -0.108 | -0.600 |
| snuggle | 0.108 | 0.238 | slur | -0.109 | -0.569 |




**But lucky they also "capture" our moral choices**

[Jentzsch, Schramowski, Rothkopf, Kersting AIES 2019]

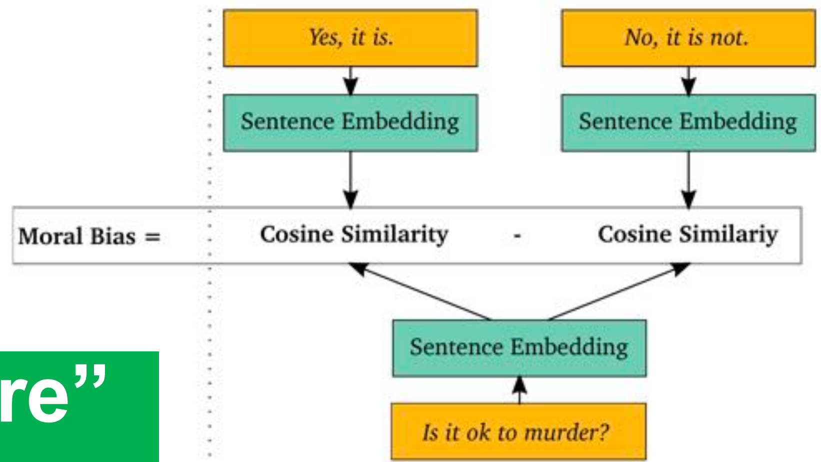
AAAI / ACM conference on
ARTIFICIAL INTELLIGENCE, ETHICS, AND SOCIETY




Video 05:10 Min.
Der Hamster gehört nicht in den Toaster – Wie Forscher von der TU Darmstadt versuchen, Maschinen ... [Videoseite]
hauptsache kultur | 14.03.19, 22:45 Uhr

# The Moral Choice Machine

| Dos | WEAT | Bias | Don'ts | WEAT | Bias |
|---|---|---|---|---|---|
| smile | 0.116 | 0.348 | rot | -0.099 | -1.118 |
| sightsee | 0.090 | 0.281 | negative | -0.101 | -0.763 |
| cheer | 0.094 | 0.277 | harm | -0.110 | -0.730 |
| celebrate | 0.114 | 0.264 | damage | -0.105 | -0.664 |
| picnic | 0.093 | 0.260 | slander | -0.108 | -0.600 |
| snuggle | 0.108 | 0.238 | slur | -0.109 | -0.569 |




**But lucky they also "capture" our moral choices**

[Jentzsch, Schramowski, Rothkopf, Kersting AIES 2019]

AAAI / ACM conference on
ARTIFICIAL INTELLIGENCE, ETHICS, AND SOCIETY

# Deep neural networks do not quantify their uncertainty
# They are not calibrated probabilistic models

**MNIST**

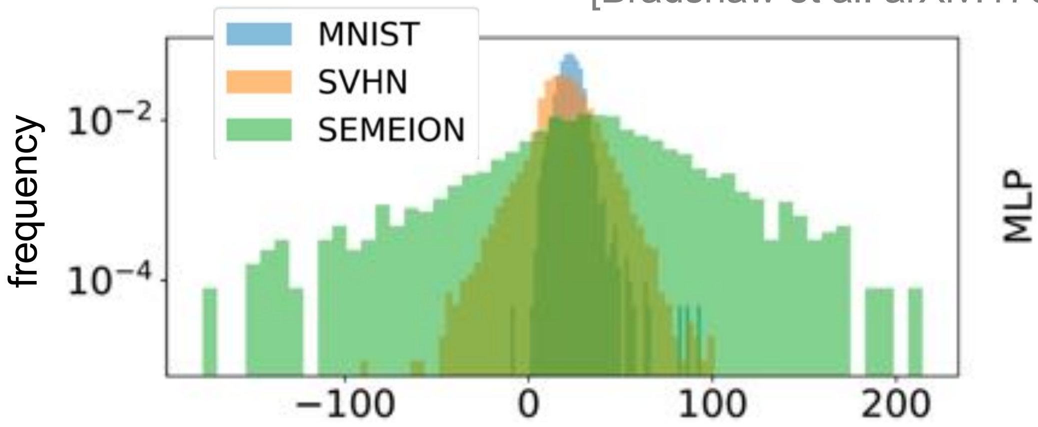
**SVHN**

**SEMEION**

**Train & Evaluate**

**Transfer Testing**

[Bradshaw et al. arXiv:1707.02476 2017]

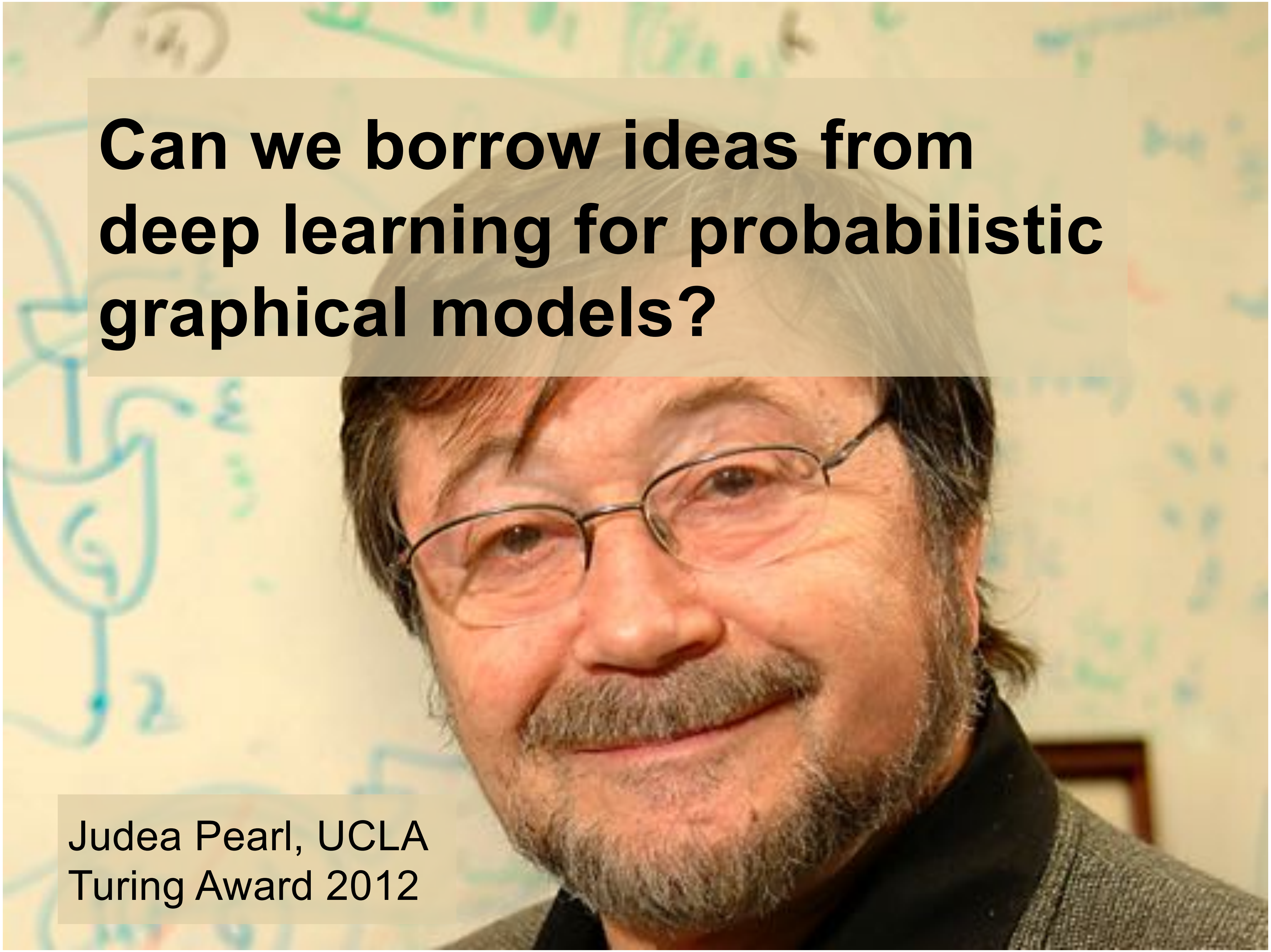



Input log „likelihood“ (sum over outputs)

[Peharz, Vergari, Molina, Stelzner, Trapp, Kersting, Ghahramani UDL@UAI 2018]

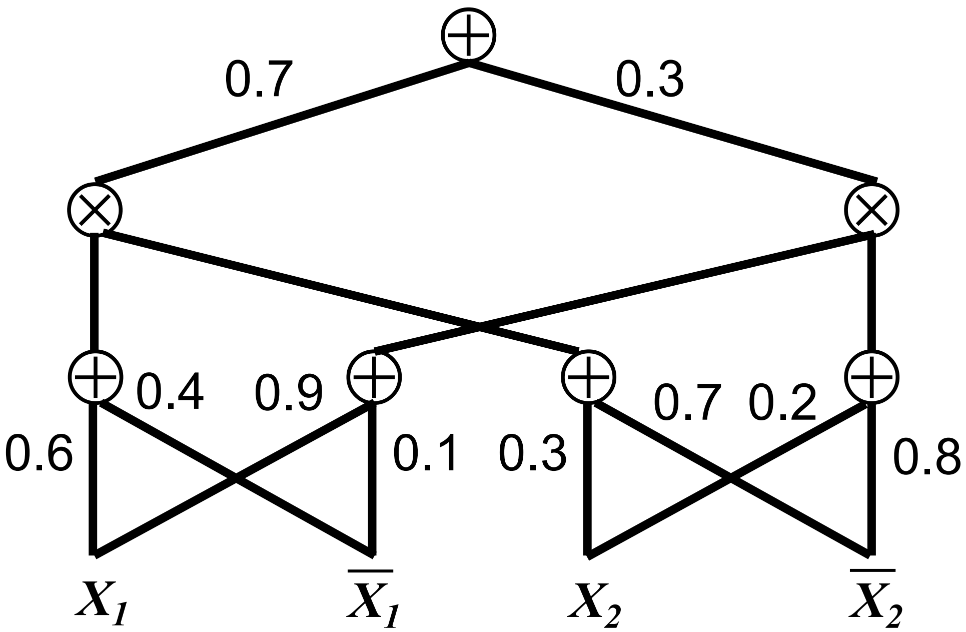
**Getting deep systems that know when they don't know.**

# Can we borrow ideas from deep learning for probabilistic graphical models?



Judea Pearl, UCLA
Turing Award 2012

# This results in Sum-Product Networks, a deep probabilistic learning framework

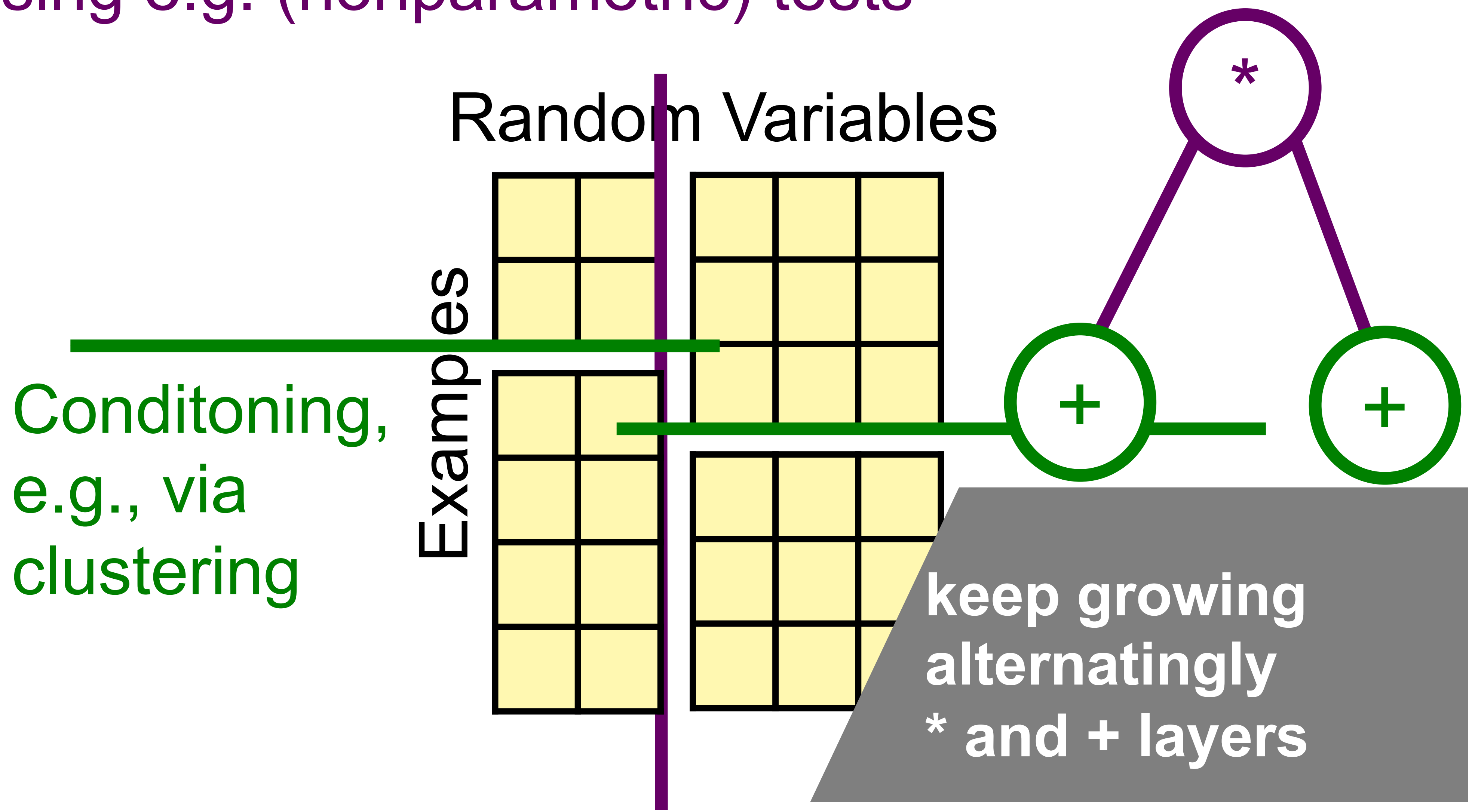




Computational graph (kind of TensorFlow graphs) that encodes how to compute probabilities

**Inference is linear in size of network**

# And there is a way to select models

Testing independence of random variables using e.g. (nonparametric) tests



[Poon, Domingos UAI'11; Molina, Natarajan, Kersting AAAI'17; Vergari, Peharz, Di Mauro, Molina, Kersting, Esposito AAAI '18; Molina, Vergari, Di Mauro, Esposito, Natarajan, Kersting AAAI '18]

# SPFlow: An Easy and Extensible Library for Sum-Product Networks

[Molina, Vergari, Stelzner, Peharz, Subramani, Poupart, Di Mauro, Kersting 2019]

TECHNISCHE UNIVERSITÄT DARMSTADT · UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO · UNIVERSITY OF WATERLOO · CAML · MADESI · Max Planck Institute for Intelligent Systems · UNIVERSITY OF CAMBRIDGE · VECTOR INSTITUTE · DFG · Federal Ministry of Education and Research

195 commits · 2 branches · 0 releases · 6 contrib...

