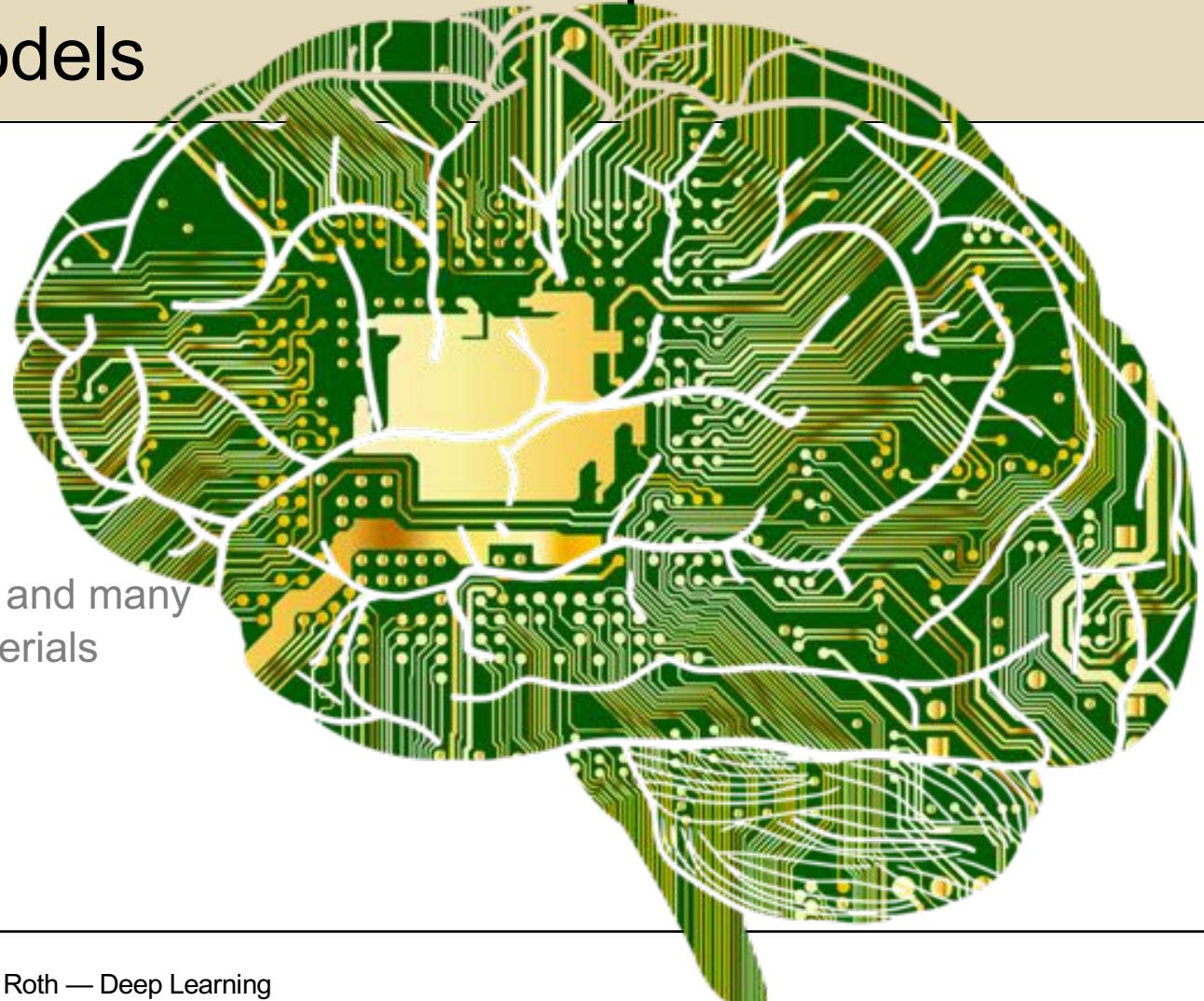


Deep Learning

Architectures and Methods: Deep Probabilistic Models



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Thanks to Pedro Domingos and many others for making their materials publically available.

Judea Pearl received 2012 the ACM Turing Award 2012 for his work on graphical models



Goals

Get in touch with

1. one of the most exciting topics in AI and ML in recent years, namely **Graphical Models (GMs)**, and their basics for **inference**, and
2. tools for making GM „**tractable**“

Caution! Necessarily incomplete!



Judea Pearl received 2012 the ACM Turing Award 2012 for his work on graphical models



Roadmap of the course

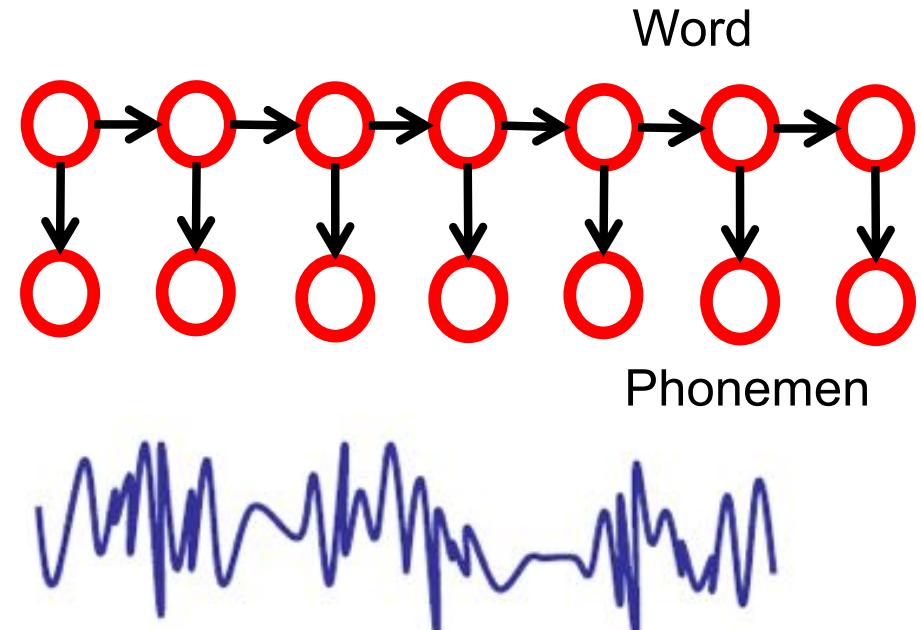
1 Basics of graphical models

2a Exploiting symmetries for inference

2b Sum-product networks



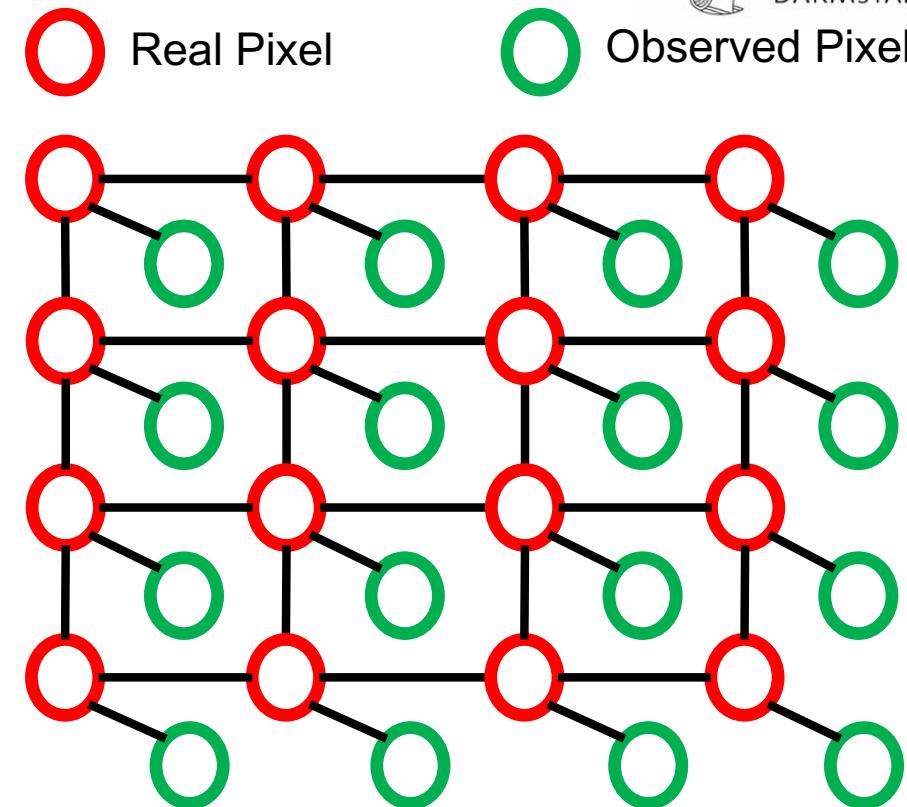
Application: Speech recognition



Infer the words from spoken language:
Hidden Markov Model



Application: Image Denoising



Infer the original image from the noisy observed one: Markov networks



Graphical models are omnipresent



Information retrieval, search, collaborative filtering, gene expression analysis, natural language processing, bioinformatics, and many,

many,

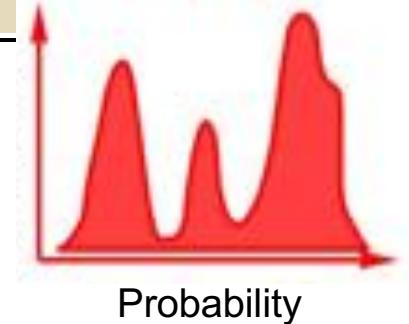
many,

many

more!

OK, so what are probabilities and graphical models?

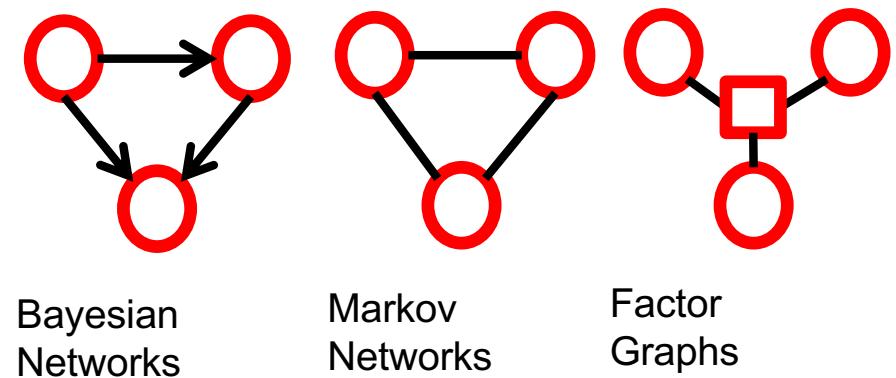




In a Nutshell, Graphical Models are ...

... a graphical notation for (conditional) independency assumptions and therefore a (hopefully) compact specification of probability distributions

Nodes=
Random Variables (RVS)
Edges=
Dependencies among RVs

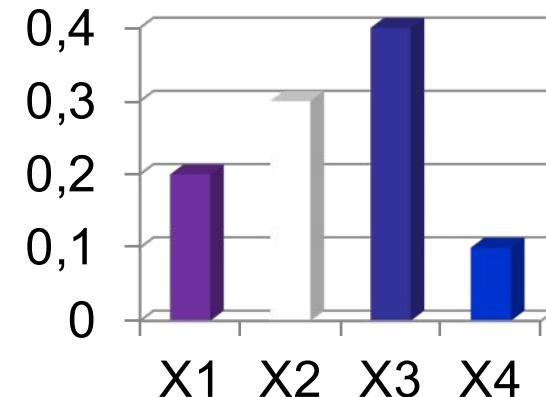


Discrete Random Variables

Finite set X of possible states

$$X \in \{x_1, x_2, x_3, \dots, x_n\}$$

$$P(x_i) \geq 0 \quad \sum_{i=1}^n P(x_i) = 1$$



OK, but answering the question requires the joint distribution

What is the probability that smoking causes cancer?



no	few	many
0.800	0.150	0.050



no	benigne	maligne
0.935	0.046	0.019

Joint Distribution

Probability that $X=x$ and $Y=y$ are “true”

$$P(x, y) \equiv P(X = x \wedge Y = y)$$

		Cancer		
		no	benigne	maligne
Smoking	no	0.768	0.024	0.008
	few	0.132	0.012	0.006
	many	0.035	0.010	0.005

The joint distribution allows us to answer any question! But how?