Branch: master · New pull request · Create new file · Upload files · Find file · Clone or download

**https://github.com/SPFlow/SPFlow**

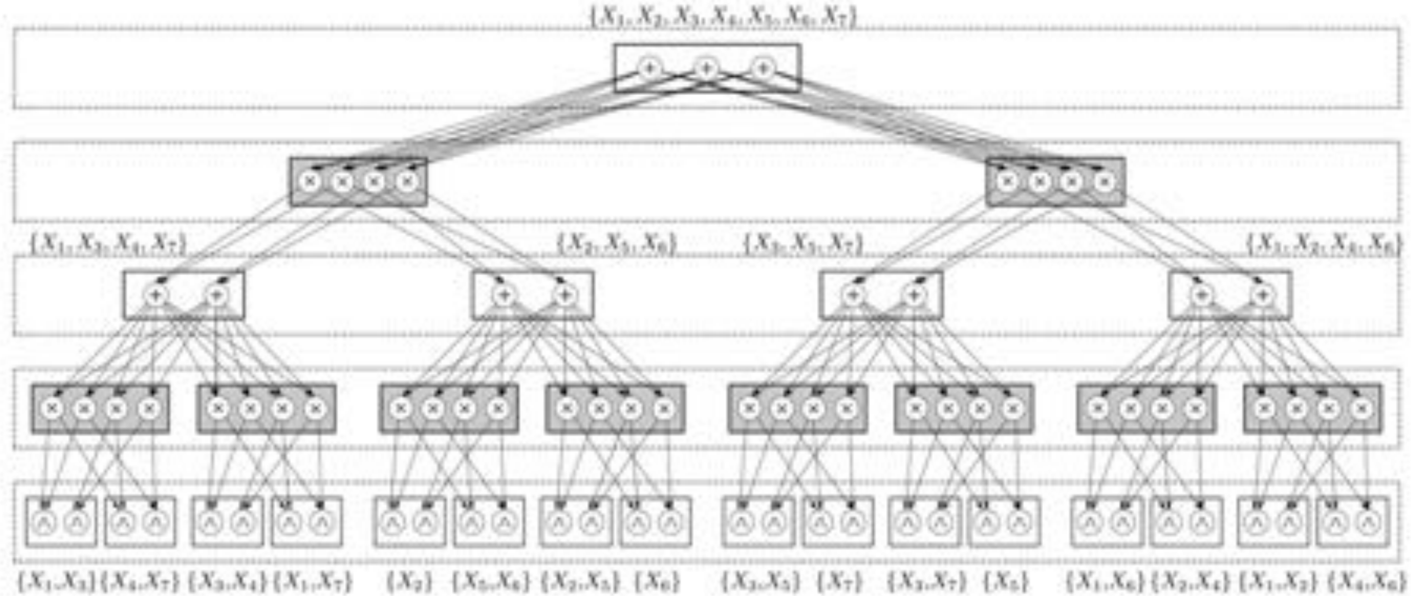
```
from spn.structure.leaves.parametric.Parametric import Categorical

from spn.structure.Base import Sum, Product

from spn.structure.base import assign_ids, rebuild_scopes_bottom_up


p0 = Product(children=[Categorical(p=[0.3, 0.7], scope=1), Categorical(p=[0.4, 0.6], scope=2)])
p1 = Product(children=[Categorical(p=[0.5, 0.5], scope=1), Categorical(p=[0.6, 0.4], scope=2)])
s1 = Sum(weights=[0.3, 0.7], children=[p0, p1])
p2 = Product(children=[Categorical(p=[0.2, 0.8], scope=0), s1])
p3 = Product(children=[Categorical(p=[0.2, 0.8], scope=0), Categorical(p=[0.3, 0.7], scope=1)])
p4 = Product(children=[p3, Categorical(p=[0.4, 0.6], scope=2)])
spn = Sum(weights=[0.4, 0.6], children=[p2, p4])

assign_ids(spn)
rebuild_scopes_bottom_up(spn)

return spn
```

**Domain Specific Language, Inference, EM, and Model Selection as well as Compilation of SPNs into TF and PyTorch and also into flat, library-free code even suitable for running on devices: C/C++,GPU, FPGA**

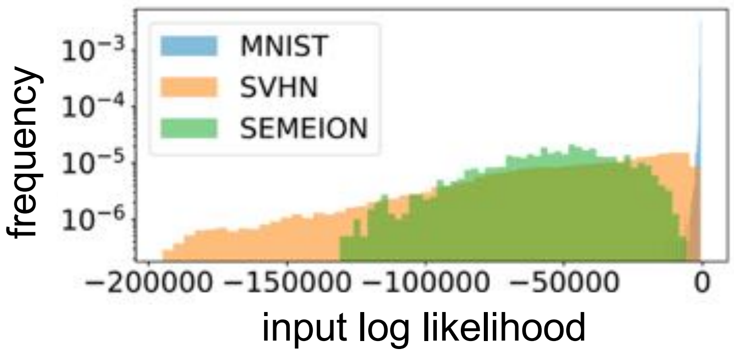
SPFlow, an open-source Python library providing a simple interface to inference, learning and manipulation routines for deep and tractable probabilistic models called Sum-Product Networks (SPNs). The library allows one to quickly create SPNs both from data and through a domain specific language (DSL). It efficiently implements several probabilistic inference

# Random sum-product networks

[Peharz, Vergari, Molina, Stelzner, Trapp, Kersting, Ghahramani UDL@UAI 2018]



| | | RAT-SPN | MLP | vMLP |
|---|---|---|---|---|
| Accuracy | MNIST | 98.19 (8.5M) | 98.32 (2.64M) | 98.09 (5.28M) |
| | F-MNIST | 89.52 (0.65M) | 90.81 (9.28M) | 89.81 (1.07M) |
| | 20-NG | 47.8 (0.37M) | 49.05 (0.31M) | 48.81 (0.16M) |
| Cross-Entropy | MNIST | 0.0852 (17M) | 0.0874 (0.82M) | 0.0974 (0.22M) |
| | F-MNIST | 0.3525 (0.65M) | 0.2965 (0.82M) | 0.325 (0.29M) |
| | 20-NG | 1.6954 (1.63M) | 1.6180 (0.22M) | 1.6263 (0.22M) |




outliers
prototypes
outliers
prototypes

# Learning the Structure of Autoregressive Deep Models such as PixelCNNs [van den Oord et al. NIPS 2016]


**Learn Conditional SPN by testing conditional independence and using conditional clustering, using e.g.** [Zhang et al. UAI 2011; Lee, Honovar UAI 2017; He et al. ICDM 2017; Zhang et al. AAAI 2018; Runge AISTATS 2018]

# Conditional SPNs

[Shao, Molina, Vergari, Peharz, Kersting 2019]

CAML

DFG

# Functional weights realized as neural network

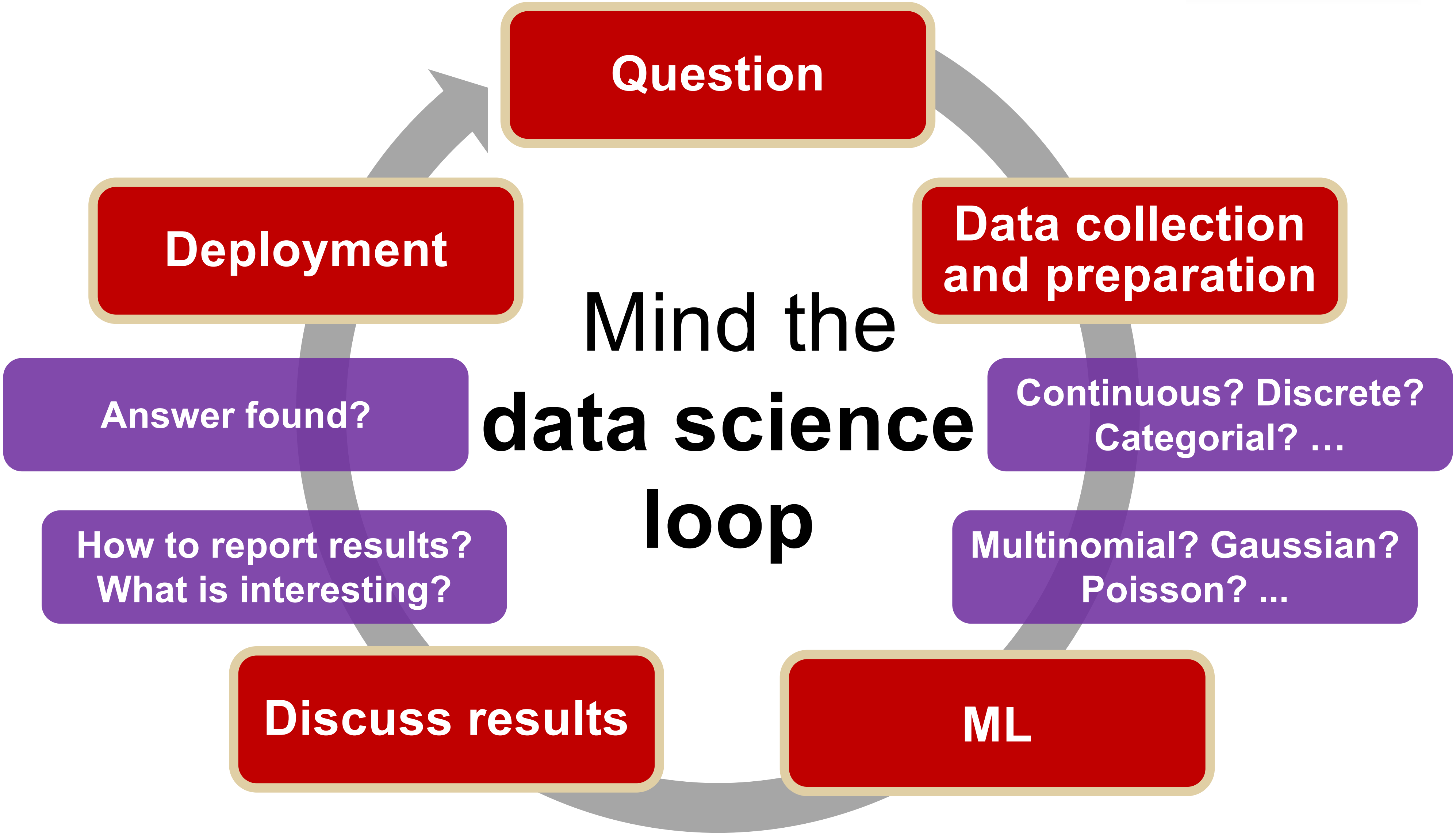
**Learn Conditional SPN by testing conditional independence and using conditional clustering, using e.g.** [Zhang et al. UAI 2011; Lee, Honovar UAI 2017; He et al. ICDM 2017; Zhang et al. AAAI 2018; Runge AISTATS 2018]

# Conditional SPNs

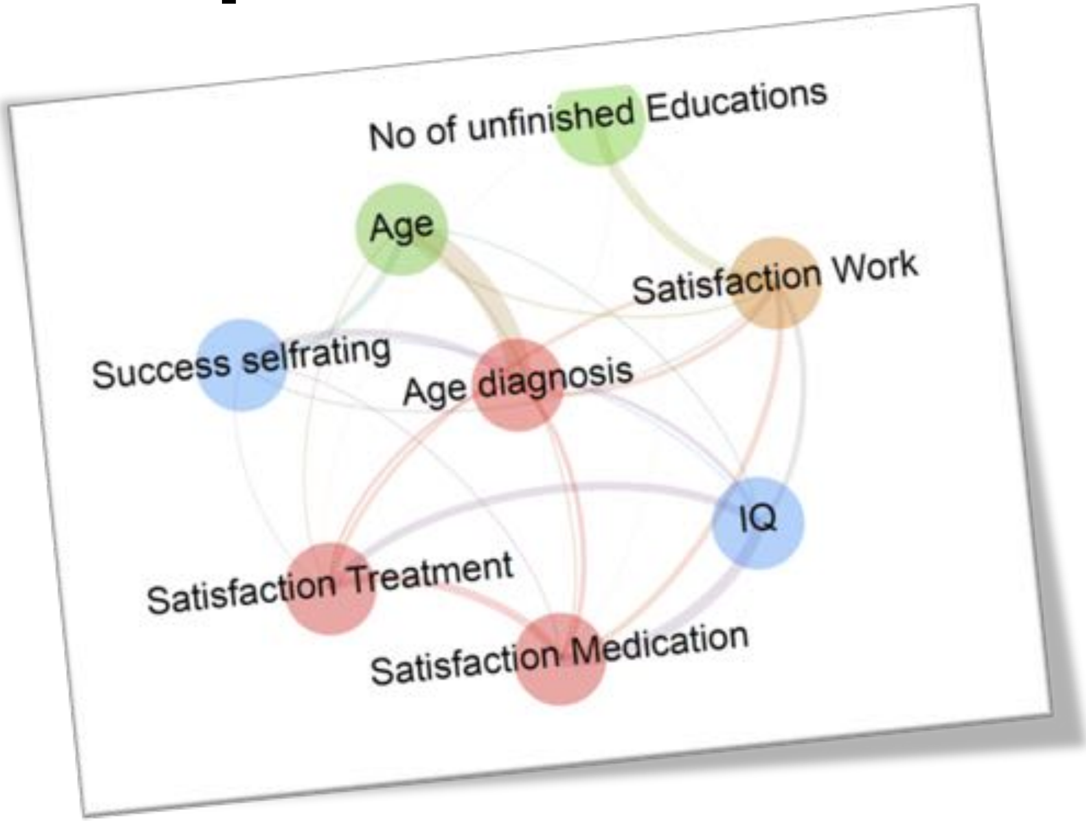
[Shao, Molina, Vergari, Peharz, Kersting 2019]

CAML

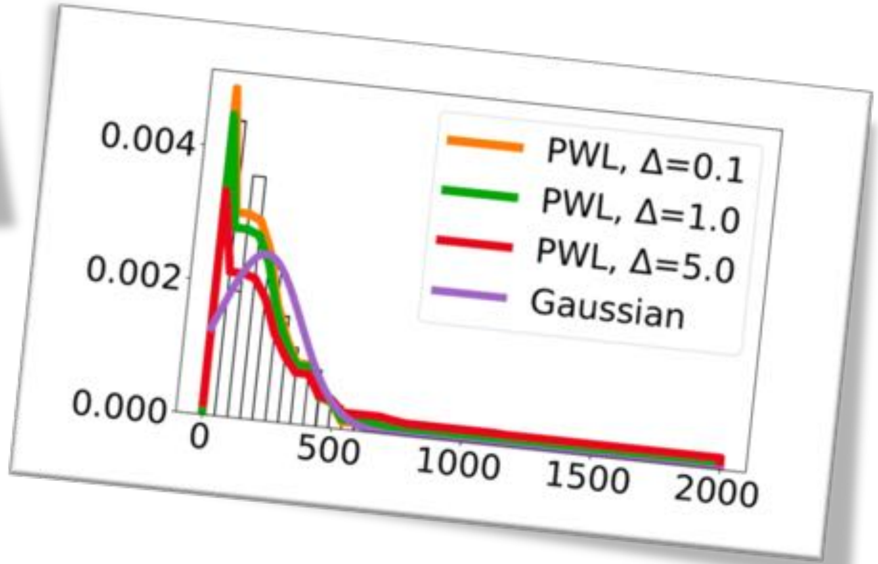
DFG


Question
Data collection and preparation
Continuous? Discrete? Categorial? …
Multinomial? Gaussian? Poisson? …
ML
Discuss results
How to report results? What is interesting?
Answer found?
Deployment
Mind the data science loop

# Distribution-agnostic Deep Probabilistic Learning




**Use nonparametric independency tests and piece-wise linear approximations**



# Distribution-agnostic Deep Probabilistic Learning


**Use nonparametric independency tests and piece-wise linear approximations**

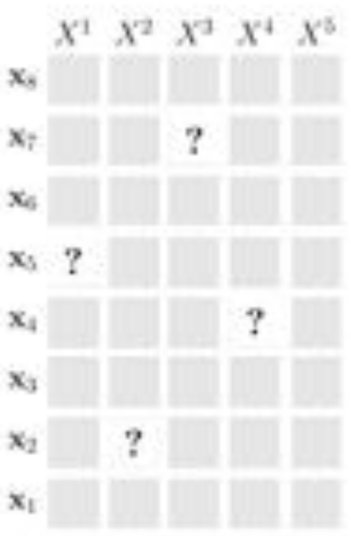

However, we have to provide the statistical types and do not gain insights into the parametric forms of the variables. **Are they Gaussians? Gammas? …**

[Vergari, Molina, Peharz, Ghahramani, Kersting, Valera AAAI 2019]

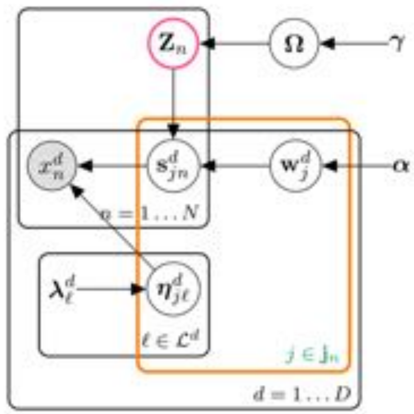
# The Explorative Automatic Statistician

UBER AI Labs · UNIVERSITY OF CAMBRIDGE · Max Planck Institute for Intelligent Systems · TECHNISCHE UNIVERSITÄT DARMSTADT

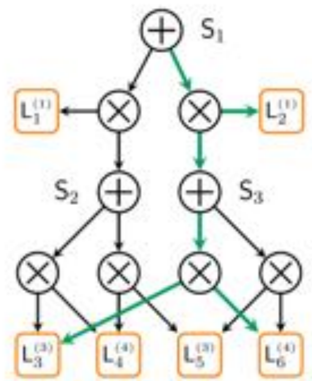
missing value



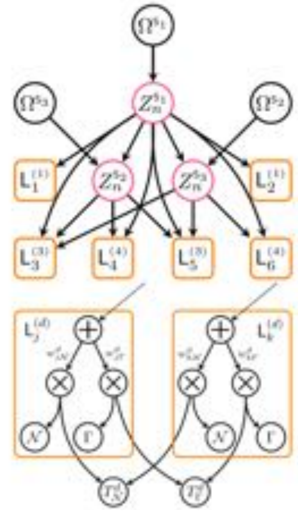

**We can even automatically discovers the statistical types and parametric forms of the variables**




Bayesian Type Discovery

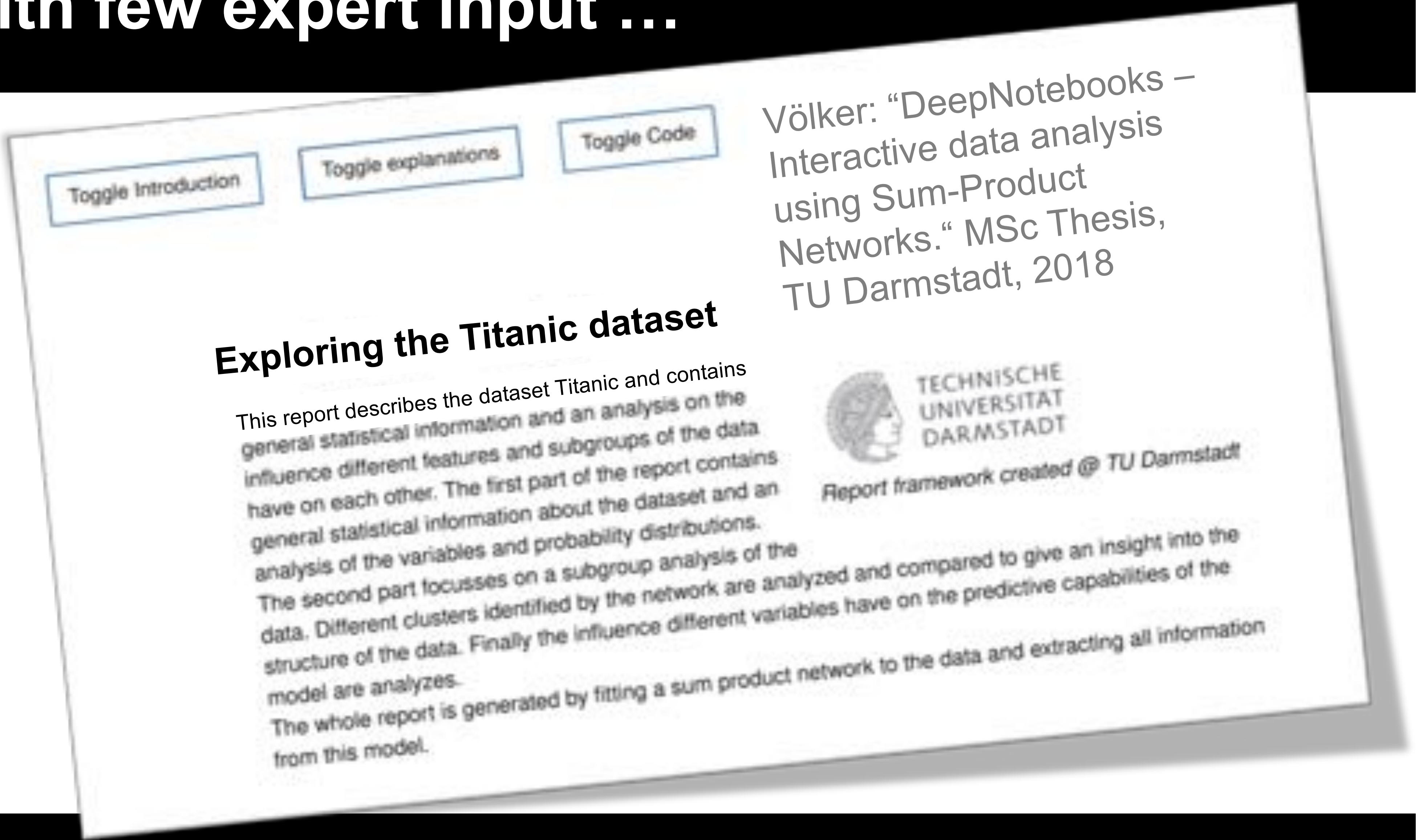



Mixed Sum-Product Network



Automatic Statistician

# That is, the machine understands the data with few expert input …




**Exploring the Titanic dataset**

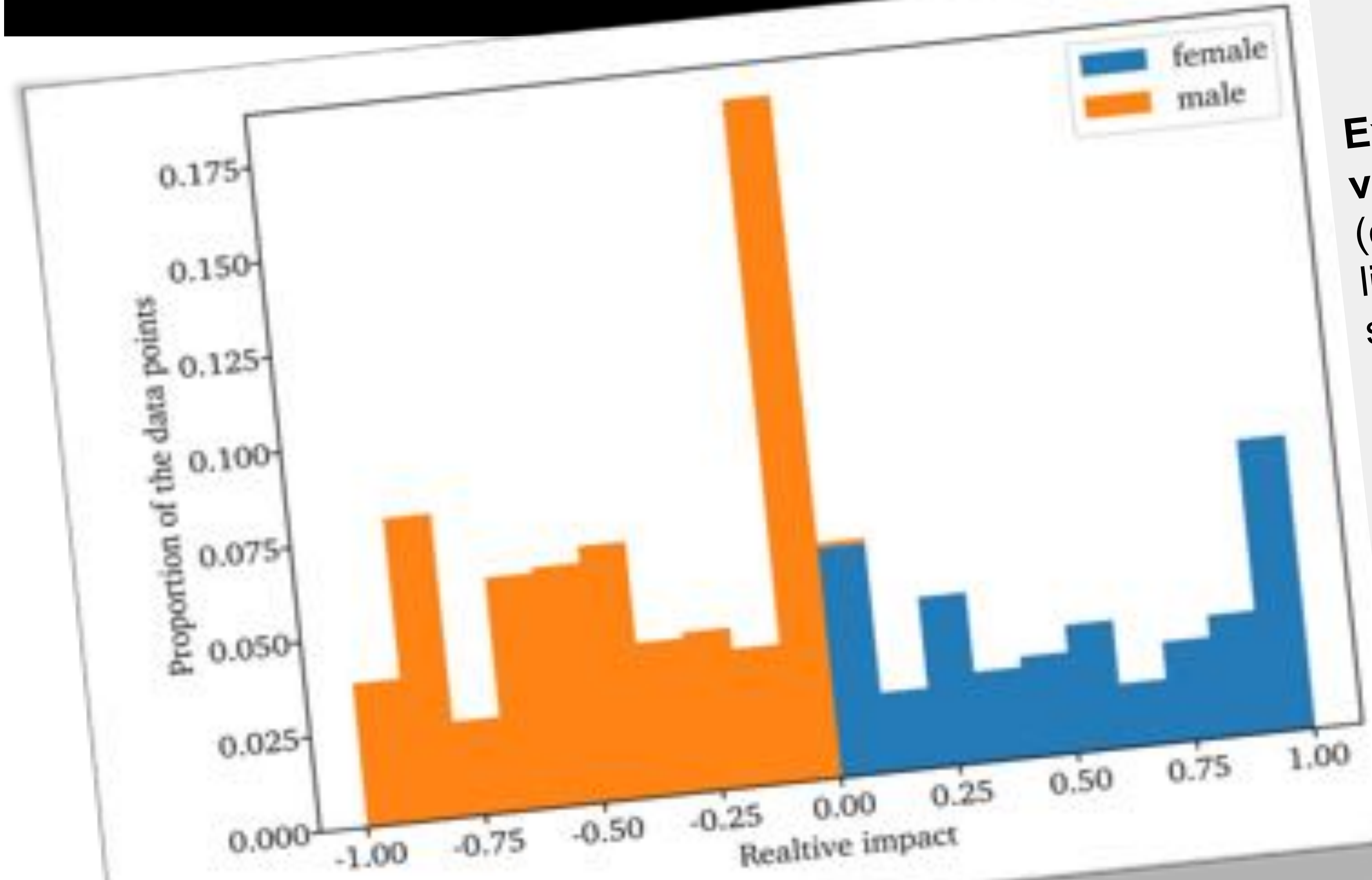
This report describes the dataset Titanic and contains general statistical information and an analysis on the influence different features and subgroups of the data have on each other. The first part of the report contains general statistical information about the dataset and an analysis of the variables and probability distributions. The second part focusses on a subgroup analysis of the data. Different clusters identified by the network are analyzed and compared to give an insight into the structure of the data. Finally the influence different variables have on the predictive capabilities of the model are analyzes.
The whole report is generated by fitting a sum product network to the data and extracting all information from this model.

Völker: "DeepNotebooks – Interactive data analysis using Sum-Product Networks." MSc Thesis, TU Darmstadt, 2018

# …and can compile data reports automatically

*[Baehrens, Schroeter, Harmeling, Kawanabe, Hansen, Müller JMLR 11:1803-1831, 2010]

# The machine understands the data with no expert input …




**Explanation vector*** (computable in linear time in the sizre of the SPN) **showing the impact of "gender" on the chances of survival for the Titanic dataset**

# …and can compile data reports automatically

P( heart attack |  )?

The New York Times

Opinion

A.I. Is Harder Than You Think

and Data Science

By Gary Marcus and Ernest Davis

Mr. Marcus is a professor of psychology and neural science. Mr. Davis is a professor of computer science.

May 18, 2018

P( heart attack |  )?

The New York Times

Opinion

A.I. Is Harder Than You Think

and Data Science

By Gary Marcus and Ernest Davis

Mr. Marcus is a professor of psychology and neural science. Mr. Davis is a professor of computer science.

May 18, 2018

P( heart attack |  )?

The New York Times

Opinion

# A.I. Is Harder Than You Think

and Data Science

By Gary Marcus and Ernest Davis

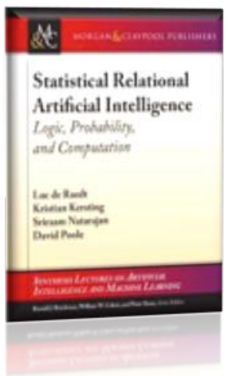
Mr. Marcus is a professor of psychology and neural science. Mr. Davis is a professor of computer science.

May 18, 2018

P( heart attack |  )?

# Crossover of ML and DS with data & programming abstractions

De Raedt, Kersting, Natarajan, Poole: Statistical Relational Artificial Intelligence: Logic, Probability, and Computation. Morgan and Claypool Publishers, ISBN: 9781627058414, 2016.




KATHOLIEKE UNIVERSITEIT LEUVEN

THE UNIVERSITY OF TEXAS AT DALLAS

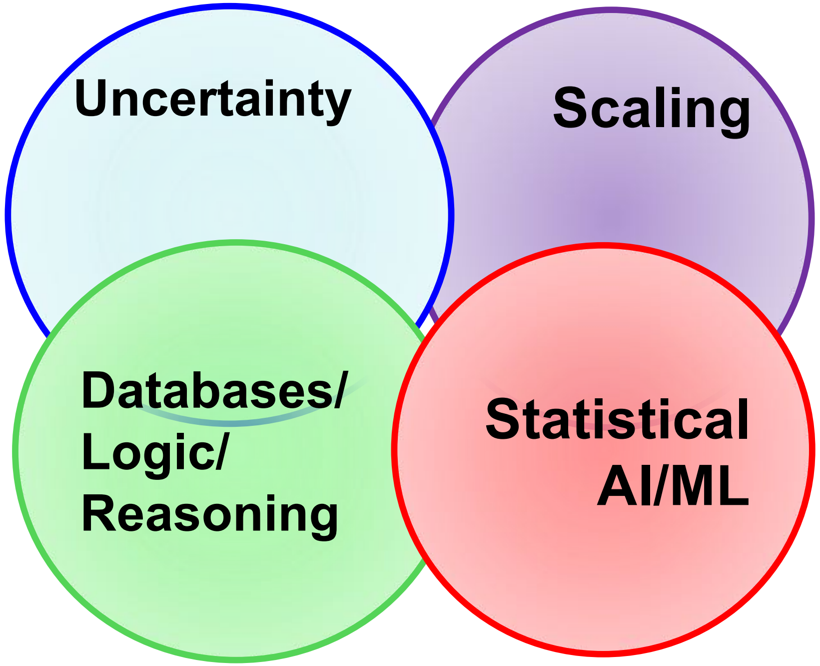
TECHNISCHE UNIVERSITÄT DARMSTADT

UBC THE UNIVERSITY OF BRITISH COLUMBIA

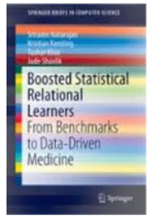
**building general-purpose data science and ML machines**

**make the ML/DS expert more effective**

**increases the number of people who can successfully build ML/DS applications**



Natarajan, Khot, Kersting, Shavlik. Boosted Statistical Relational Learners. Springer Brief 2015

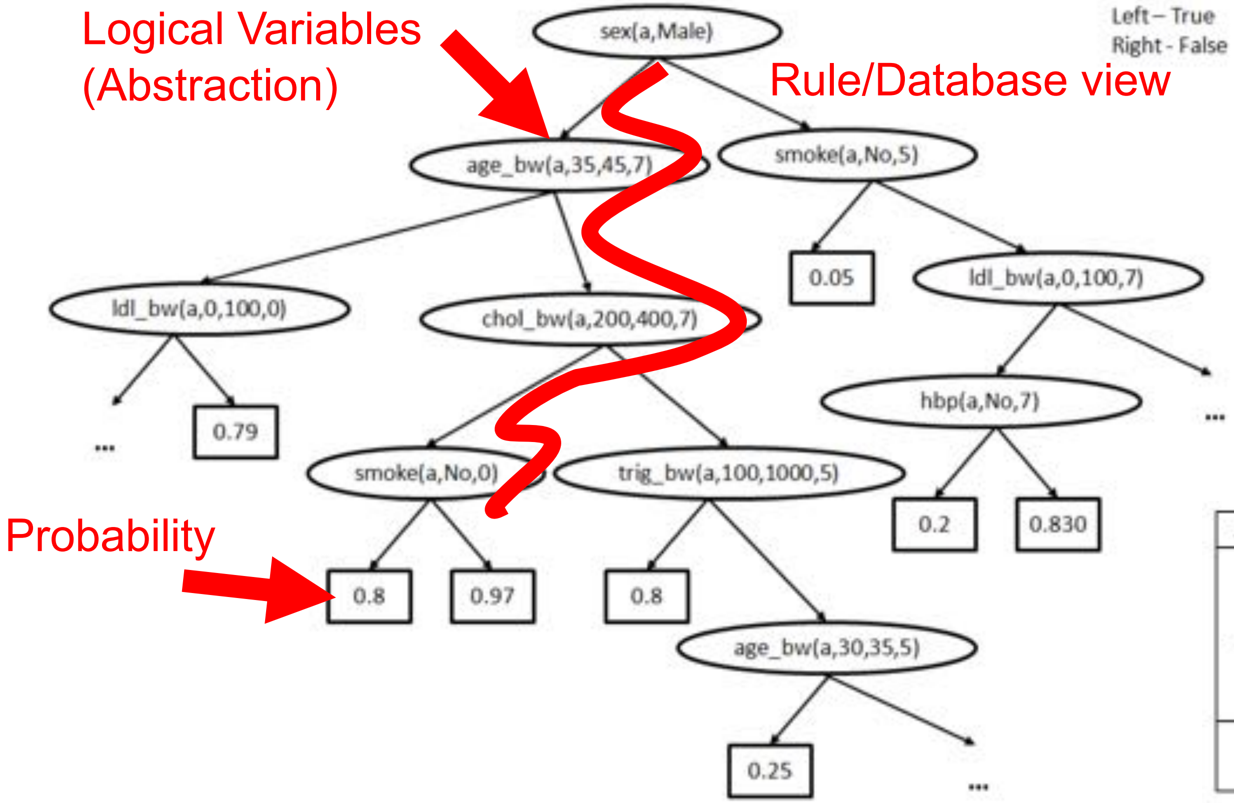


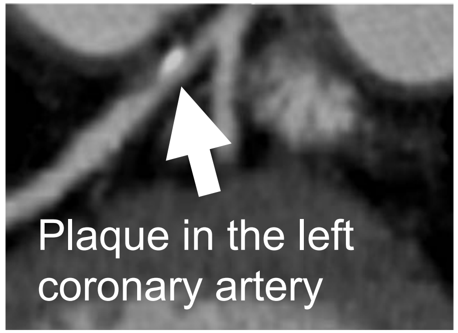

# Understanding Electronic Health Records

Atherosclerosis is the cause of the majority of Acute Myocardial Infarctions (heart attacks)

TECHNISCHE UNIVERSITÄT DARMSTADT

THE UNIVERSITY OF TEXAS AT DALLAS




[Circulation; 92(8), 2157-62, 1995; JACC; 43, 842-7, 2004]

| Algorithm | Accuracy | AUC-ROC |
|---|---|---|
| J48 | 0.667 | 0.607 |
| SVM | 0.667 | 0.5 |
| AdaBoost | 0.667 | 0.608 |
| Bagging | 0.677 | 0.613 |
| NB | 0.75 | 0.653 |
| RPT | 0.669* | 0.778 |
| RFGB | 0.667* | 0.819 |

The higher, the better

25%

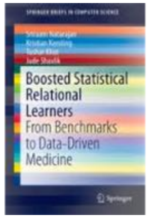
| Algorithm for Mining Markov Logic Networks | Likelihood The higher, the better | AUC-ROC The higher, the better | AUC-PR The higher, the better | Time The lower, the better |
|---|---|---|---|---|
| **Boosting** | 0.81 | 0.96 | 0.93 | **9s** |
| **LSM** | 0.73 | 0.54 | 0.62 | **93 hrs** |

11% 78% 50% 37200x faster

**state-of-the-art**

[Kersting, Driessens ICML´08; Karwath, Kersting, Landwehr ICDM´08; Natarajan, Joshi, Tadepelli, Kersting, Shavlik. IJCAI´11; Natarajan, Kersting, Ip, Jacobs, Carr IAAI `13; Yang, Kersting, Terry, Carr, Natarajan AIME ´15; Khot, Natarajan, Kersting, Shavlik ICDM´13, MLJ´12, MLJ´15, Yang, Kersting, Natarajan BIBM`17]

Natarajan, Khot, Kersting, Shavlik. Boosted Statistical Relational Learners. Springer Brief 2015




TECHNISCHE UNIVERSITÄT DARMSTADT

THE UNIVERSITY OF TEXAS AT DALLAS

**https://starling.utdallas.edu/software/boostsrl/wiki/**

StARLinG LAB — People | Publications | Projects | Software | Datasets | Blog

**BOOSTSRL BASICS**

- Getting Started
- File Structure
- Basic Parameters
- Advanced Parameters
- Basic Modes
- Advanced Modes

**ADVANCED BOOSTSRL**

- Default (RDN-Boost)
- MLN-Boost
- Regression
- One-Class Classification
- Cost-Sensitive SRL
- Learning with Advice
- Approximate Counting
- Discretization of Continuous-Valued Attributes
- Lifted Relational Random Walks
- Grounded Relational Random Walks

**APPLICATIONS**

- Natural Language Processing

# BoostSRL Wiki

**BoostSRL** (Boosting for Statistical Relational Learning) is a gradient-boosting based approach to learning different types of SRL models. As with the standard gradient-boosting approach, our approach turns the model learning problem to learning a sequence of regression models. The key difference to the standard approaches is that we learn relational regression models i.e., regression models that operate on relational data. We assume the data in a predicate logic format and the output are essentially first-order regression trees where the inner nodes contain conjunctions of logical predicates. For more details on the models and the algorithm, we refer to our book on this topic.