Make use of basic probability theory

Marginalization

$$P(Y) = \sum_{i=1}^n P(Y, x_i)$$

sum →

		Cancer			P(smoking)
		No	Benigne	Maligne	
Smoking	No	0.768	0.024	0.008	0.800
	few	0.132	0.012	0.006	0.150
	many	0.035	0.010	0.005	0.050
	TOTAL	0.935	0.046	0.019	

P(cancer)

Product rule & **conditional probability**

$$P(X, Y) = P(X | Y)P(Y) = P(Y | X)P(X)$$

Probability that $X=x$ if we have observed $Y=y$ ($P(y)>0$)

Probably the most important rule: Bayes

$$P(R, K) = P(R | K)P(K) = P(K | R)P(R)$$

$$P(R | K) = \frac{P(K | R)P(R)}{P(K)} = \frac{P(K, R)}{P(K)}$$

		cancer		
		no	benigne	maligne
smoking	no	0.768	0.024	0.008
	few	0.132	0.012	0.006
	many	0.035	0.010	0.005
	TOTAL	0.935	0.046	0.019

P(cancer)

Since we know already $P(R, K)$ und auch $P(K)$, just divide them.



Probably the most important rule: Bayes

$$P(R, K) = P(R | K)P(K) = P(K | R)P(R)$$

$$P(R | K) = \frac{P(K | R)P(R)}{P(K)} = \frac{P(K, R)}{P(K)}$$

		cancer		
		no	benigne	maligne
smoking	no	0.768/0.935	0.024/ 0.46	0.008/ 0.019
	few	0.132/0.935	0.012/ 0.46	0.006/ 0.019
	many	0.035/0.935	0.010/ 0.46	0.005/ 0.019
	TOTAL	0.935	0.046	0.019

P(cancer)



Probably the most important rule: Bayes

$$P(R, K) = P(R | K)P(K) = P(K | R)P(R)$$

$$P(R | K) = \frac{P(K | R)P(R)}{P(K)} = \frac{P(K, R)}{P(K)}$$

		cancer= ...)		
		no	benigne	maligne
P(smoking	no	0.821	0.522	0.421
	few	0.141	0.261	0.316
	many	0.037	0.217	0.263

As long as all entries are >0 , everything can be computed! Mission completed?

No!! Our mission has just started

Joint distribution is enumerating everything

- Worst-case run time: $O(2^n)$
 - $n = \# \text{ of RVs}$
- Space is $O(2^n)$ too
 - Size of the table of the joint distribution

Main idea: make use of independencies to compress the representation



Indpendency

(Current) age and the gender of a person
are independent



$$P(G, A) = P(G) \cdot P(A)$$

$$P(A | G) = P(A)$$

$$P(G | A) = P(G)$$

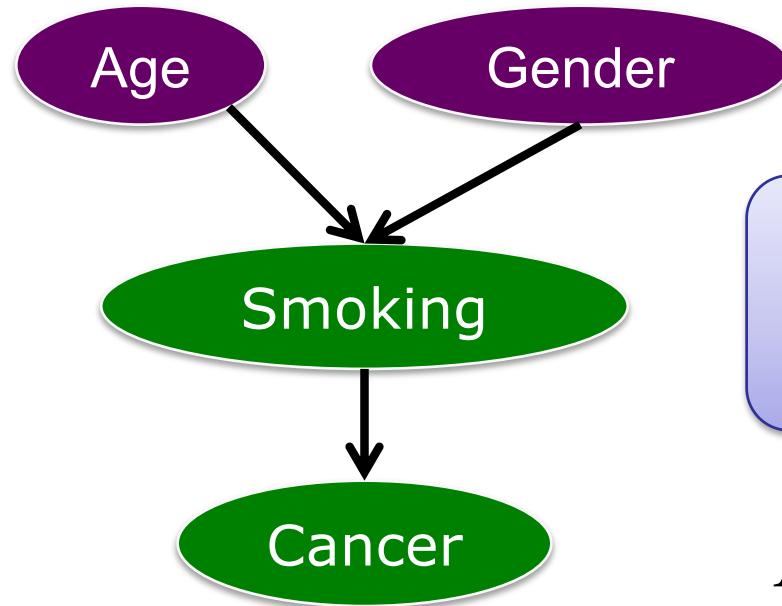
You would not give me money for information on the gender
to know the age of a person!



Conditional Independence

Cancer is independent of age and gender, if the person smokes.

If you have not observed anything, age and gender are independent.



Less entries, therefore lower complexity

$$P(C | S, G, A) = P(C | S)$$



Bayesian Networks

[Pearl 1989]

Set of random variables $\{X_1, \dots, X_n\}$

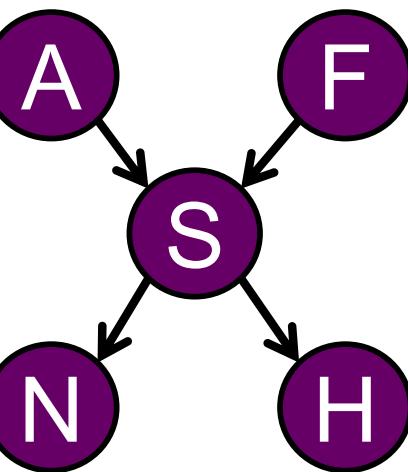
Directed, acyclic graph (DAG)

To each RV X_i we associate the
conditional probability distribution:

$$P(X_i | \text{Pa}(X_i))$$

The **joint distribution** is

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | \text{Pa}(X_i))$$



BN semantics

Local Markov Assumption

Each RV X is independent of its „non-descendant“ given its parents ($X_i \perp \text{nonDescendants} | \text{Pa}_{X_i}$)



Example

But how do we do inference?

$$R \in \{no, few, many\}$$



$P(S=n)$	0.80
$P(S=f)$	0.15
$P(S=m)$	0.05

$$K \in \{no, benigne, maligne\}$$

Smoking=	n	f	m
$P(C=n)$	0.96	0.88	0.60
$P(C=b)$	0.03	0.08	0.25
$P(C=m)$	0.01	0.04	0.15



What is Inference?

Query: $P(X | e)$

Definition of conditional probability $P(X | e) = \frac{P(X, e)}{P(e)}$

Up to normalization $P(X | e) \propto P(X, e)$

Hence, this rewrites to

$$P(Y) = \sum_{X_i \notin Y} \left[\prod_{i=1}^n P(X_i | \text{Pa}(X_i)) \right]$$

BN semantics

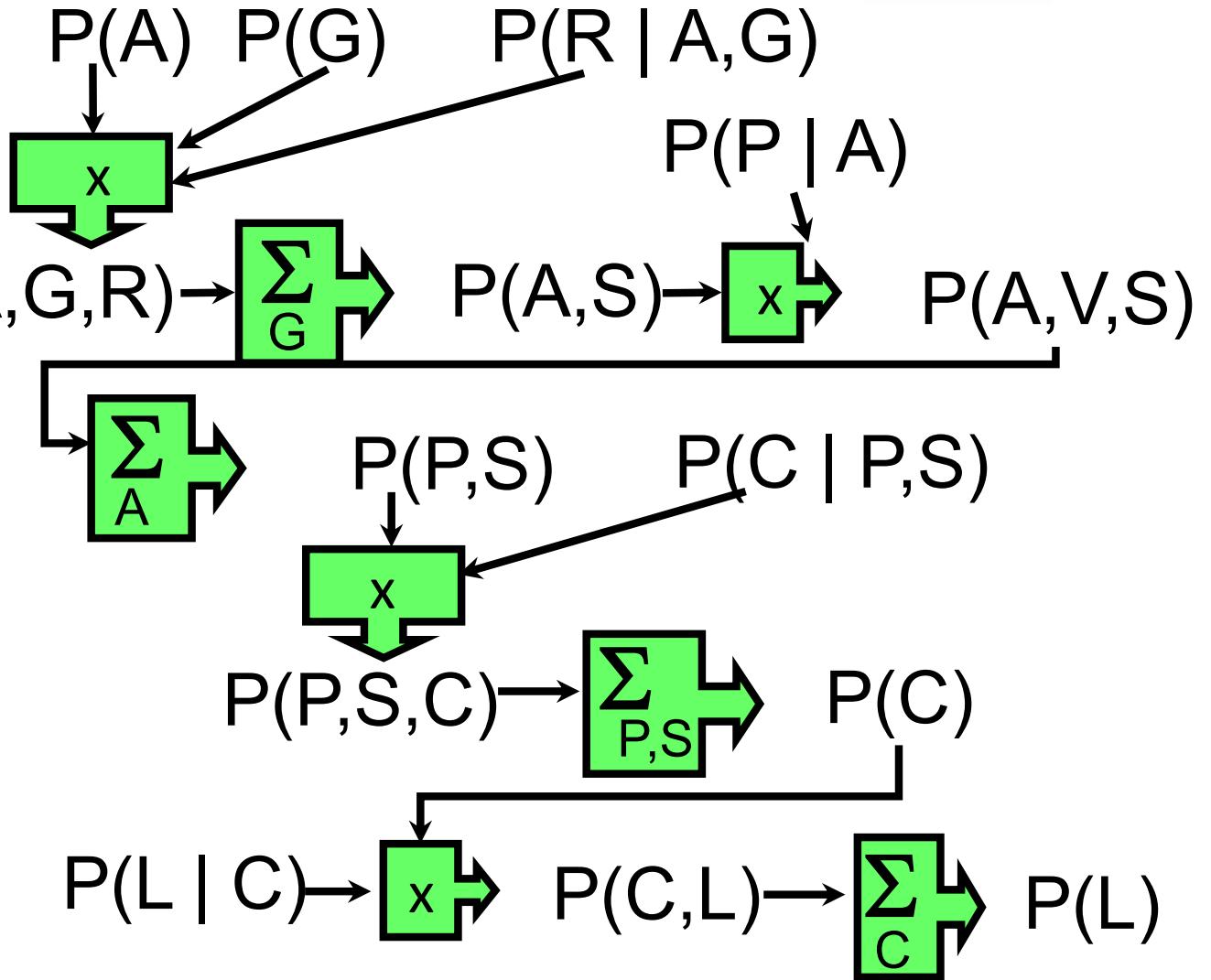
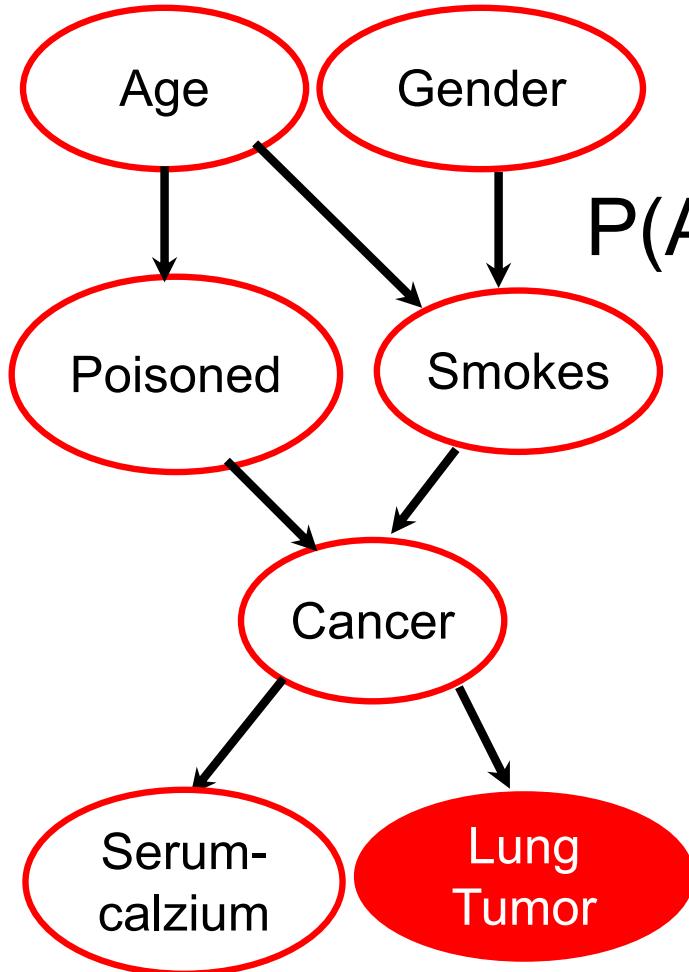
Marginalization