Sriraam Natarajan, Tushar Khot, Kristian Kersting and Jude Shavlik, Boosted Statistical Relational Learners: From Benchmarks to Data-Driven Medicine . SpringerBriefs in Computer Science, ISBN: 978-3-319-13643-1, 2015

**Human-in-the-loop learning**

# A simple example

Guy van den Broeck
UCLA

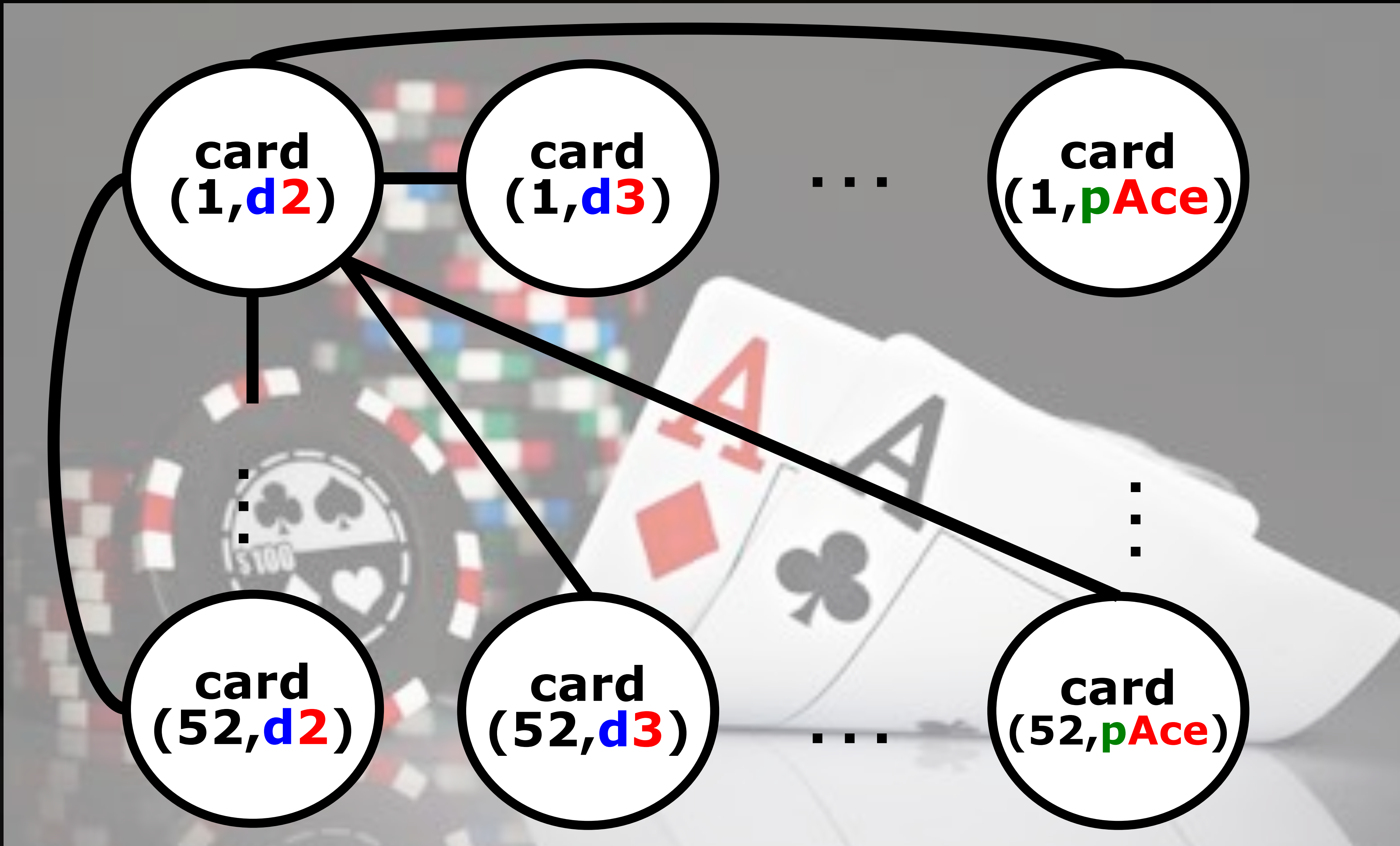
**What is the problem that the first card of a randomly shuffled deck with 52 cards is an Ace?**

**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!**

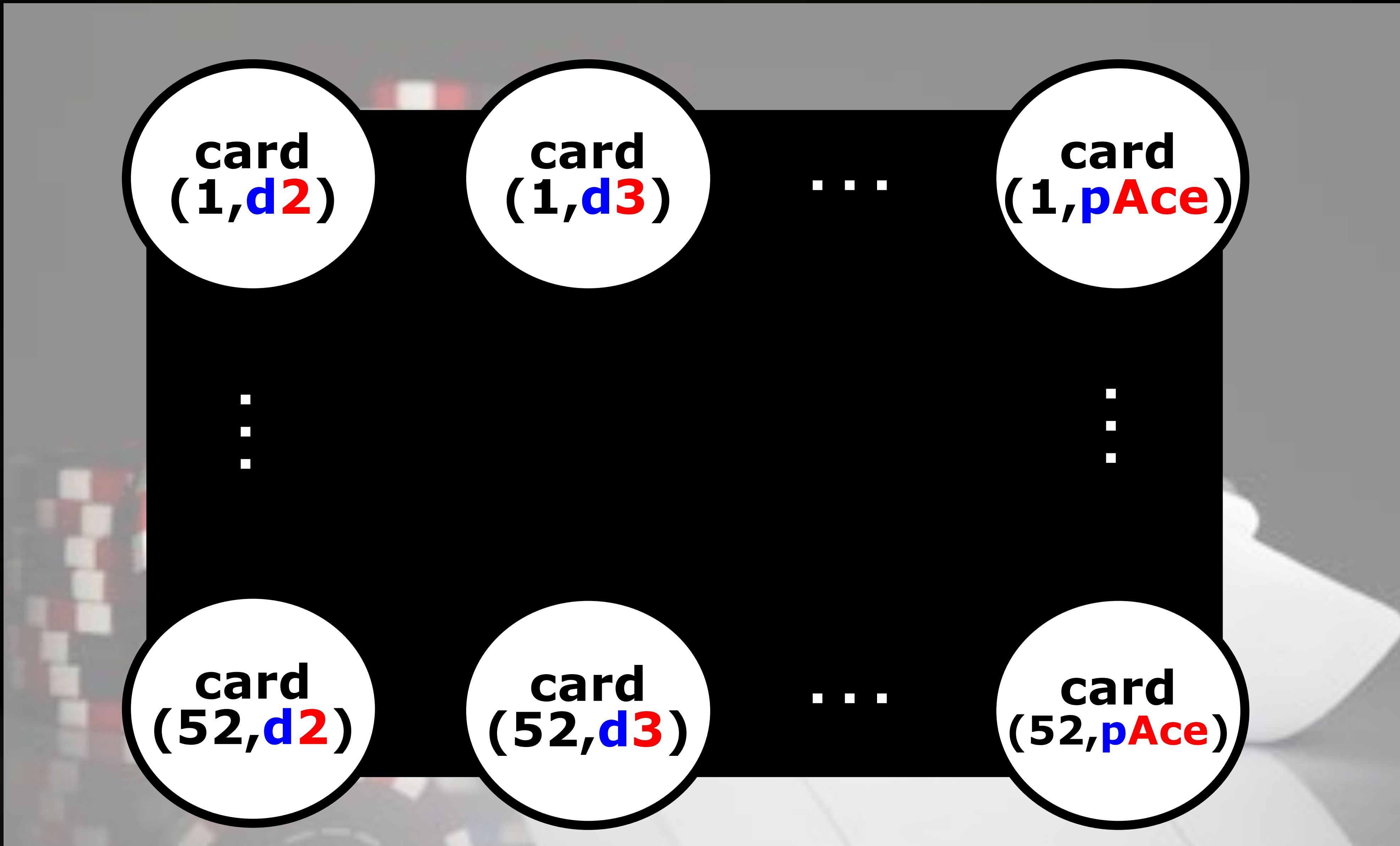
# A simple example

Guy van den Broeck
UCLA



# A simple example

Guy van den Broeck
UCLA




We do not want to write down all the rules!

## Faster modelling

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

**w1:** $\forall p,x,y$: card(P,X),card(P,Y) $\Rightarrow x=y$

**w2:** $\forall c,x,y$: card(X,C),card(Y,C) $\Rightarrow x=y$

**We do not want to write down all the rules!**

# A simple example

Guy van den Broeck
UCLA

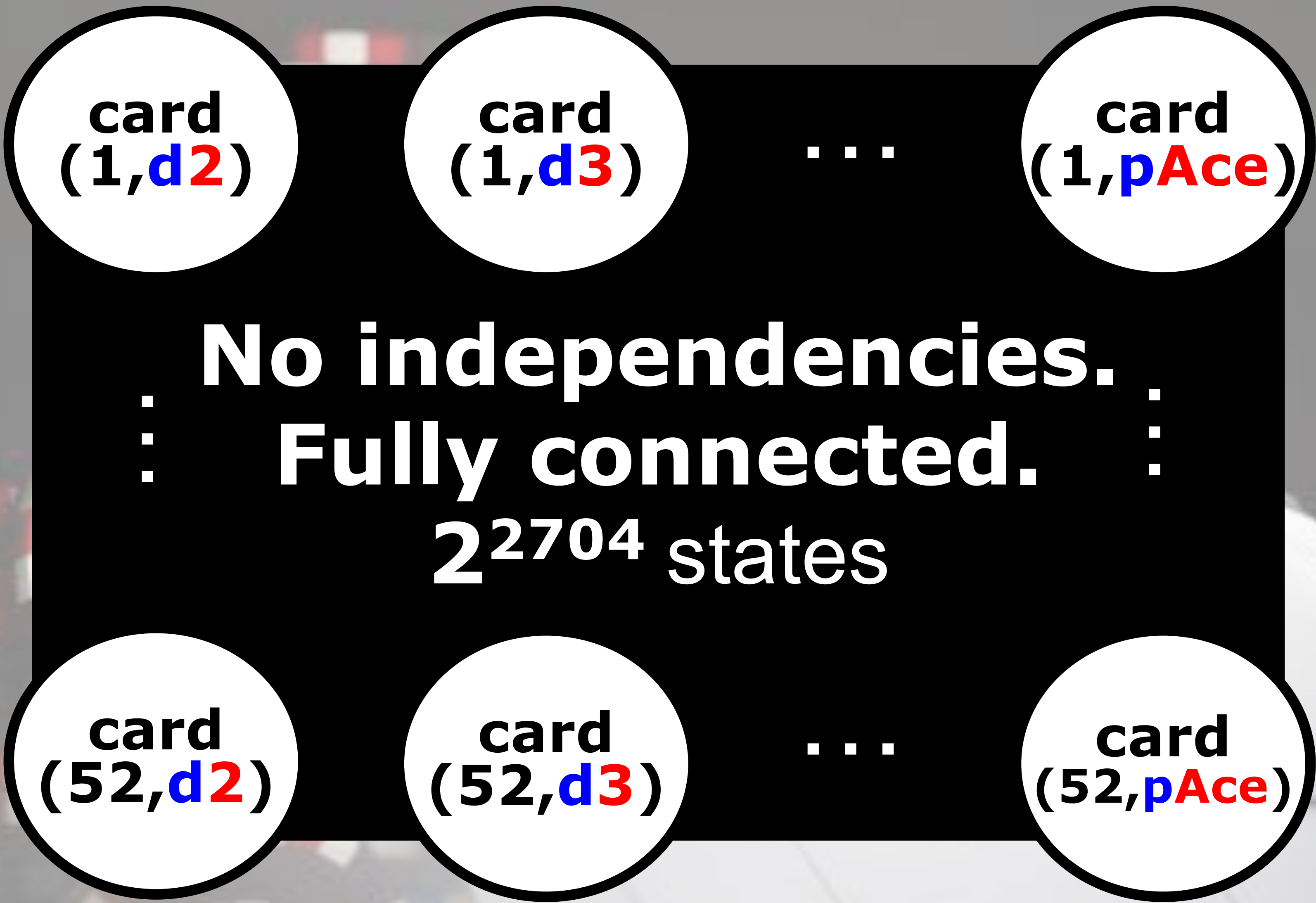
card(1,d2) card(1,d3) ... card(1,pAce)

⋮ What about inference? ⋮

card(52,d2) card(52,d3) ... card(52,pAce)

# A simple example

Guy van den Broeck
UCLA



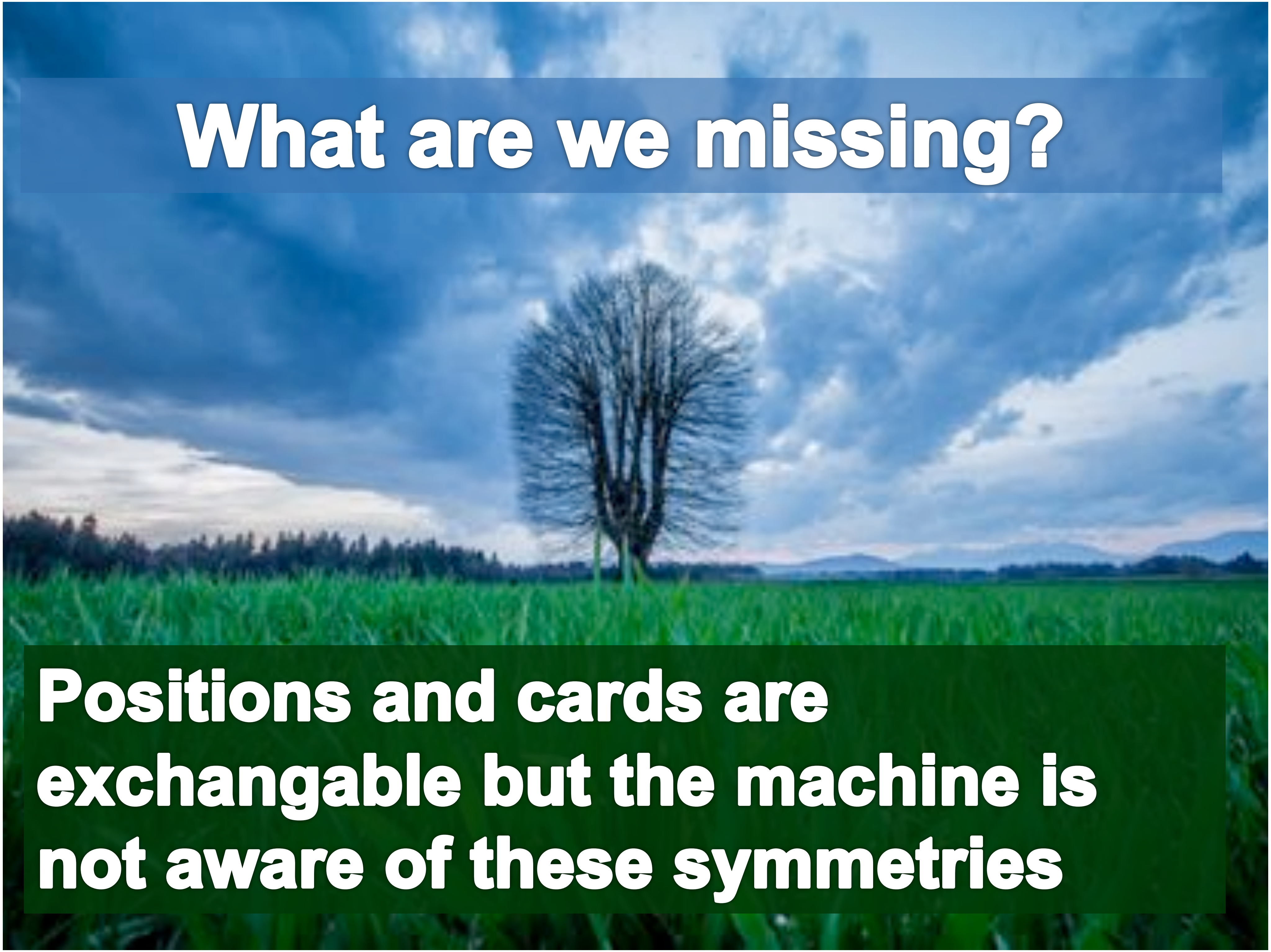
# A simple example

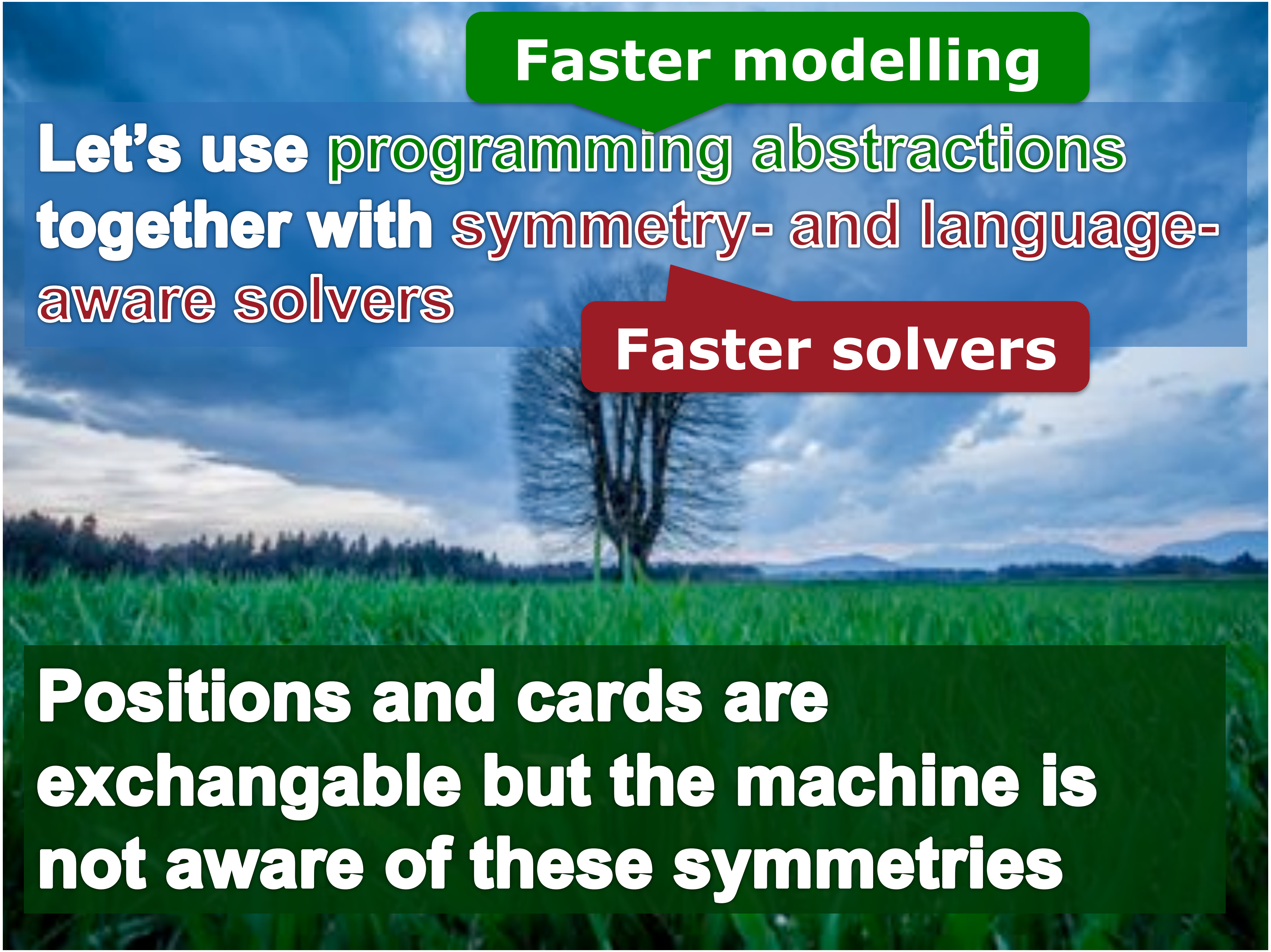
Guy van den Broeck
UCLA

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

: A machine will not solve the problem :

card(52,d2) card(52,d3) ... card(52,pAce)


What are we missing?
Positions and cards are exchangable but the machine is not aware of these symmetries



Faster modelling

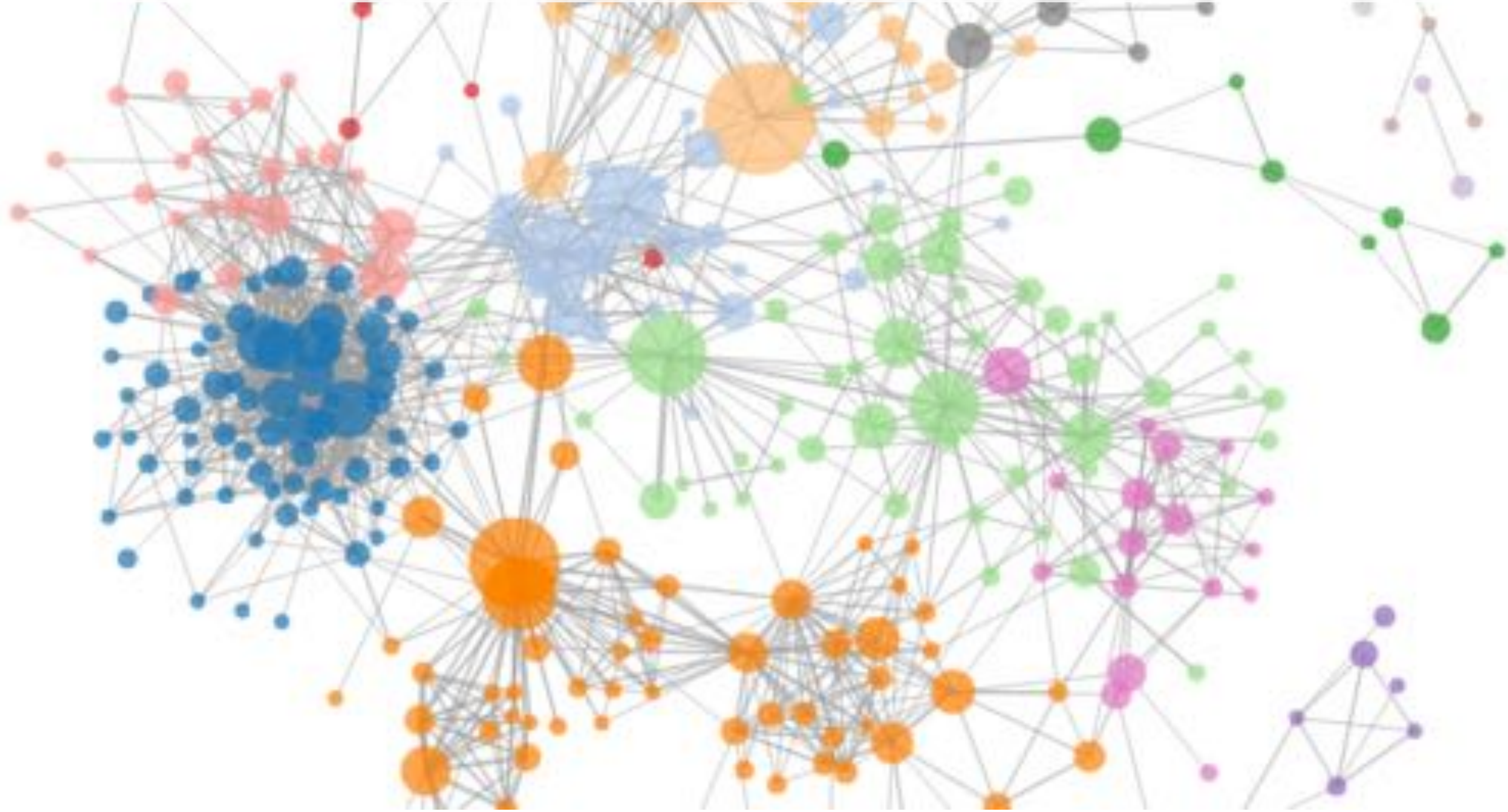
Let's use programming abstractions together with symmetry- and language-aware solvers

Faster solvers

**Positions and cards are exchangable but the machine is not aware of these symmetries**

# Let’s make it more “optimization”-like

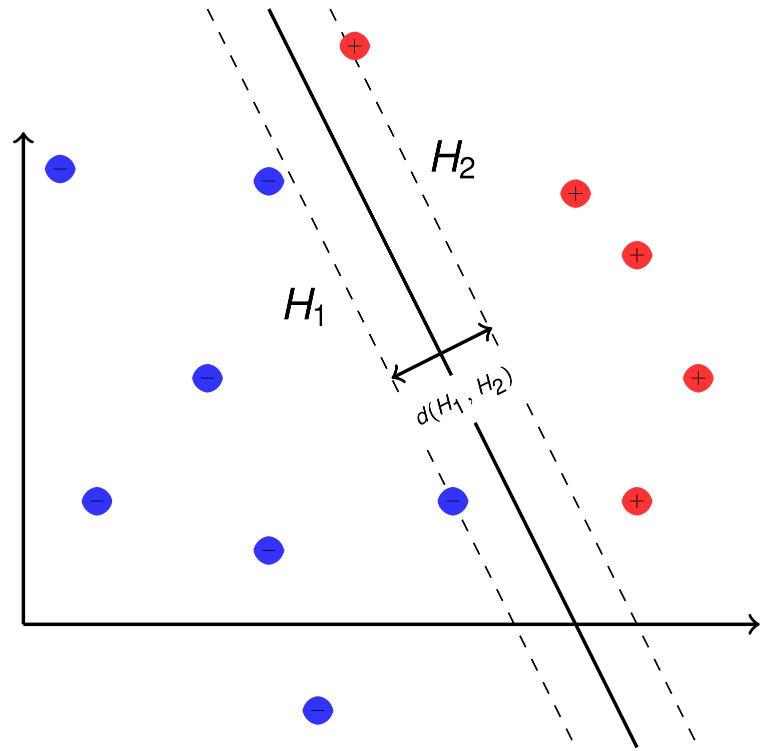
# Let’s say we want to classify publications into scientific disciplines



# Classification using LP SVMs

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

$$H^* = \left\{ \vec{x} \mid \left\langle \vec{x}, \vec{\beta} \right\rangle + \beta_0 = 0 \right\}$$




$$d(H_1, H_2) = \frac{2}{||\vec{\beta}||}$$

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]

**Lifted LP-SVM**

```
var pred/1;        #predicted label for unlabeled instances
var slack/1;       #the slacks
var coslack/2;       #sl
var weight/1;      #the
var b/0;           #the
var r/0;           #marg

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

#find      gest margin. here the C's encode trade-off parameters
minimize: -r + C(1) * slack + C(2) * coslack;
```

**Logically parameterized LP variable (set of ground LP variables)**

**Logically parameterized LP objective**

Write down the LP-SVM in „paper form“.
The machine compiles it into solver form.

reloop

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**

```
#examples should be on the correct side of the hyperplane
subject to forall {I in label(I)}:
    label(I)*(innerProd(I) + b) + slack(I) >= r;
#weights are between -1 and 1
subject to for
subject to : r
subject to forall {I in label(I)}: slack(I) >= 0;     #slacks are positive
```

**Logically parameterized LP constraint**

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

```
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);
                                   I2))} slack(I1,I2)
                                   I1))} slack(I1,I2)

                                   ode trade-off parameters
minimize: -r + C(I) * s             * coslack;

subject to forall {I in query        pred(I) = innerProd(I) + b;
#related instances should have th    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
```

**Logical query defines scope of abstract collective constraint**

**Citing papers share topics**

**Collective constraints**

**No kernel, the structure is expressed within the constraints!**

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]

**If exchanging two variables preserves optimality, group them together**

**Big Model**

Run Solver

automatically compressed

**Small Model**

Run Solver

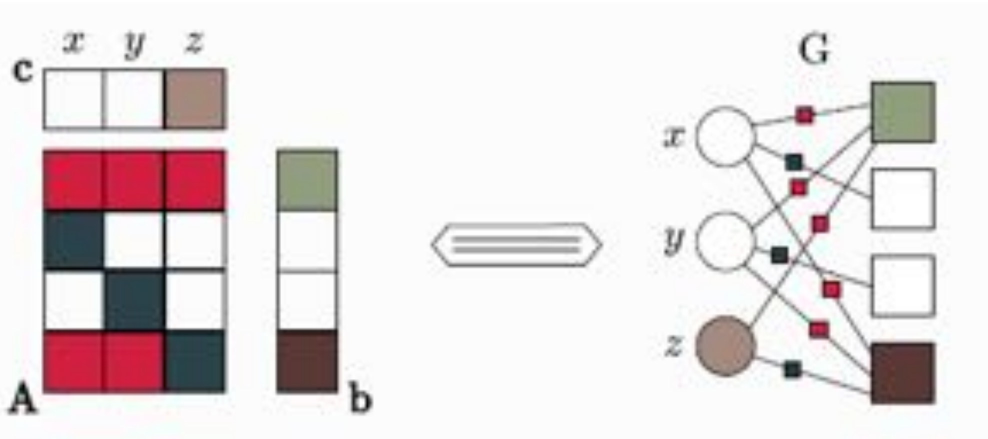
# Lifted Mathematical Programming

## Exploiting computational symmetries

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

$$\max_{[x,y,z]^T \in \mathbb{R}^3} \quad 0x + 0y + 1z$$

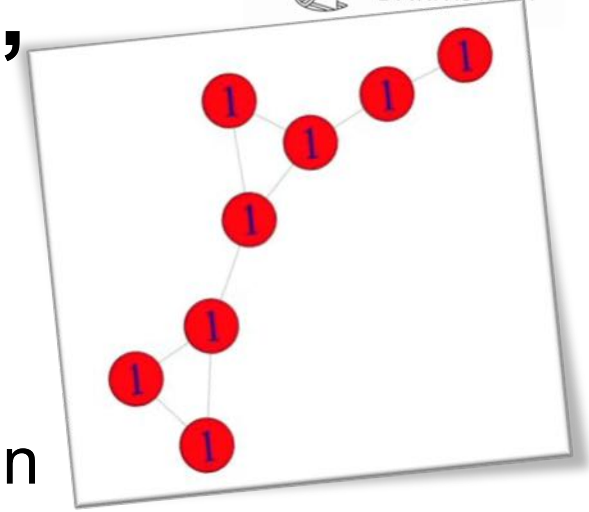
$$\text{s.t.} \quad \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$




**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]

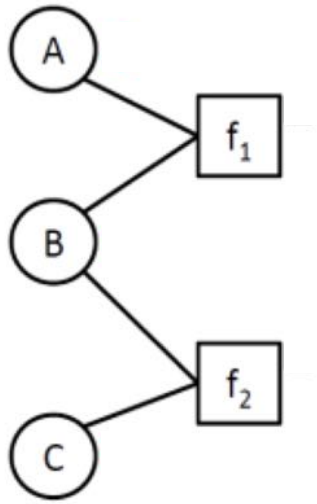
Has lead to highly performant graph kernels
[Shervashidze, Schweitzer, van Leeuwen, Mehlhorn, Borgwardt JMLR 12:2539-2561 ´11]

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]

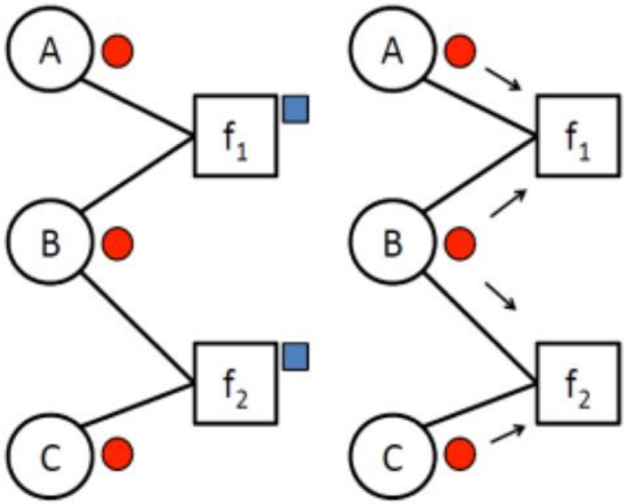



**Color nodes initially with the same color**, **say red**

**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

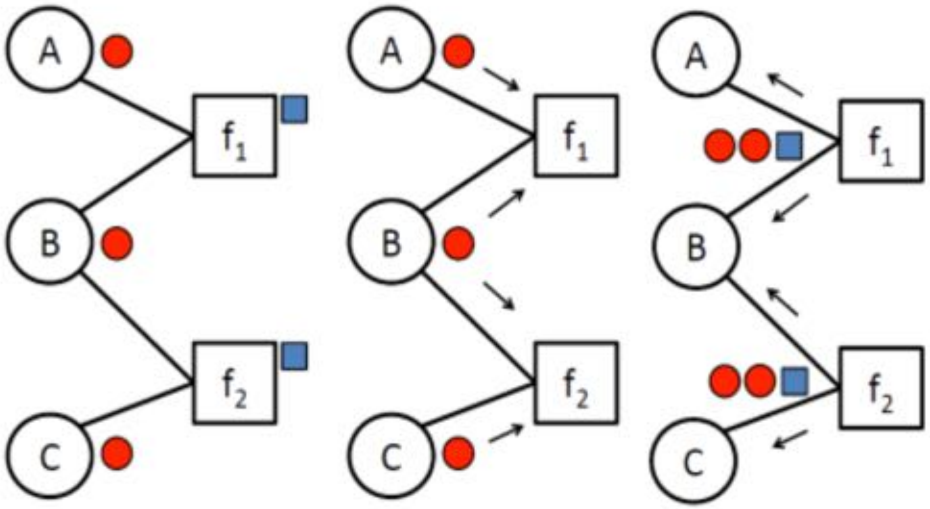
[Kersting, Ahmadi, Natarajan UAI'09; Ahmadi, Kersting, Mladenov, Natarajan MLJ'13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]