Main observation: Σ and \sqcap commute

$$\Sigma_a (P_1 \times P_2) = (\Sigma_a P_1) \times P_2 \text{ if } A \text{ is not in } P_2$$

Complete example: Let's comput $P(L)$

elimination order is : G,A,V,R,K



Exponentially in the size of the largest (induced) factor (table) also called treewidth: 2^3 vs 2^7

As an algorithm, this is called: Variable elimination

Given a BN and a query $P(X|e) / P(X,e)$

Instantiate evidence e

Choose an elimination order over the variables, e.g., X_1, \dots, X_n

Initial factors $\{f_1, \dots, f_n\}$: $f_i = P(X_i | \text{Pa}_{X_i})$ (CPT for X_i)

For $i = 1$ to n , if $X_i \notin \{X, E\}$

- Collect factors f_1, \dots, f_k that include X_i
- Generate a new factor by eliminating X_i from these factors

$$g = \sum_{X_i} \prod_{j=1}^k f_j$$

- Variable X_i has been eliminated! Add g to the set of factors

Normalize $P(X,e)$ to obtain $P(X|e)$



What have we learned so far?

Uncertainty is omnipresent

Uncertainty can be captured using probability distributions

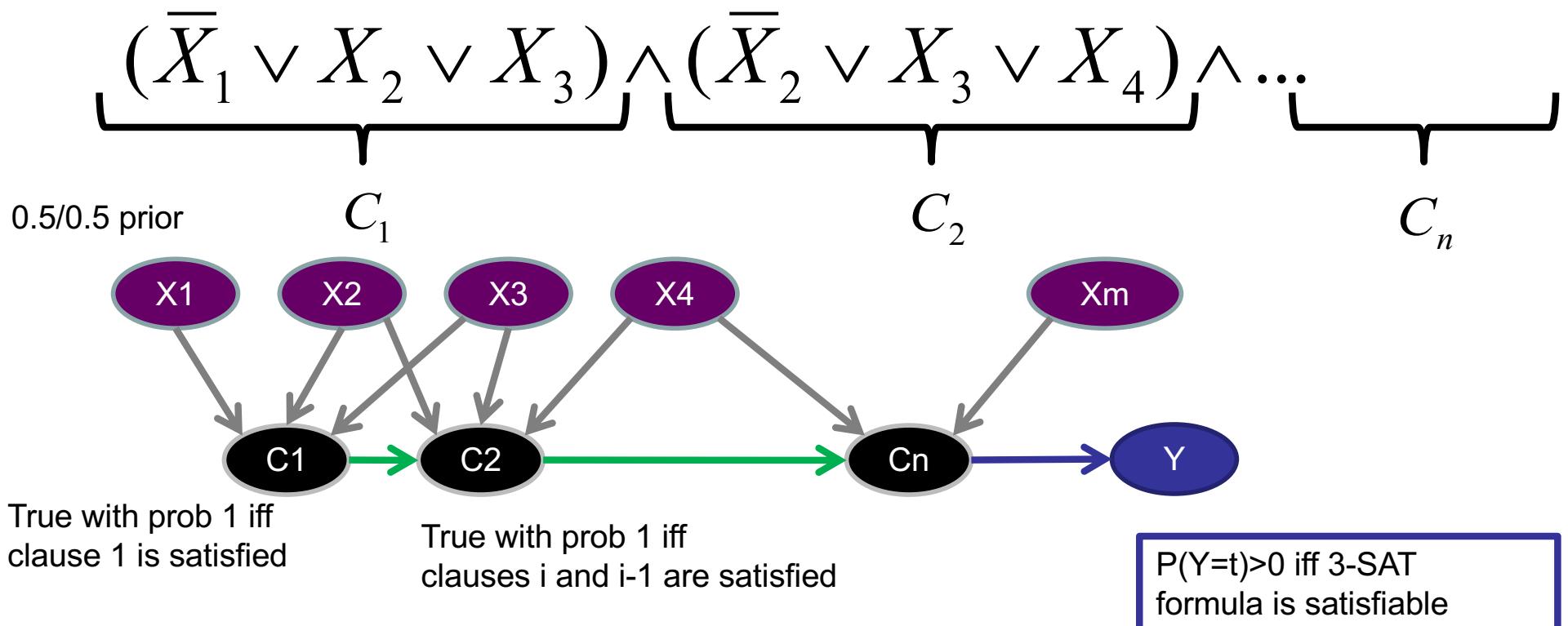
Graphical models are compact encodings of probability distributions

They lead to „efficient“ algorithms for inference such as Variablen-Elimination



Mission Completed? No ...

Theorem: Inference (even approximate) in Bayesian networks is NP-hard ($\#P$; via reduction to 3-SAT)



- ◆ What to do when we find a problem that looks hard...



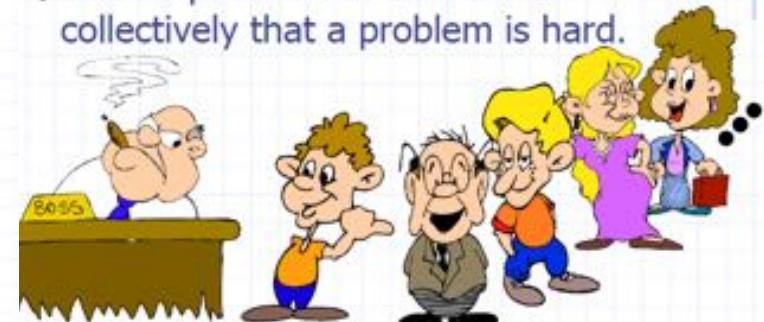
I couldn't find a polynomial-time algorithm;
I guess I'm too dumb.

- ◆ Sometimes we can prove a strong lower bound... (but not usually)

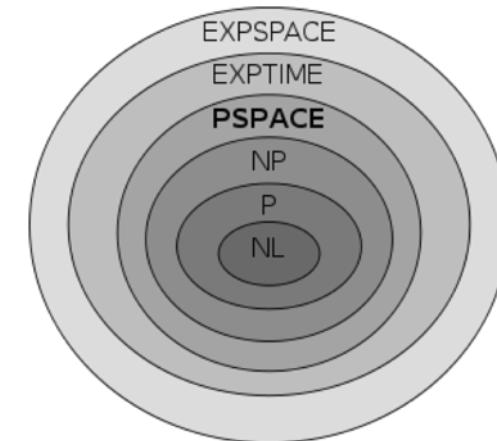
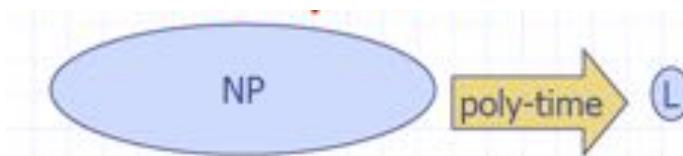
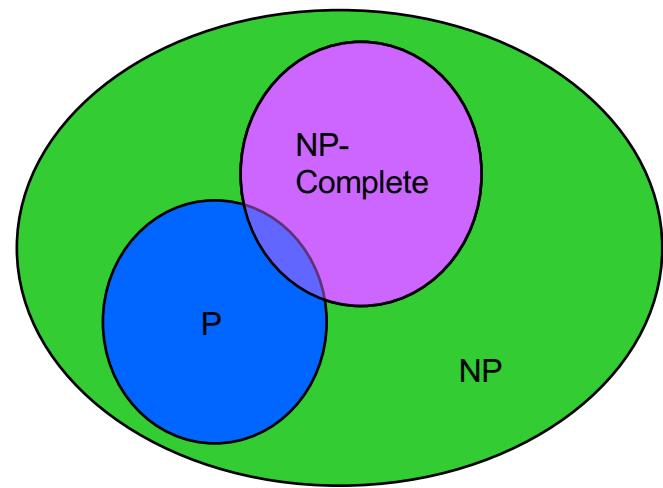


I couldn't find a polynomial-time algorithm,
because no such algorithm exists!

- ◆ NP-completeness let's us show collectively that a problem is hard.



I couldn't find a polynomial-time algorithm,
but neither could all these other smart people.

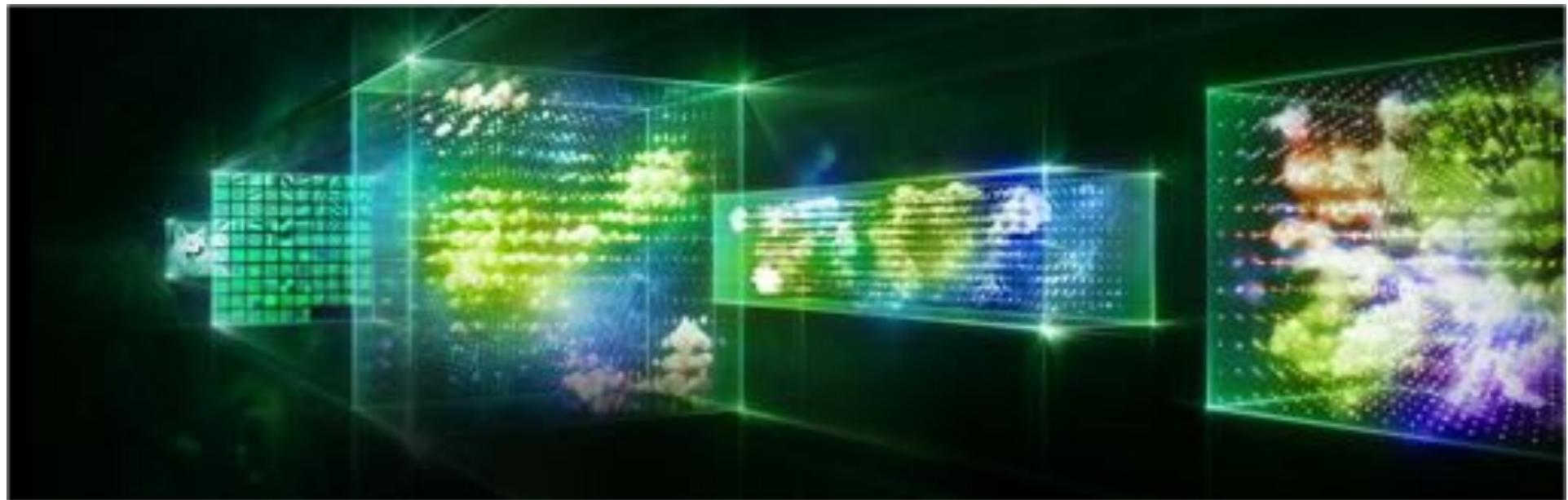


What have we learnt about Bayesian networks?

- Bayesian networks (BNs) encode joint distributions
- They are DAGs
 - (nodes = RVs, edges = dependencies)
- Inference is NP-hard
- Variable Elimination is one of the most basic inference approaches; there are many other inference approaches
- We have skipped learning BNs



Can we borrow ideas from deep learning for graphical models? And if so, are there advantages?



There is another major trend in AI/ML
DEEP LEARNING



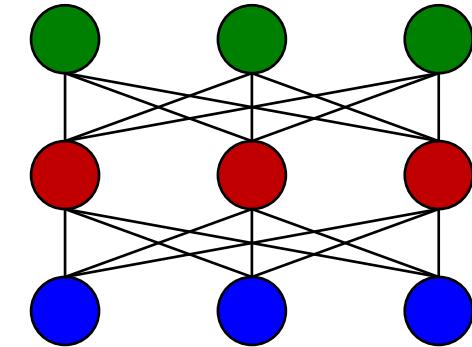
Deep Learning

Stack many layers

E.g.: DBN [Hinton & Salakhutdinov,2006]

CDBN [Lee et al.,2009]

DBM [Salakhutdinov & Hinton,2010]



Potentially much more powerful than shallow architectures, represent computations [Bengio, 2009]

But ...

- Often no probabilistic semantics
 - Learning requires extensive efforts



Alternative Representation: Graphical Models as Deep Networks

X_1	X_2	$P(X)$
1	1	0.4
1	0	0.2
0	1	0.1
0	0	0.3

$$\begin{aligned}
 P(X) = & 0.4 \cdot I[X_1=1] \cdot I[X_2=1] \\
 & + 0.2 \cdot I[X_1=1] \cdot I[X_2=0] \\
 & + 0.1 \cdot I[X_1=0] \cdot I[X_2=1] \\
 & + 0.3 \cdot I[X_1=0] \cdot I[X_2=0]
 \end{aligned}$$



Alternative Representation: Graphical Models as (Deep) Networks

X_1	X_2	$P(X)$
1	1	0.4
1	0	0.2
0	1	0.1
0	0	0.3

$$\begin{aligned} P(X) = & \mathbf{0.4} \cdot I[X_1=1] \cdot I[X_2=1] \\ & + 0.2 \cdot I[X_1=1] \cdot I[X_2=0] \\ & + 0.1 \cdot I[X_1=0] \cdot I[X_2=1] \\ & + 0.3 \cdot I[X_1=0] \cdot I[X_2=0] \end{aligned}$$



Shorthand for Indicators

X_1	X_2	$P(X)$
1	1	0.4
1	0	0.2
0	1	0.1
0	0	0.3

$$\begin{aligned}P(X) = & 0.4 \cdot X_1 \cdot X_2 \\& + 0.2 \cdot X_1 \cdot \bar{X}_2 \\& + 0.1 \cdot \bar{X}_1 \cdot X_2 \\& + 0.3 \cdot \bar{X}_1 \cdot \bar{X}_2\end{aligned}$$



Sum Out Variables

X_1	X_2	$P(X)$
1	1	0.4
1	0	0.2
0	1	0.1
0	0	0.3

$$e: X_1 = 1$$

$$\begin{aligned} P(e) &= \mathbf{0.4} \cdot X_1 \cdot X_2 \\ &\quad + \mathbf{0.2} \cdot X_1 \cdot \bar{X}_2 \\ &\quad + 0.1 \cdot \bar{X}_1 \cdot X_2 \\ &\quad + 0.3 \cdot \bar{X}_1 \cdot \bar{X}_2 \end{aligned}$$

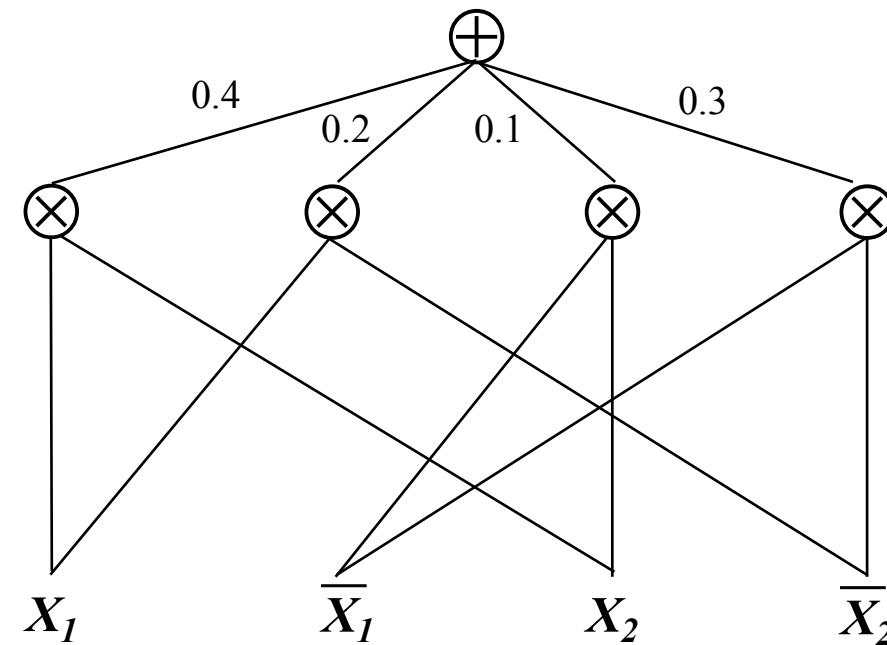
Set $X_1 = 1, \bar{X}_1 = 0, X_2 = 1, \bar{X}_2 = 1$

Easy: Set both indicators of X_2 to 1



Idea: Deeper Network Representation of a Graphical Model that encodes how to compute probabilities

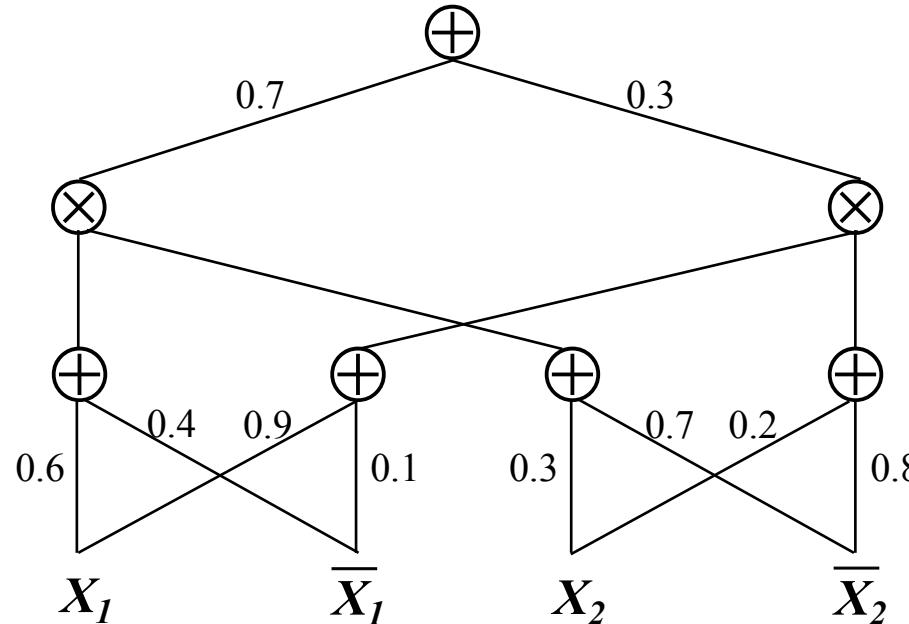
X_1	X_2	$P(X)$
1	1	0.4
1	0	0.2
0	1	0.1
0	0	0.3



Sum-Product Networks* (SPNs)

[Poon, Domingos UAI 2011]

A SPN **S** is a rooted DAG where:
Nodes: Sum, product, input indicator
Weights on edges from sum to children

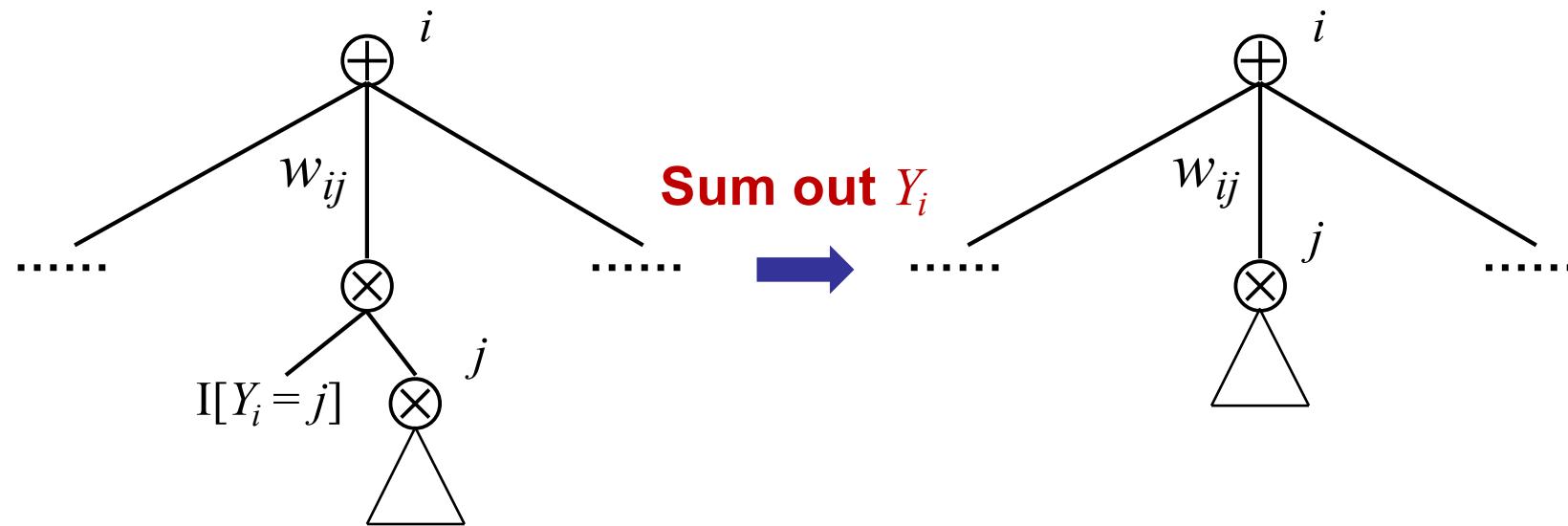


*SPNs are an instance of Arithmetic Circuits (ACs). ACs have been introduced into the AI literature more than 15 years ago as a tractable representation of probability distributions
[Darwiche CACM 48(4):608-647 2001]

Semantics of Sums and Products

Product ~ **Feature** → Form feature hierarchy

Sum ~ **Mixture** (with hidden var. summed out)