1. Each factor collects the colors of its neighboring nodes

# Compression: Pass colors around

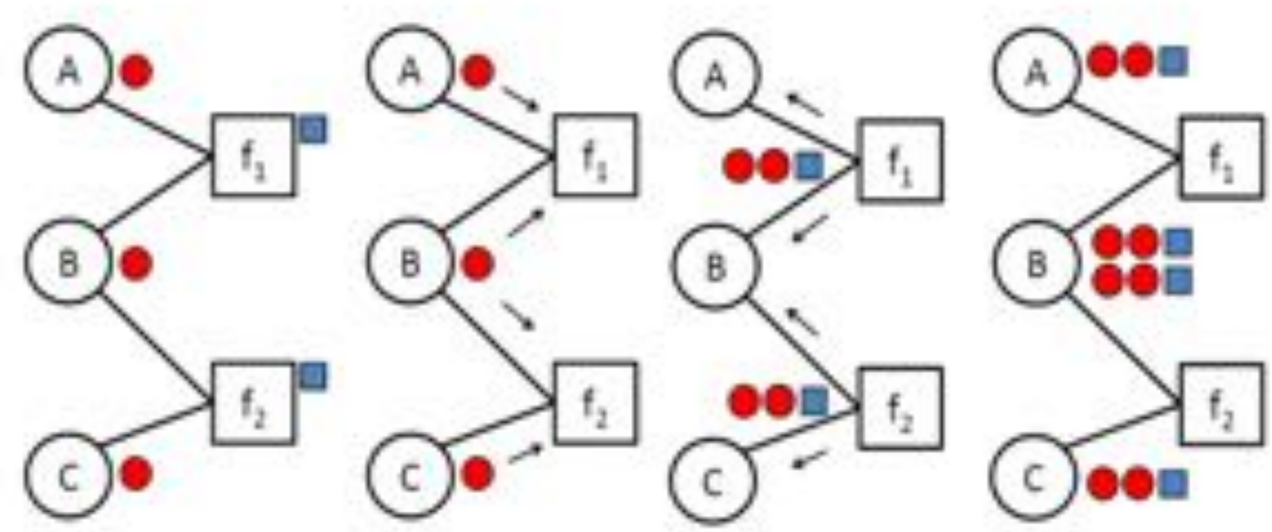
[Kersting, Ahmadi, Natarajan UAI'09; Ahmadi, Kersting, Mladenov, Natarajan MLJ'13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]




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

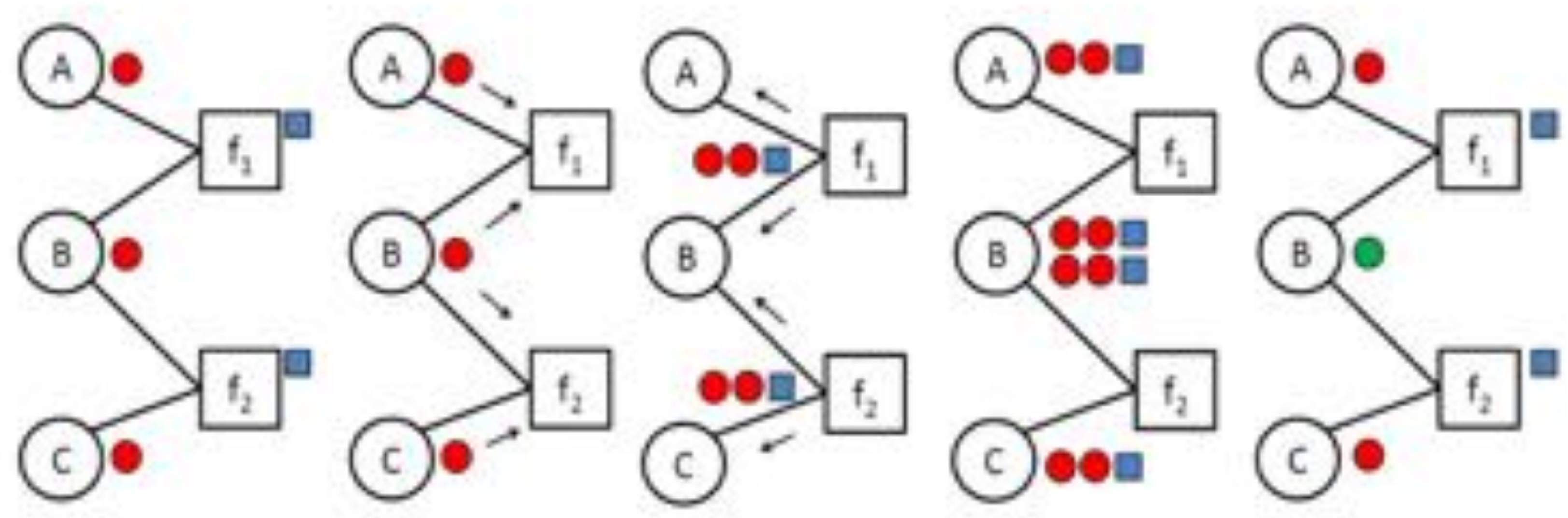
[Kersting, Ahmadi, Natarajan UAI'09; Ahmadi, Kersting, Mladenov, Natarajan MLJ'13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]




1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs" its color signature with its own color
3. Each node collects the signatures of its neighboring factors

# Compression: Pass colors around

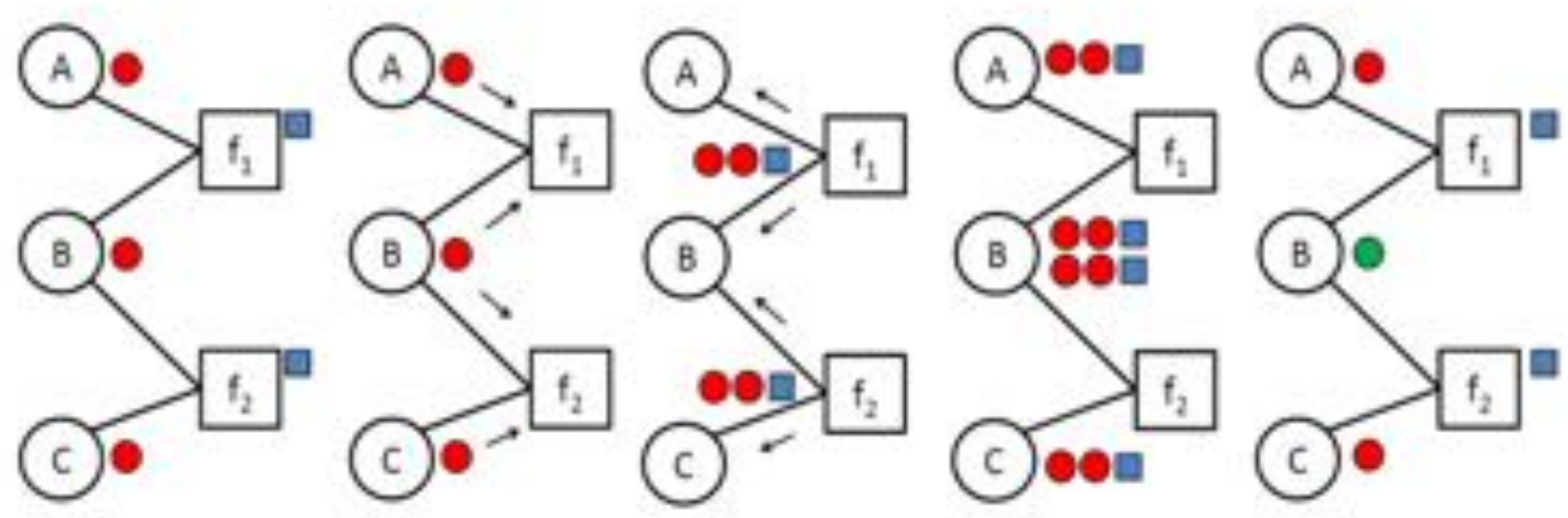
[Kersting, Ahmadi, Natarajan UAI'09; Ahmadi, Kersting, Mladenov, Natarajan MLJ'13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]




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

# Compression: Pass colors around

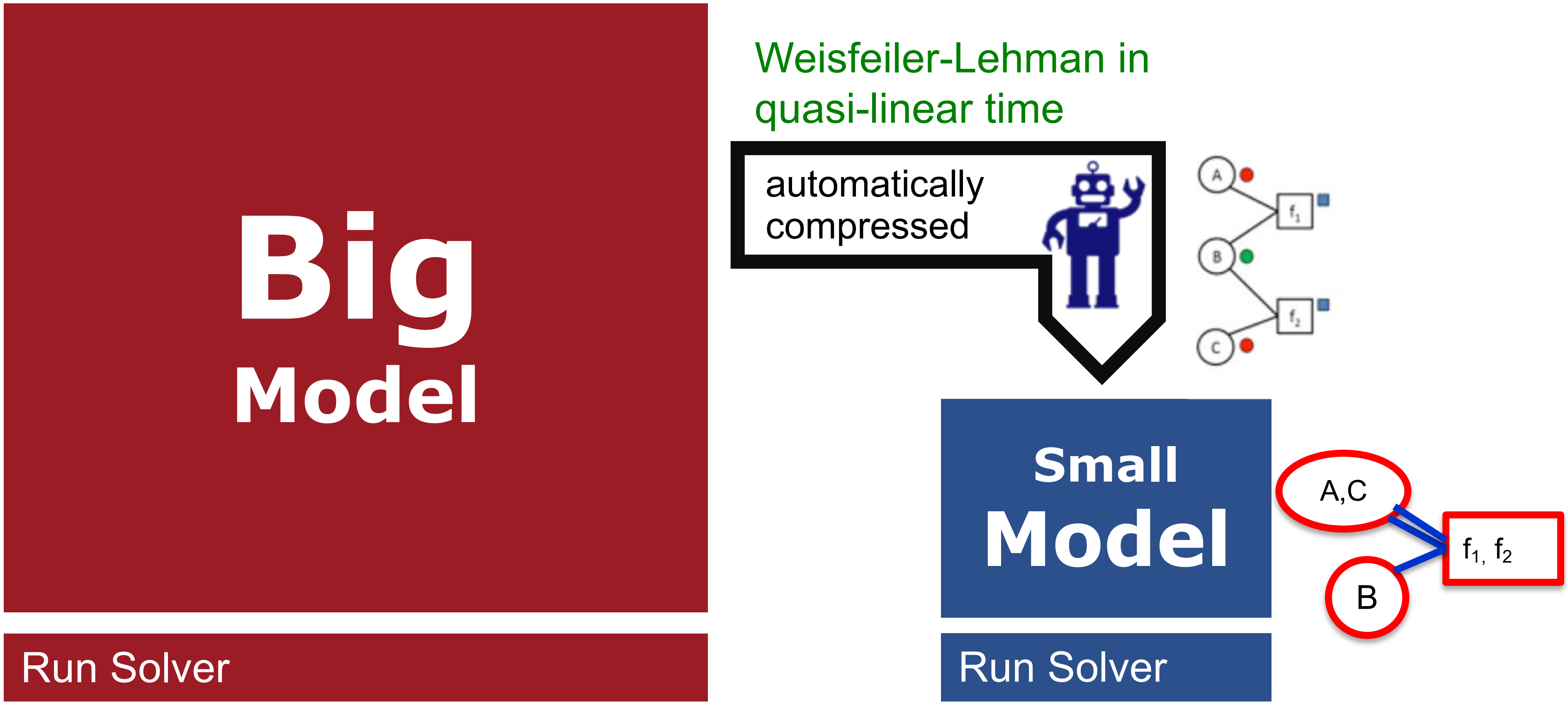
[Kersting, Ahmadi, Natarajan UAI'09; Ahmadi, Kersting, Mladenov, Natarajan MLJ'13, Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]



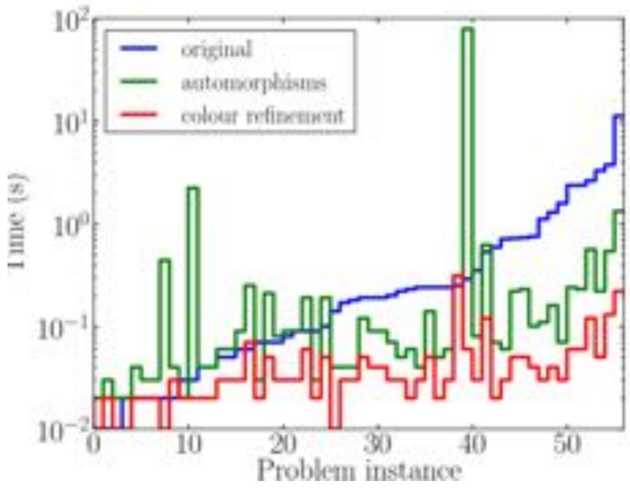

1. Each factor collects the colors of its neighboring nodes
2. Each factor „signs" its color signature with its own color
3. Each node collects the signatures of its neighboring factors
4. Nodes are recolored according to the collected signatures
5. If no new color is created stop, otherwise go back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

# Lifted Mathematical Programming
## Exploiting computational symmetries

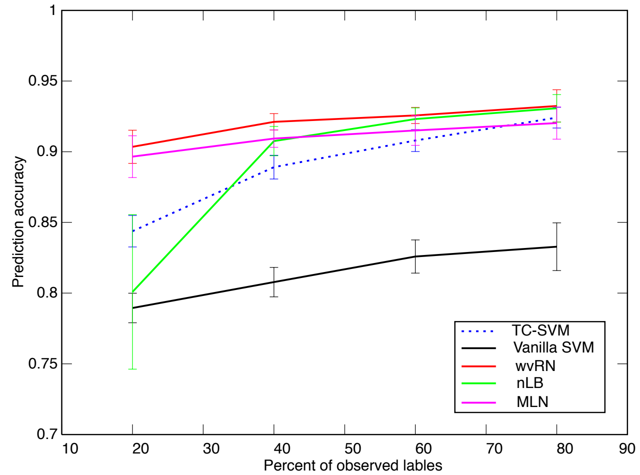
[Mladenov, Ahmadi, Kersting AISTATS´12, Grohe, Kersting, Mladenov, Selman ESA´14, Kersting, Mladenov, Tokmatov AIJ´17]



TECHNISCHE UNIVERSITÄT DARMSTADT

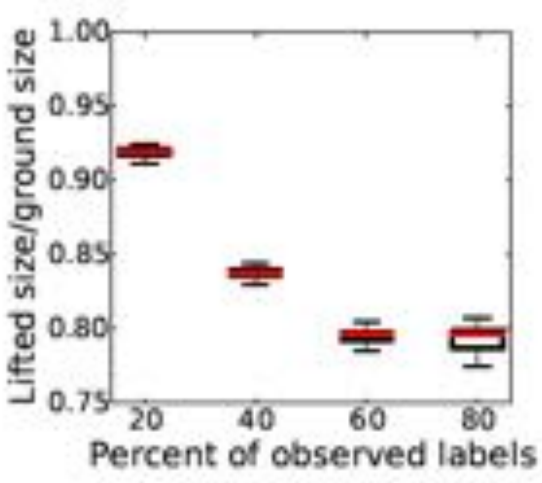


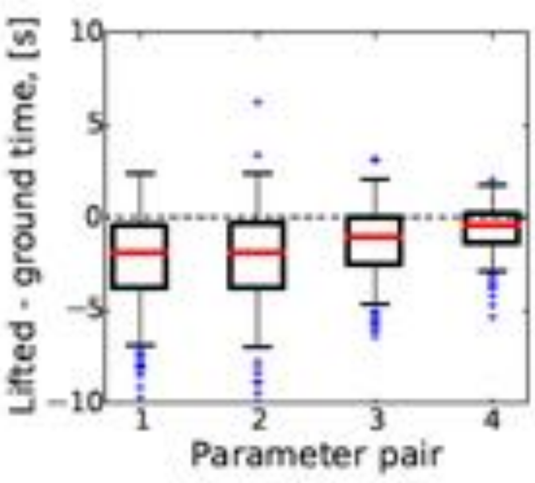

# Margout's ILPs with symmetries (relaxed)

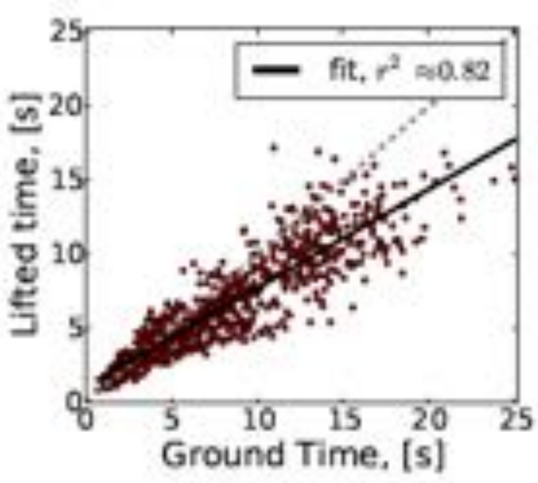



# Collective Classification

**Cora (most common vs. rest)**



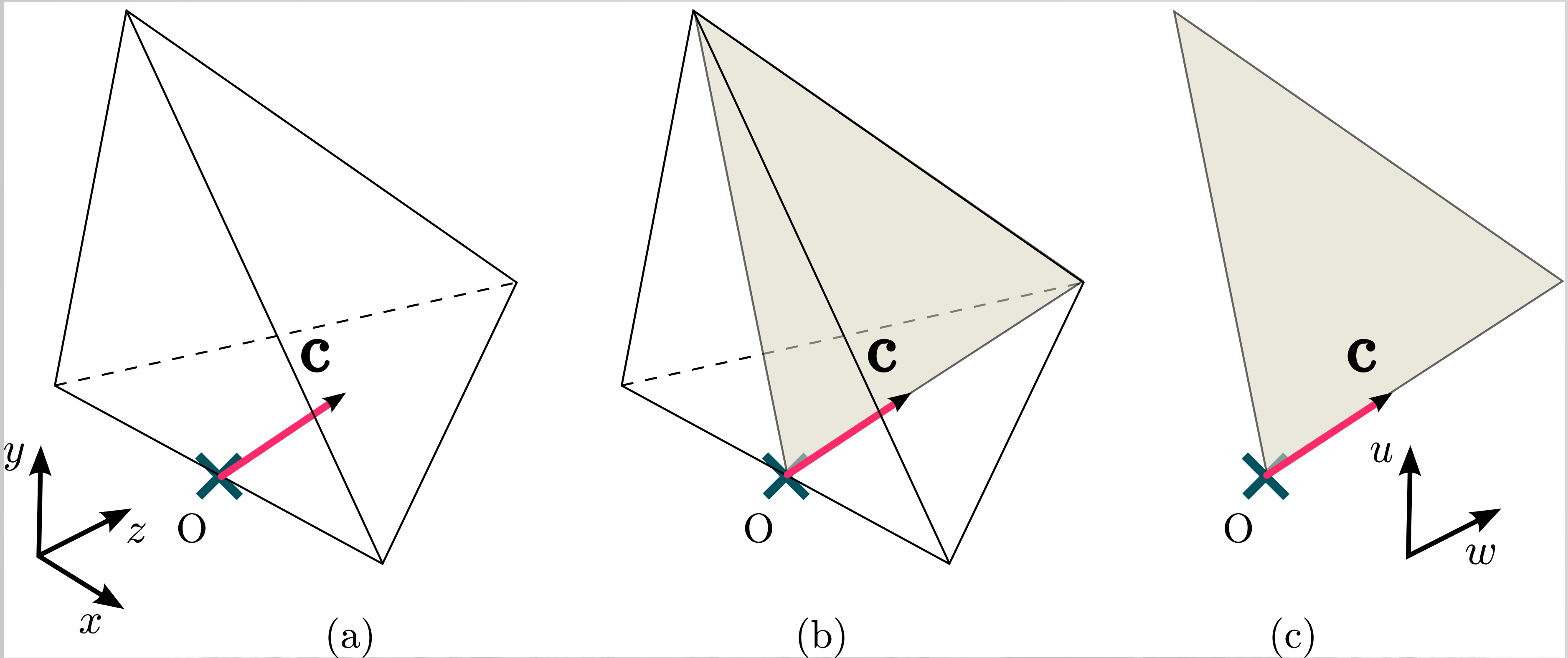

**The more observed the more lifting**

**Faster end-to-end even in the light of Gurobi's fast pre-solving heuristics**

[Boyd, Diaconis, Parrilo, Xiao: Internet Mathematics 2(1):31-71´05]