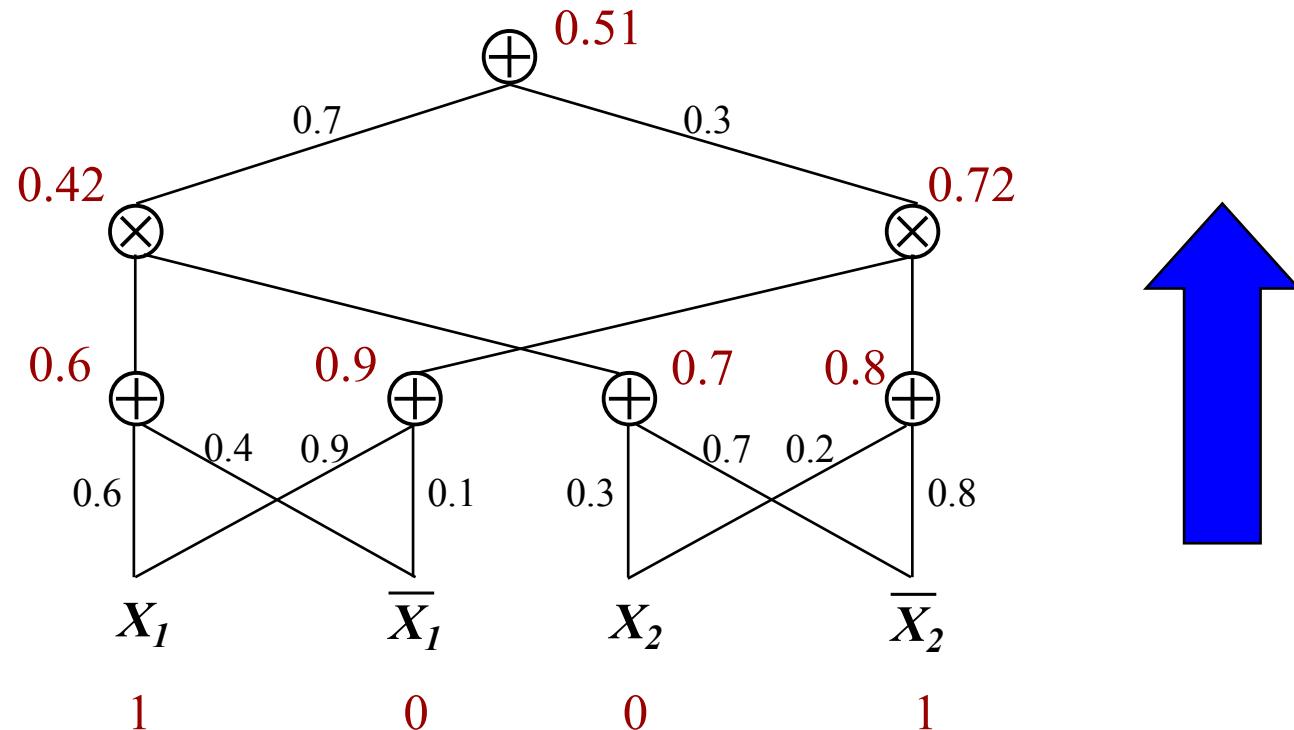
Inference: Linear in Size of Network

As long as weights sum to 1
at each sum node

$$P(X) = S(X)$$

$$X: X_1 = 1, X_2 = 0$$

X_1	1
\bar{X}_1	0
X_2	0
\bar{X}_2	1

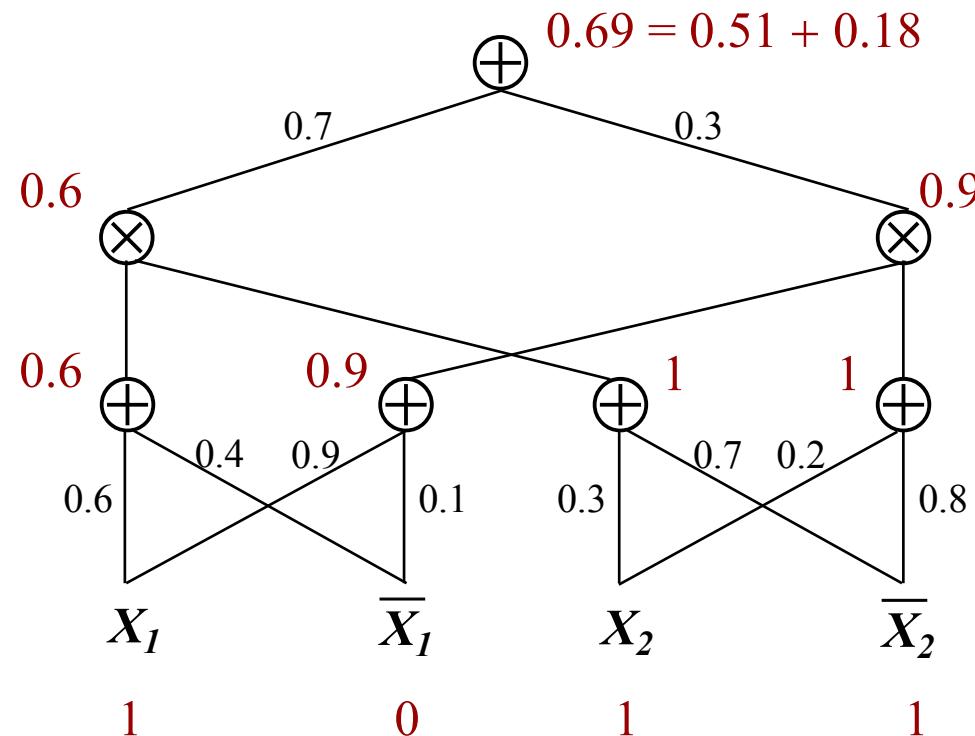


Inference: Linear in Size of Network

Marginal: $P(e) = S(e)$

$e: X_1 = 1$

X_1	1
\bar{X}_1	0
X_2	1
\bar{X}_2	1



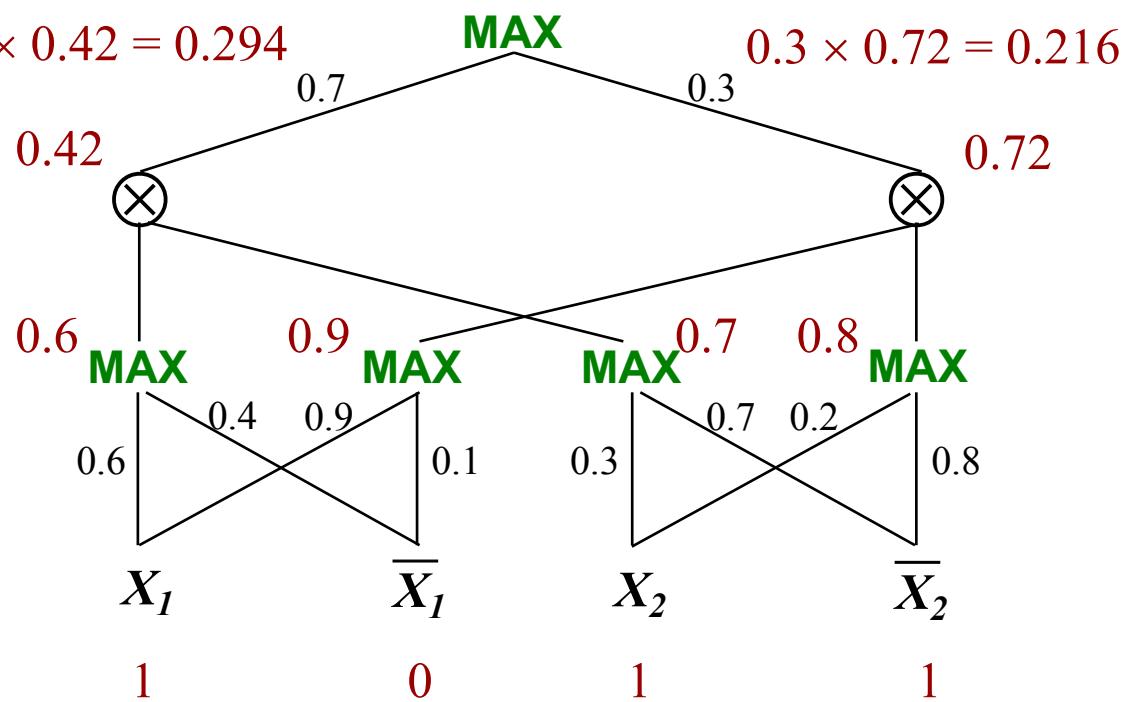
Inference: Linear in Size of Network

MAP: Replace sums with maxs

$$e: X_1 = 1$$

$$0.7 \times 0.42 = 0.294$$

X_1	1
\bar{X}_1	0
X_2	1
\bar{X}_2	1



Inference: Linear in Size of Network

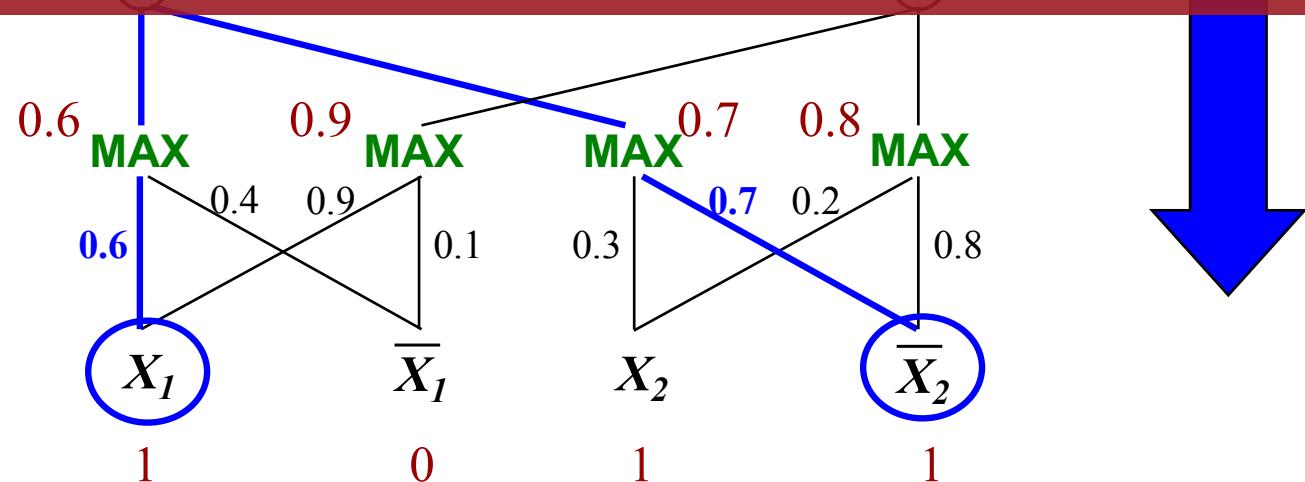
MAX: Pick child with highest value

MAP State: $X_1 = 1, X_2 = 0$

$e: X_1 = 1$

X_1	1
\bar{X}_1	0
X_2	1
\bar{X}_2	0
X_1	1

And also learning is conceptually easy



General Approach via Parameter Estimation assuming a fixed network (like in Deep Learning)

- Start with a dense SPN
- Find the structure by (online) learning weights
Zero weights signify absence of connections
- (Hard) EM beneficial to avoid gradient vanishing
Each sum node is a mixture over children

In principle you can turn a given SPN into a TensorFlow computation graph and apply any known algorithm from there



Image Completion

Main evaluation: Caltech-101 [Fei-Fei et al., 2004]

- 101 categories, e.g., faces, cars, elephants
- Each category: 30 – 800 images

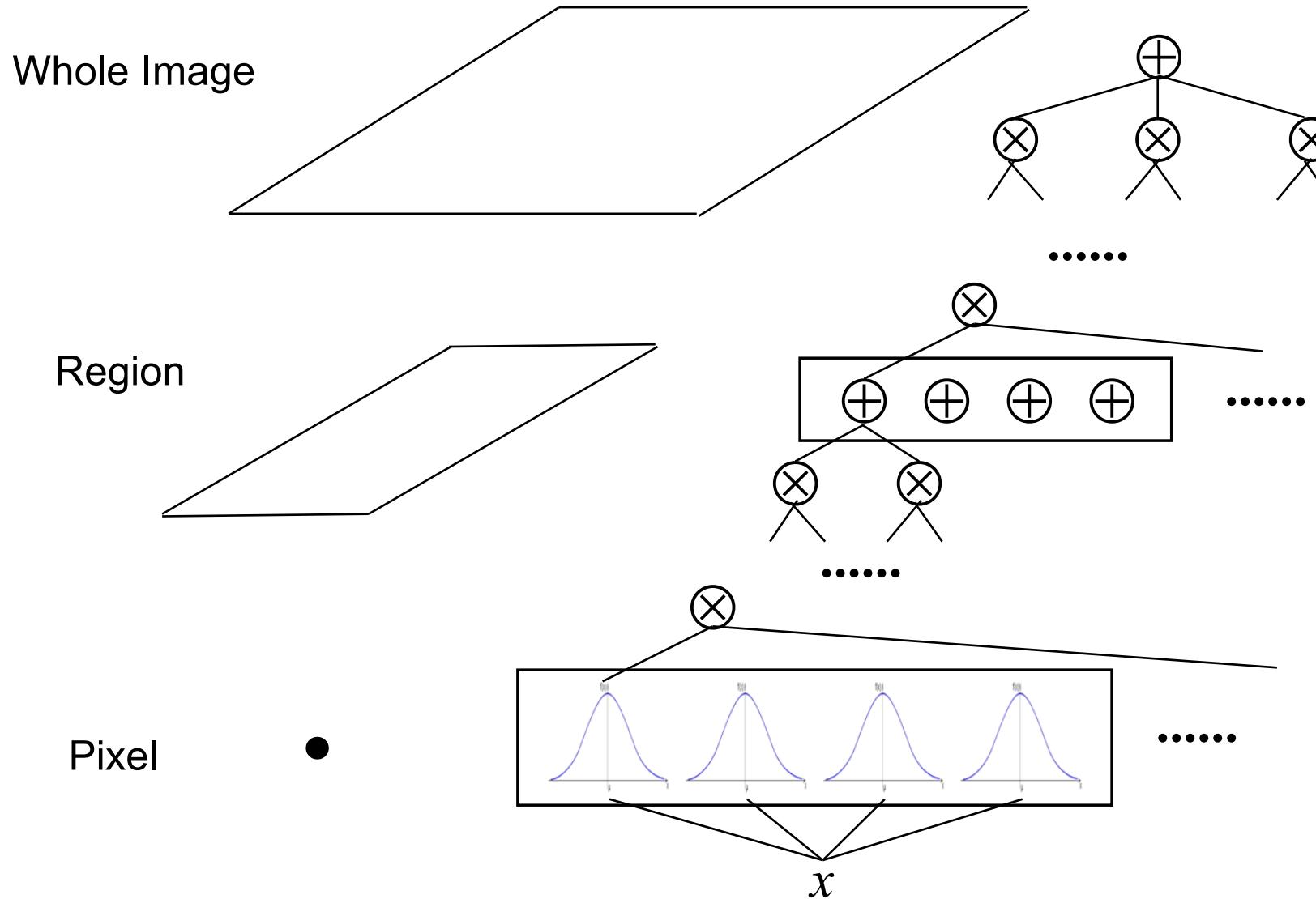
Also, Olivetti [Samaria & Harter, 1994] (400 faces)

Each category: Last third for test

Test images: Unseen objects



SPN Architecture



Caltech: Mean-Square Errors

	LEFT	BOTTOM
SPN	3551	3270
DBM	9043	9792
DBN	4778	4492
PCA	4234	4465
Nearest Neighbor	4887	5505



SPN vs. DBM / DBN

SPN is order of magnitude faster

	SPN	DBM / DBN
Learning	2-3 hours	Days
Inference	< 1 second	Minutes or hours

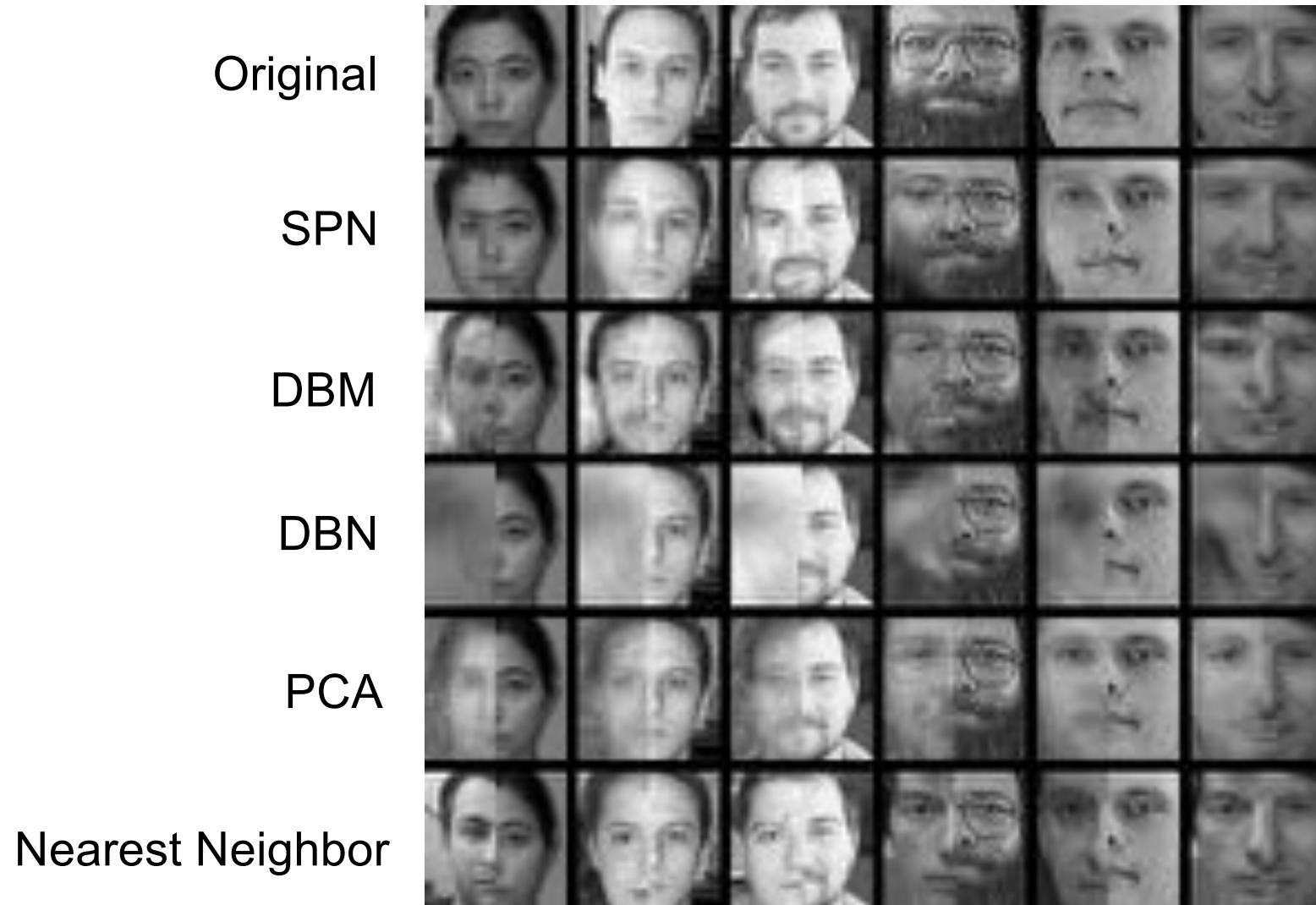
No elaborate preprocessing, tuning

Reduced errors by 30-60%

Learned up to 46 layers



Example Completions



Random sum-product networks

[Peharz, Vergari, Molina, Stelzner, Trapp, Kersting, Ghahramani UAI 2019]



UNIVERSITY OF
CAMBRIDGE



Max Planck Institute for
Intelligent Systems

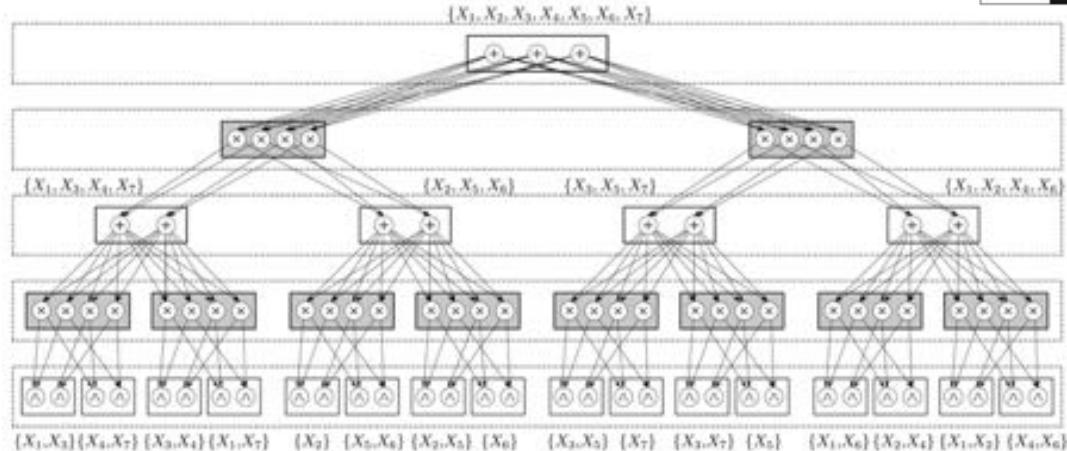


TECHNISCHE
UNIVERSITÄT
DARMSTADT

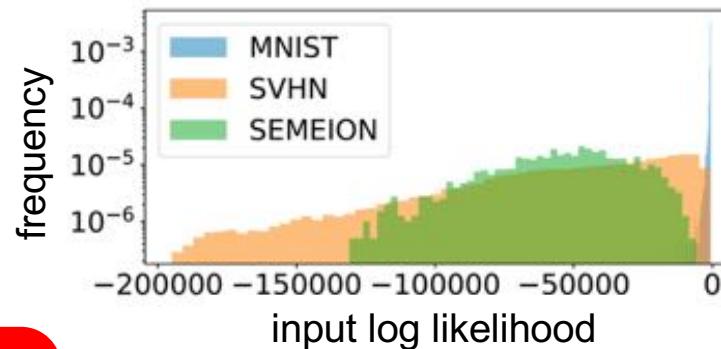


Conference on Uncertainty in Artificial Intelligence
Tel Aviv, Israel
July 22 - 25, 2019

uai2019



	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)



SPNs can have similar predictive performances as (simple) DNNs

SPNs can distinguish the datasets

SPNs know when they do not know by design



Einsum Networks

[Peharz, Lang, Vergari, Stelzner, Molina, Trapp, Van den Broeck, Kersting, Ghahramani ICML 2020]



TU/e



TECHNISCHE
UNIVERSITÄT
DARMSTADT

UBER AI Labs

UCLA



ICML | 2020

Thirty-seventh International
Conference on Machine Learning



(a) Real SVHN images.



(b) EiNet SVHN samples.



(c) Real images (top), covered images, and EiNet reconstructions



(d) Real Celeba samples.