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-morpishm of the LP

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

# Why does this work?

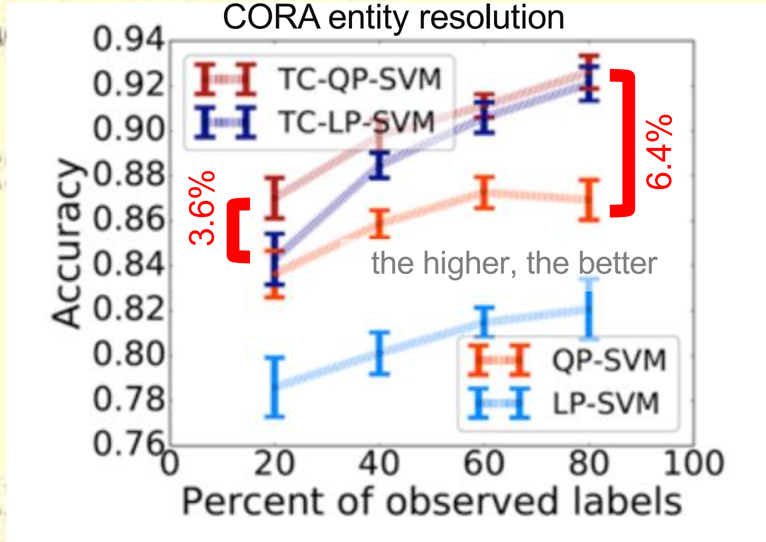
# Holds also for Convex QPs

Mladenov, Kleinhans, Kersting AAAI´17

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

```
#QUADRATIC OBJECTIVE
minimize: sum{J in feature(I,J)} weight(J)**2 + c1 * sla

#labeled examples should be on the correct side
subject                                        t(

#slacks
subject

#TRANSDUCTIVE PART
#cited instances should have the same labels.
subject to forall {I1, I2 in linked(I1, I2)}: labeled(I1                              ;
subject to forall {I1, I2 in linked(I1, I2)}: coslack(I1, I2) >= 0; #coslacks are positive
```

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

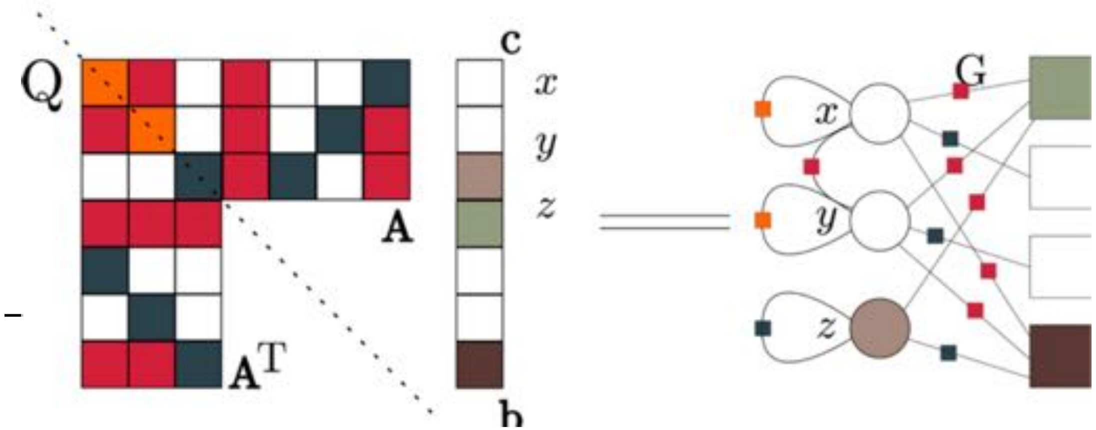



Papers that cite each other should be on the same side of the hyperplane

## Reduce the QP by running Weisfeiler-Lehman on the QP-Graph

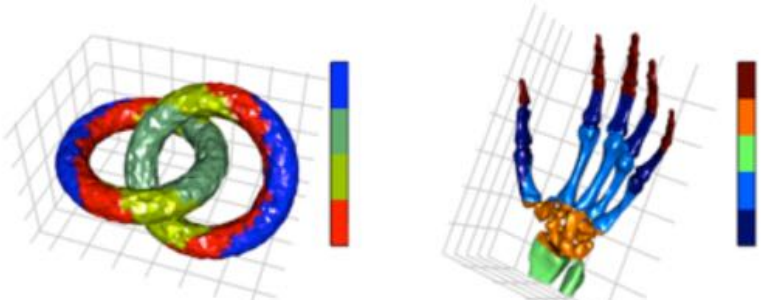
$$\max_{[x,y,z]^T \in \mathbb{R}^3} \quad 0x + 0y + 1z - 1z^2 - 2x^2 - 2y^2 + 1xy + 1yx$$

$$\text{s.t.} \quad \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} -$$

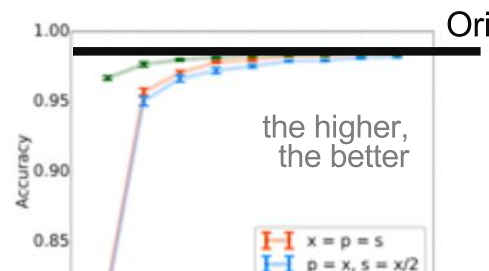


Mladenov, Kleinhans, Kersting AAAI´17

**Approximately Lifted SVM:**
Cluster data points via K-means using sorted distance vectors. Solve SVM on cluster representatives only



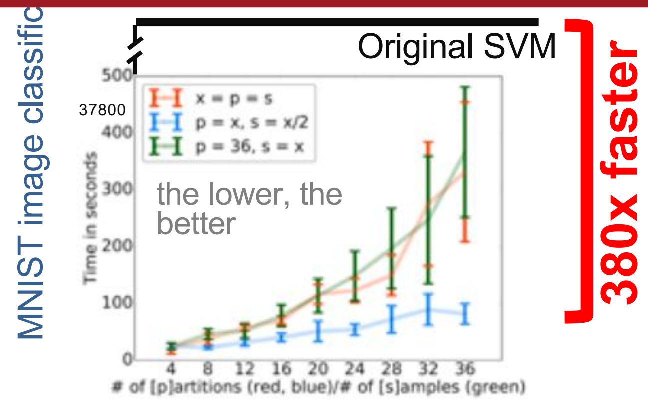
**Symmetry-based Data Programming:**
fractional autom. of label-preserving data trans-formations




**Same should work for deep networks**

**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.

**Similar predictive performance but 47x faster**

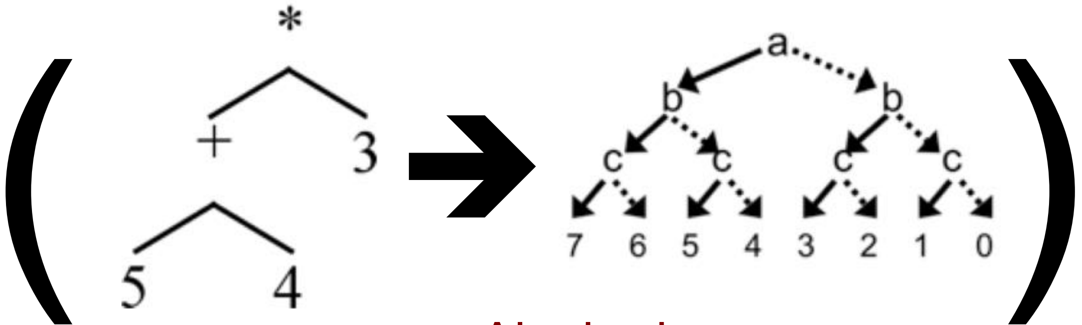


(d) Learning time: Translation

[Mladenov, Belle, Kersting AAAI´17]

TECHNISCHE UNIVERSITÄT DARMSTADT

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

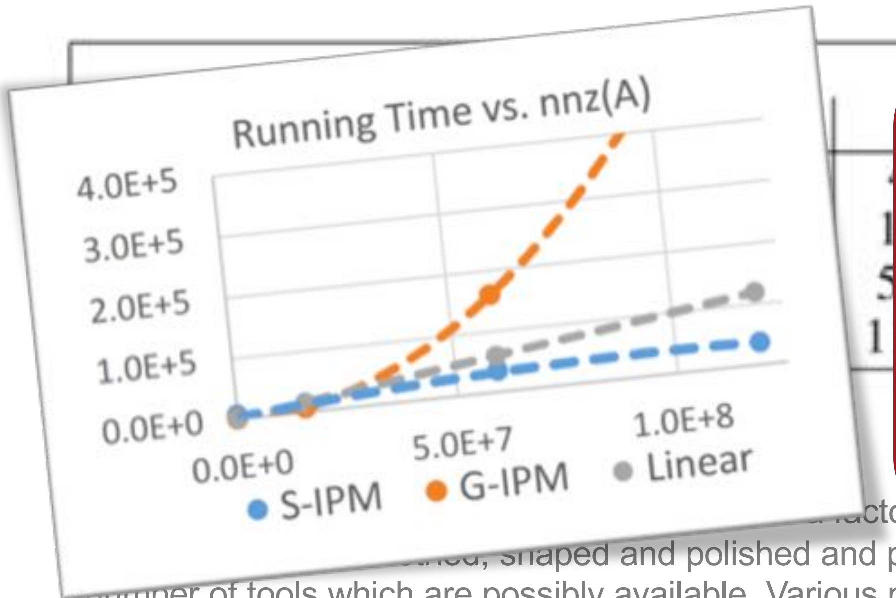



$$+ \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Formulae parse trees

Algebraic Decision Diagrams

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, …

…factory … …shed, shaped and polished and possibly drilled before painting, each of which actions require a number of tools which are possibly available. Various painting and connection methods are represented, each having an effect on the quality of the job, and each requiring tools. Rewards (required quality) range from 0 to 10 and a discounting factor of 0. 9 was used used

# There are strong invests into probabilistic programming

RelationalAI, Apple, Microsoft and Uber are investing hundreds of millions of US dollars

# Since we need languages for Systems AI,

the computational and mathematical modeling of complex AI systems.

[Laue et al. NeurIPS 2018; Kordjamshidi, Roth, Kersting: "Systems AI: A Declarative Learning Based Programming Perspective." IJCAI-ECAI 2018]

Th ... y

...bu ... e,
d

Eric Schmidt, Executive Chairman, Alphabet Inc.: Just Say "Yes", Stanford Graduate School of Business, May 2, 2017.https://www.youtube.com/watch?v=vbb-AjiXyh0.

# Overall, AI/ML/DS indeed refine "formal" science, but …

- **AI is more than deep neural networks.** Probabilistic and causal models are whiteboxes that provide insights into applications
- **AI is more than a single table.** Loops, graphs, different data types, relational DBs, … are central to data science and high-level programming languages for DS help to capture this complexity
- **AI is more than just Machine Learners and Statisticians**

**Learning-based programming offers a framework for building systems that help to go beyond, democratize, and even automize traditional AI/ML/DS**

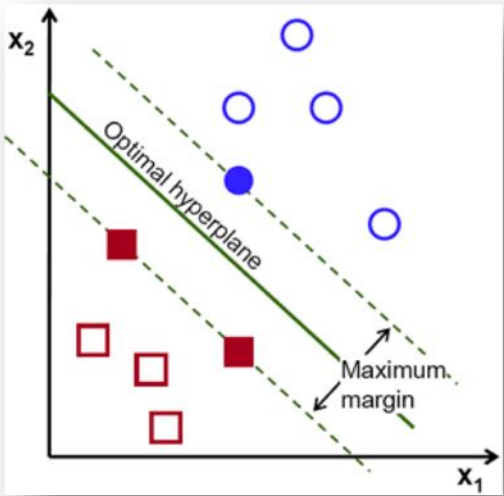
# Not every Data Science machine is generative

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \ \mathcal{P}(\mathbf{w},b,\boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to} \ \begin{cases} \forall i \quad y_i(\mathbf{w}^\top \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\ \forall i \quad \xi_i \geq 0 \end{cases}$$

## Not everyone likes to turn math into code

**Support Vector Machines**
Cortes, Vapnik MLJ 20(3):273-297, 1995



# High-level Languages for Mathematical Programs

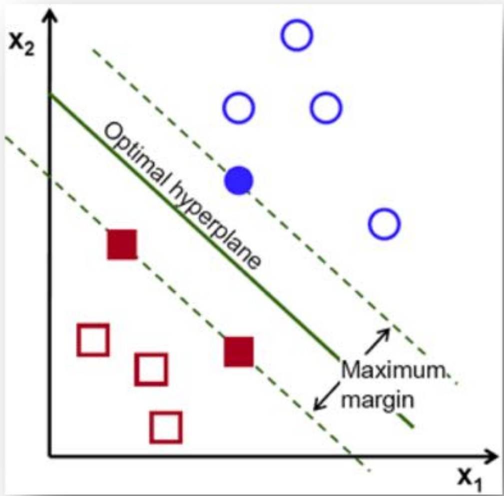
**Write down SVM in „paper form.“ The machine compiles it into solver form.**

```
#QUADRATIC OBJECTIVE
minimize: sum{J in feature(I,J)} weight(J)**2 + c1 * slack + c2 * coslack;

#labeled examples should be on the correct side
subject to forall {I in labeled(I)}: labeled(I)*predict(I) >= 1 - slack(I);

#slacks are positive
subject to forall {I in labeled(I)}: slack(I) >= 0;
```

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

reloop

RELOOP: A Toolkit for Relational Convex Optimization




**Support Vector Machines**
Cortes, Vapnik MLJ 20(3):273-297, 1995

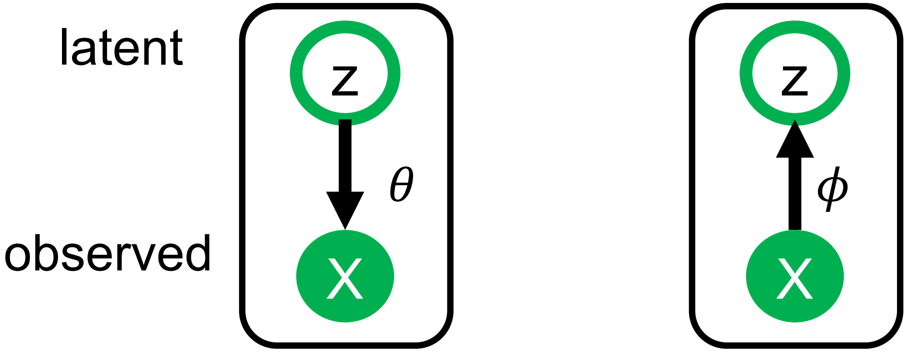
**In general, computing the exact posterior is intractable, i.e., inverting the generative process to determine the state of latent variables corresponding to an input is time-consuming and error-prone.**

# Deep Probabilistic Programming

```python
import pyro.distributions as dist

def model(data):
    # define the hyperparameters that control the beta prior
    alpha0 = torch.tensor(10.0)
    beta0 = torch.tensor(10.0)
    # sample f from the beta prior
    f = pyro.sample("latent_fairness", dist.Beta(alpha0, beta0))
    # loop over the observed data
    for i in range(len(data)):
        # observe datapoint i using the bernoulli
        # likelihood Bernoulli(f)
        pyro.sample("obs_{}".format(i), dist.Bernoulli(f), obs=data[i])
```

```python
def guide(data):
    # register the two variational parameters with Pyro.
    alpha_q = pyro.param("alpha_q", torch.tensor(15.0),
                         constraint=constraints.positive)
    beta_q = pyro.param("beta_q", torch.tensor(15.0),
                        constraint=constraints.positive)
    # sample latent_fairness from the distribution Beta(alpha_q, beta_q)
    pyro.sample("latent_fairness", dist.Beta(alpha_q, beta_q))
```