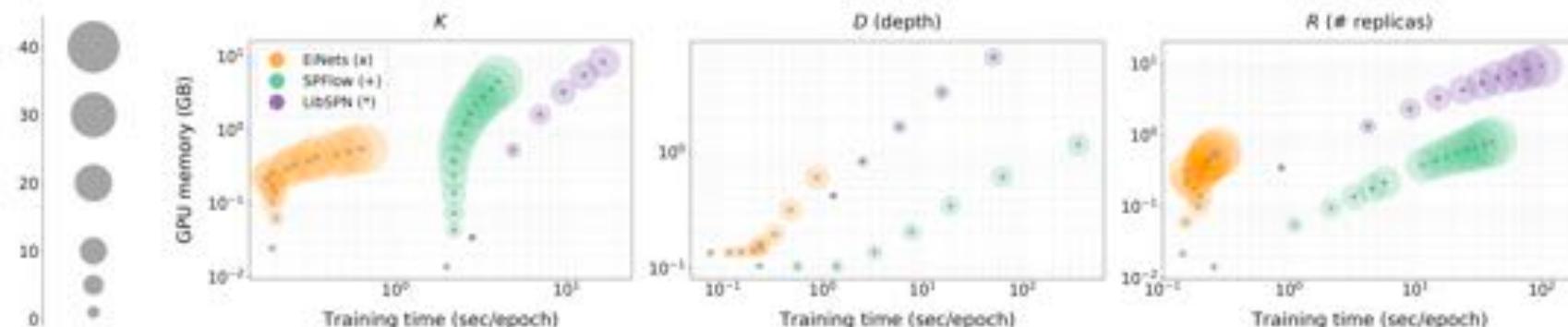
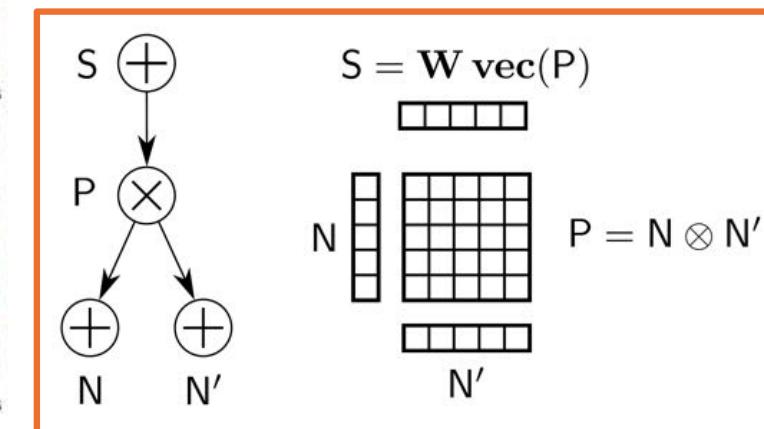


(e) EiNet Celeba samples.



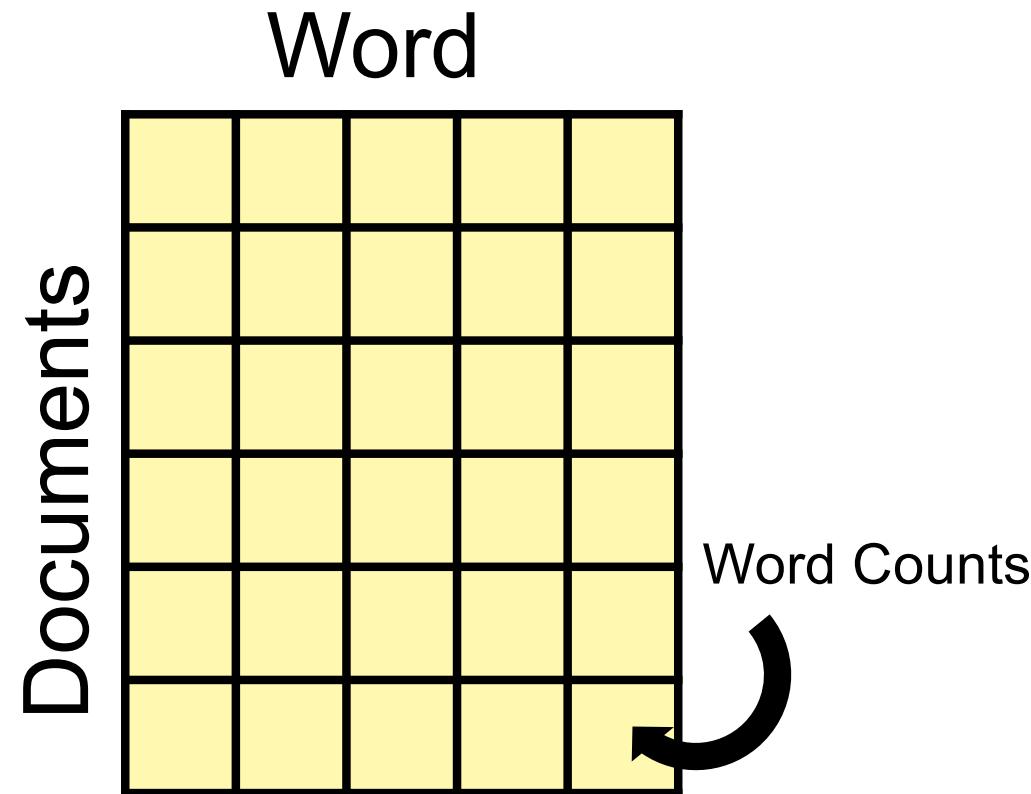
(f) Real images (top), covered images, and EiNet reconstructions

**Compilation of PCs into
efficient to evaluate
computational graphs**



Or we learn directly (Tree-)SPNs

Testing independence of
Poisson random variables

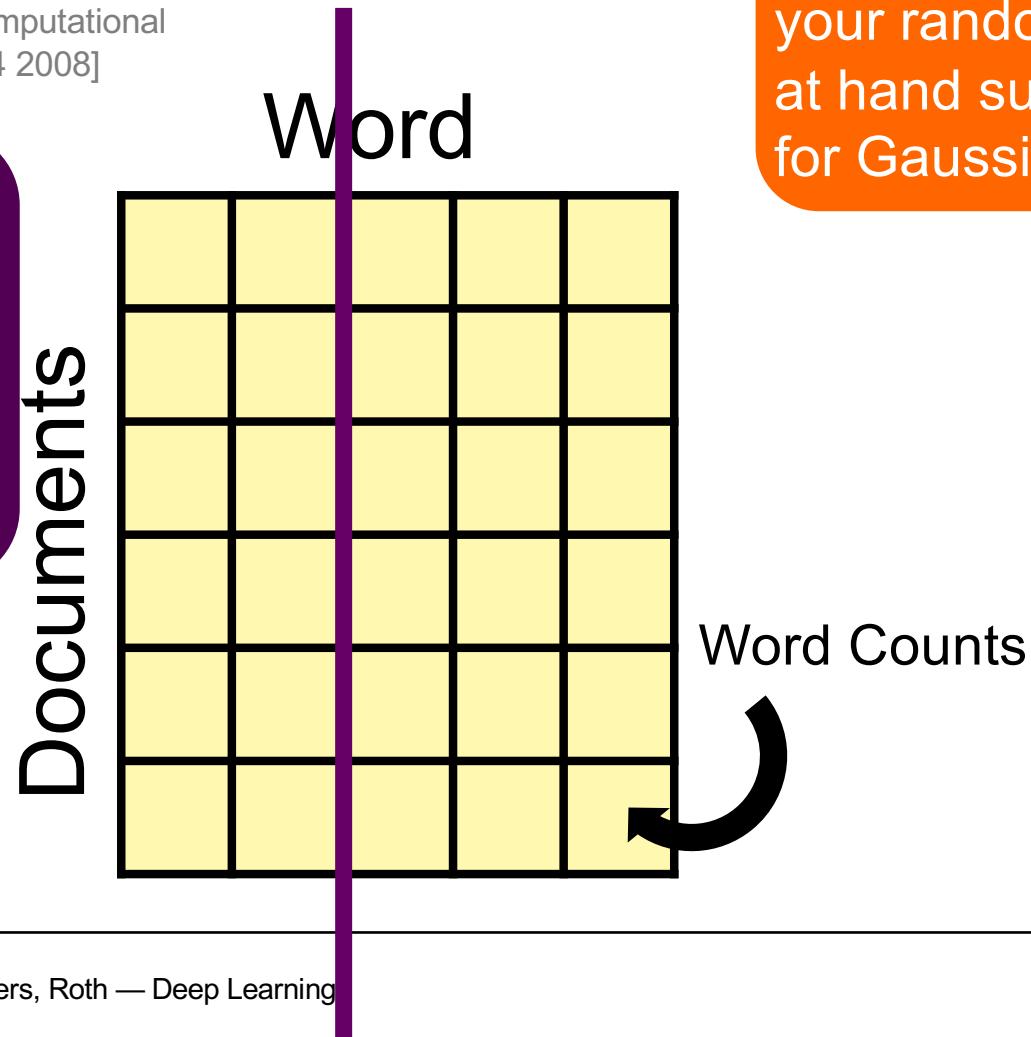


Or we learn directly (Tree-)SPNs

Testing independence of Poisson random variables

[Zeileis, Hothorn, Hornik Journal of Computational And Graphical Statistics 17(2):492–514 2008]

Learn Poisson model trees for $P(x|V-x)$ and $P(y|V-y)$. Check whether X resp. Y is significant in $P(y|V-x)$ resp. $P(x|V-y)$



In general use the independency test for your random variables at hand such as g-test for Gaussians

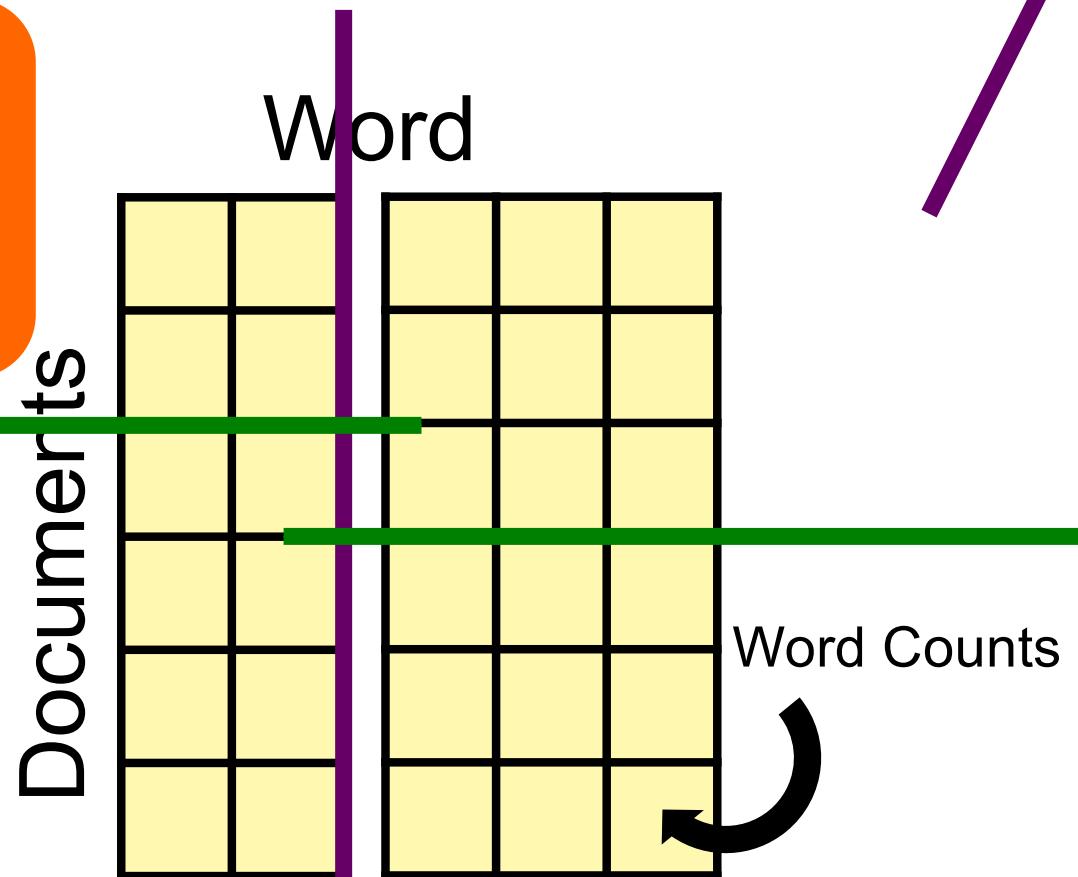


Or we learn directly (Tree-)SPNs

Testing independence of
Poisson random variables

In general some
clustering for your
random variables at
hand such as kMeans
for Gaussians