**(2) Ease the implementation by some high-level, probabilistic programming language**




Deep Neural Network

**(1) Instead of optimizating variational parameters for every new data point, use a deep network to predict the posterior given X** [Kingma, Welling 2013, Rezende et al. 2014]

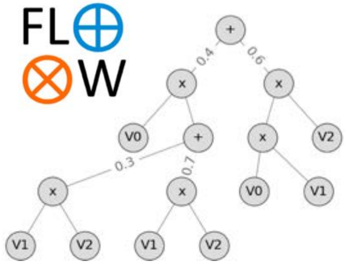
[Stelzner, Molina, Peharz, Vergari, Trapp, Valera, Ghahramani, Kersting ProgProb 2018]

# Sum-Product Probabilistic Programming

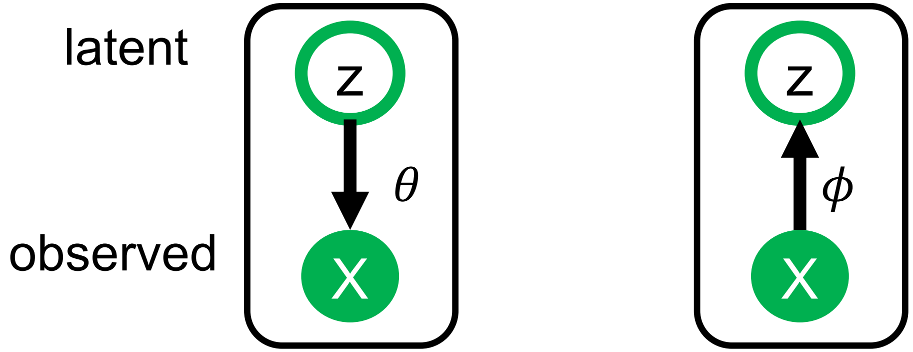
```python
import pyro.distributions as dist

def model(data):
    # define the hyperparameters that control the beta prior
    alpha0 = torch.tensor(10.0)
    beta0 = torch.tensor(10.0)
    # sample f from the beta prior
    f = pyro.sample("latent_fairness", dist.Beta(alpha0, beta0))
    # loop over the observed data
    for i in range(len(data)):
        # observe datapoint i using the bernoulli
        # likelihood Bernoulli(f)
        pyro.sample("obs_{}".format(i), dist.Bernoulli(f), obs=data[i])
```

```python
def guide(data):
    # register the two variational parameters with Pyro.
    alpha_q = pyro.param("alpha_q", torch.tensor(15.0),
                         constraint=constraints.positive)
    beta_q = pyro.param("beta_q", torch.tensor(15.0),
                        constraint=constraints.positive)
    # sample latent_fairness from the distribution Beta(alpha_q, beta_q)
    pyro.sample("latent_fairness", dist.Beta(alpha_q, beta_q))
```

Sum-Product Network




**(2) Ease the implementation by some high-level, probabilistic programming language**

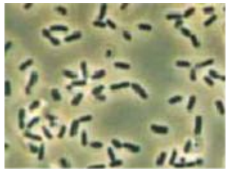


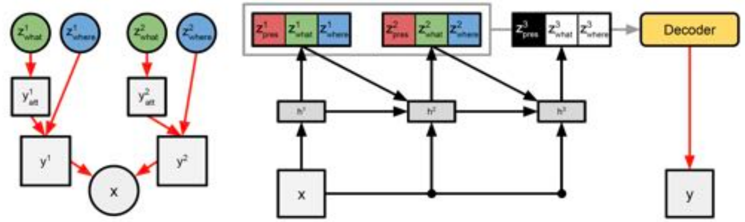

Deep Neural Network

**(1) Instead of optimizating variational parameters for every new data point, use a deep network to predict the posterior given X** [Kingma, Welling 2013, Rezende et al. 2014]

# Unsupervised scene understanding

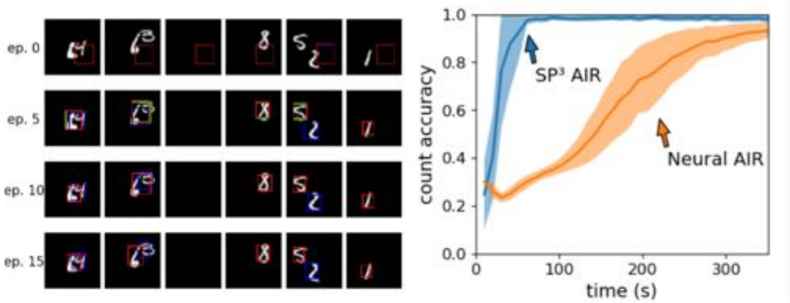
Consider e.g. unsupervised scene understanding using a generative model






[Attend-Infer-Repeat (AIR) model, Hinton et al. NIPS 2016]

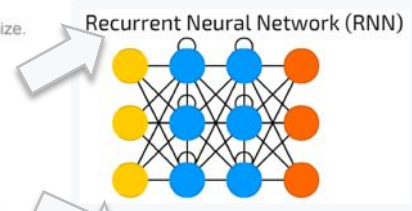
**Sum-Product Probabilistic Programming:**
Making machine learning and data science easier [Stelzner, Molina, Peharz, Vergari, Trapp, Valera, Ghahramani, Kersting ProgProb 2018]

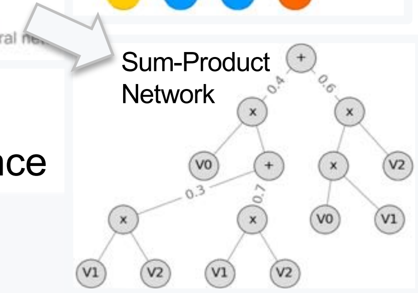



**Probabilistic Programming:**
Easier modelling by programming generative models in a high-level, prob. language

```
def prior_step(t):
    # Sample object pose. This is a 3-dimensional vector representing x,y position and size.
    z_where = pyro.sample('z_where_{}'.format(t),
                          dist.normal,
                          z_where_prior_mu, z_where_prior_sigma)

    # Sample object code. This is a 50-dimensional vector.
    z_what = pyro.sample('z_what_{}'.format(t),
                         dist.normal,
                         z_what_prior_mu, z_what_prior_sigma)

    y_att = decode(z_what)          # Map latent code to pixel space using the neural ne
```

**Deep Probabilistic Prog.:**
Modelling and inference might be hard, so use a deep neural network for it



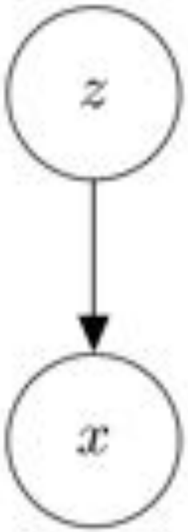




Use deep probabilistic models that feature tractable, deterministic inference

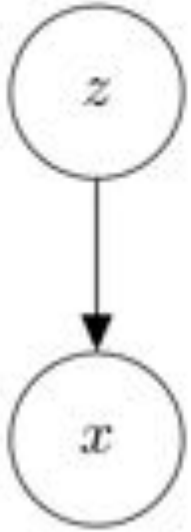
```
from spn.structure.leaves.parametric.Parametric import Categorical

from spn.structure.Base import Sum, Product

from spn.structure.base import assign_ids, rebuild_scopes_bottom_up


p0 = Product(children=[Categorical(p=[0.3, 0.7], scope=1), Categorical(p=[0.4, 0.6], scope=2)])
p1 = Product(children=[Categorical(p=[0.5, 0.5], scope=1), Categorical(p=[0.6, 0.4], scope=2)])
s1 = Sum(weights=[0.3, 0.7], children=[p0, p1])
p2 = Product(children=[Categorical(p=[0.2, 0.8], scope=0), s1])
p3 = Product(children=[Categorical(p=[0.2, 0.8], scope=0), Categorical(p=[0.3, 0.7], scope=1)])
p4 = Product(children=[p3, Categorical(p=[0.4, 0.6], scope=2)])
spn = Sum(weights=[0.4, 0.6], children=[p2, p4])

assign_ids(spn)
rebuild_scopes_bottom_up(spn)

return spn
```

# Actually, the main idea is to replace the VAEs within AIR by SPNs

VAE



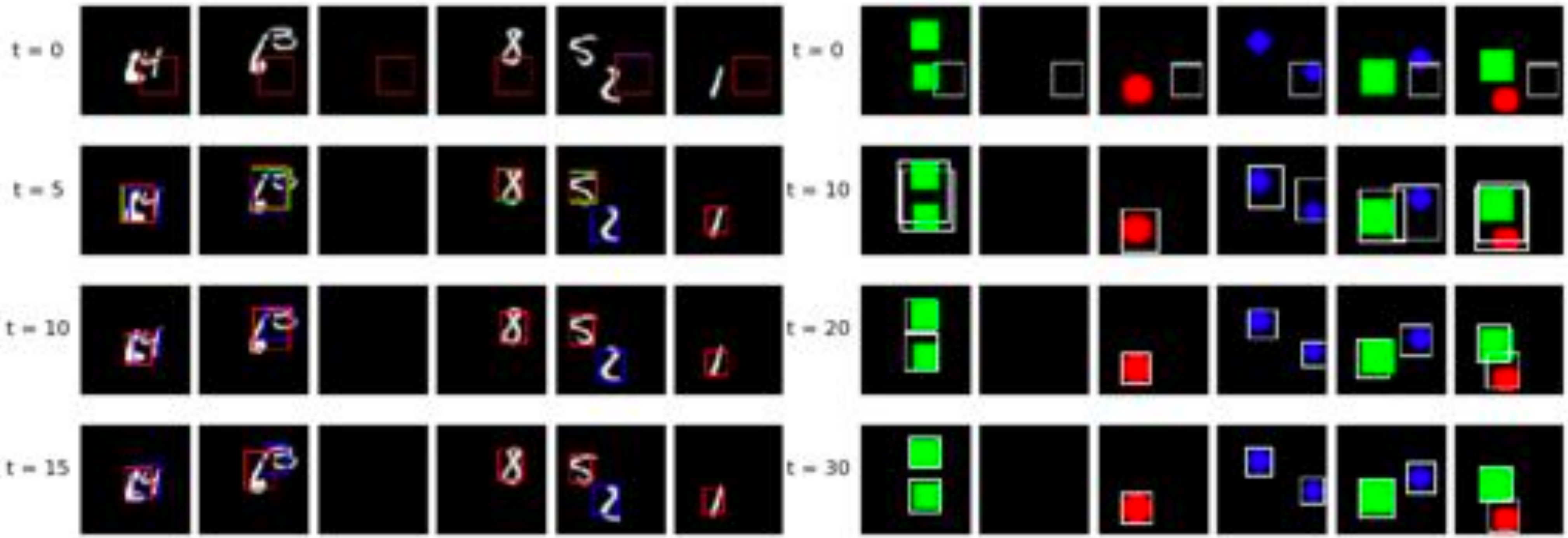

SPN




- infinite mixture model
- intractable density
- intractable posterior

- "large" but finite mixture model
- tractable density
- tractable marginals [Peharz et al., 2015]
- tractable posterior [Vergari et al., 2017]

# Sum-Product Attent-Infer Repeat

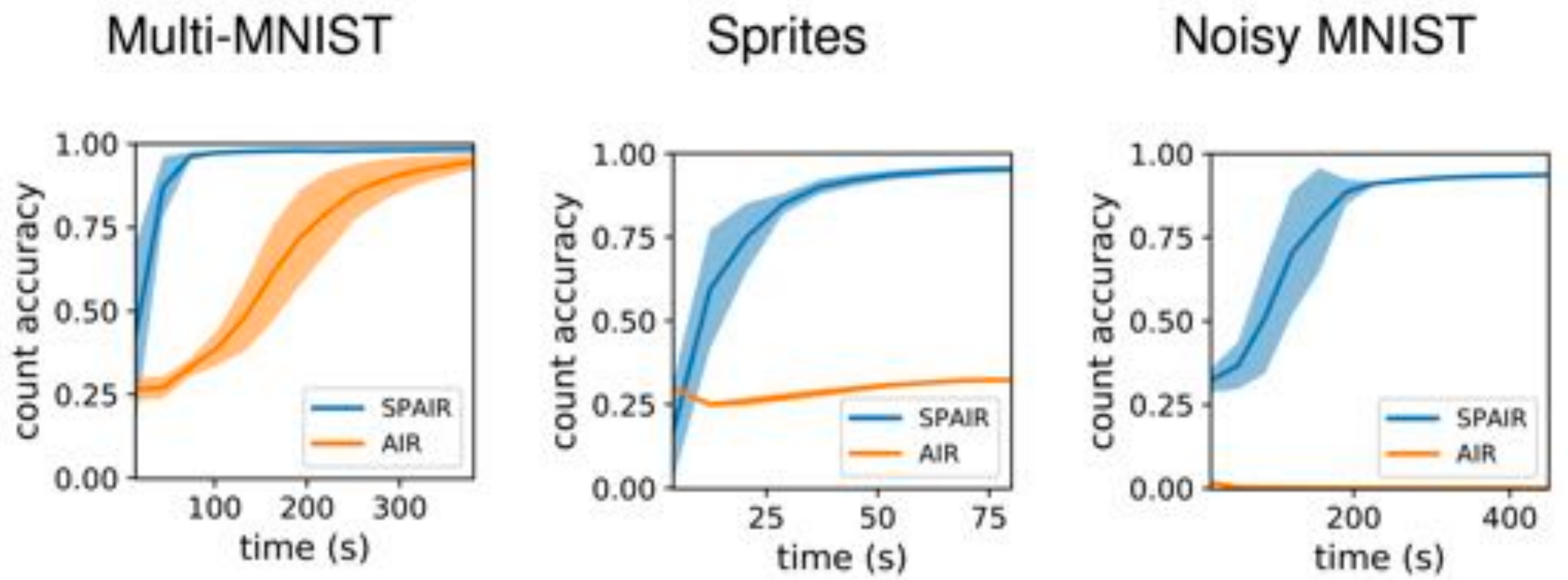



[Stelzner, Peharz, Kersting 2019]

UNIVERSITY OF CAMBRIDGE

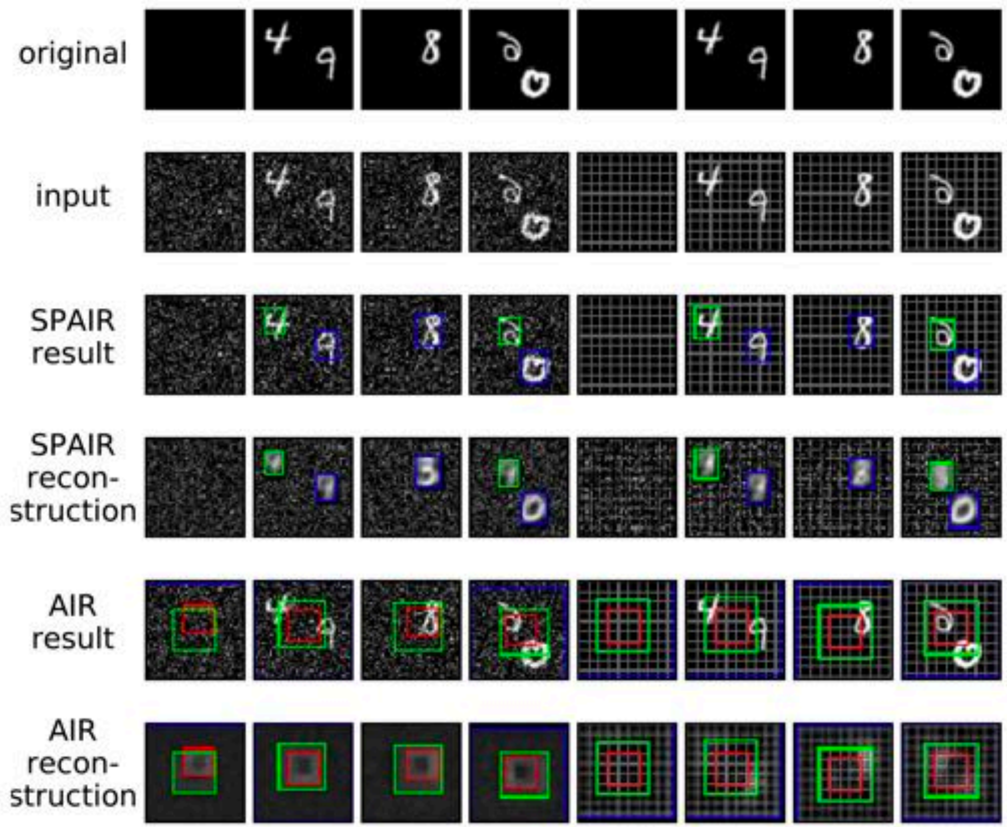
TECHNISCHE UNIVERSITÄT DARMSTADT

# Sum-Product Attent-Infer Repeat




[Stelzner, Peharz, Kersting 2019]

# Sum-Product Attent-Infer Repeat




[Stelzner, Peharz, Kersting 2019]

UNIVERSITY OF CAMBRIDGE

TECHNISCHE UNIVERSITÄT DARMSTADT