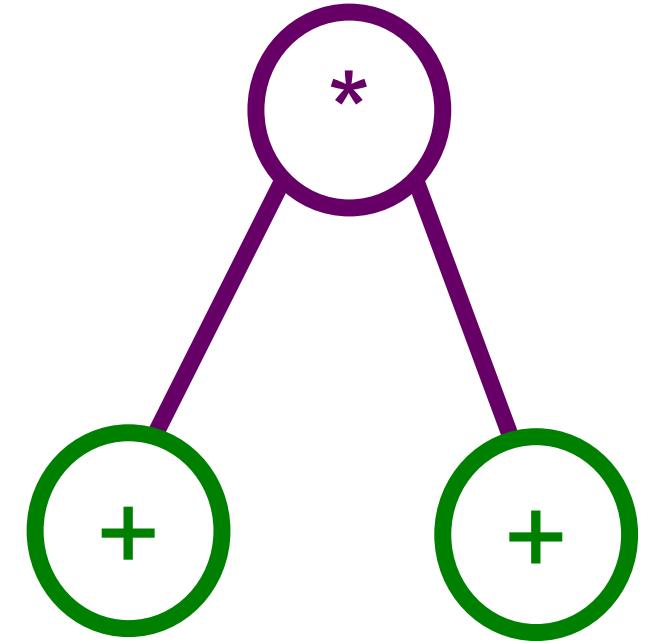
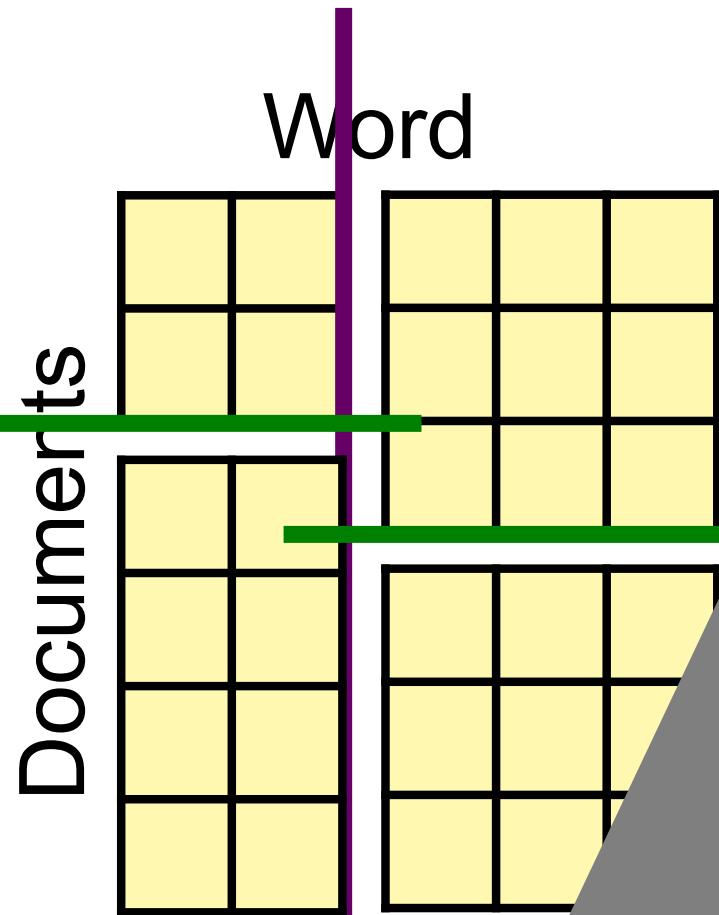
Mixture of
Poisson
Dependency
Networks or
random splits



Or we learn directly (Tree-)SPNs

Testing independence of
Poisson random variables

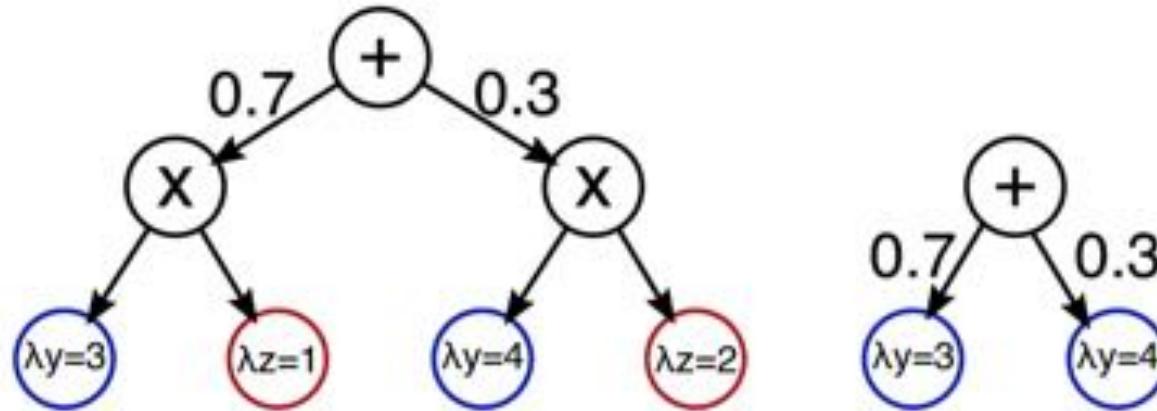
Mixture of
Poisson
Dependency
Networks or
random splits



keep growing
alternatingly
and + layers

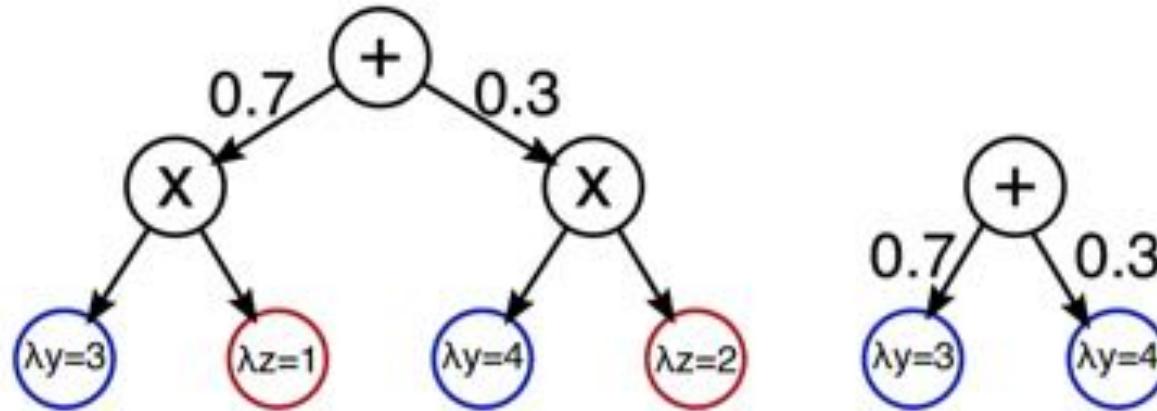


Tractable Inference via Symbolic Evaluation



SPNs encode polynomials over univariate distributions to build multivariate distributions. They can be fed into symbolic evaluation methods for inference

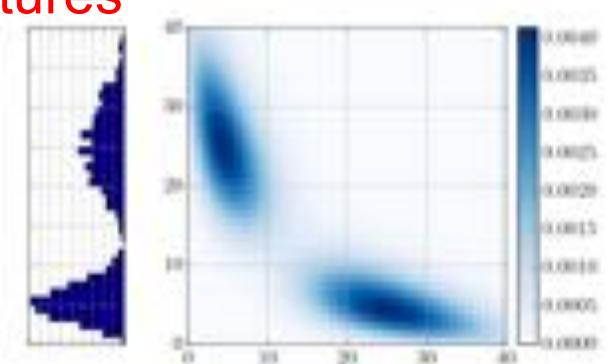
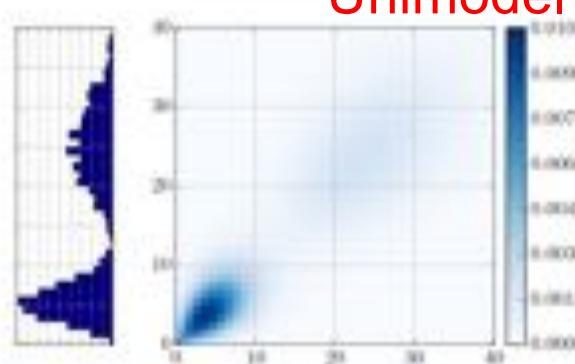
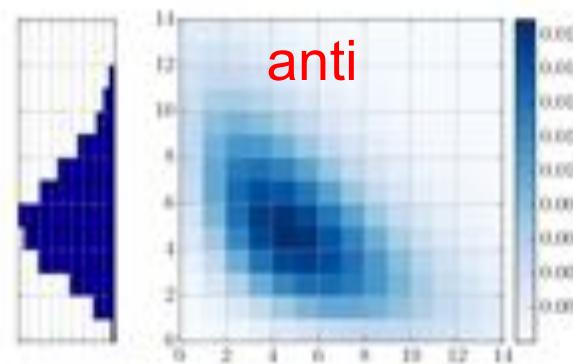
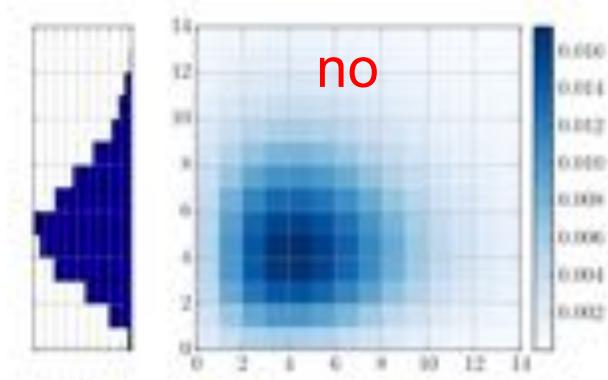
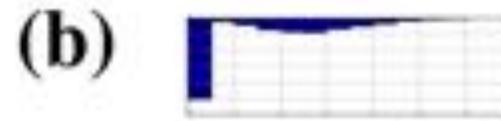
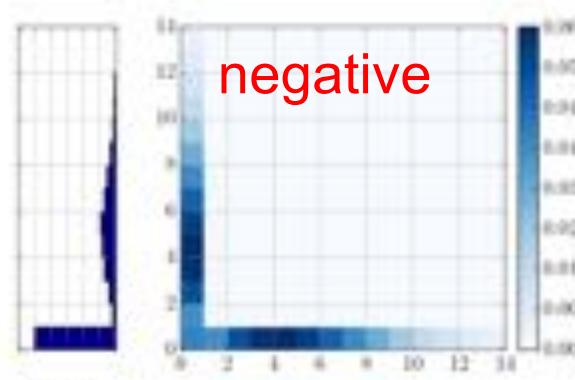
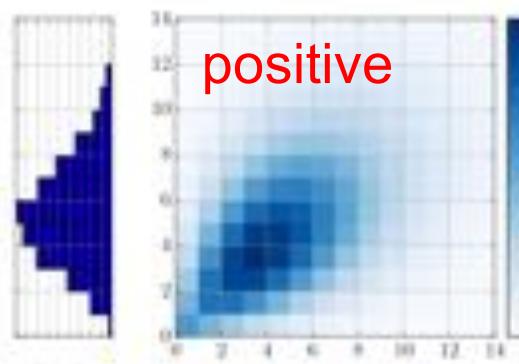
Inference on Devices



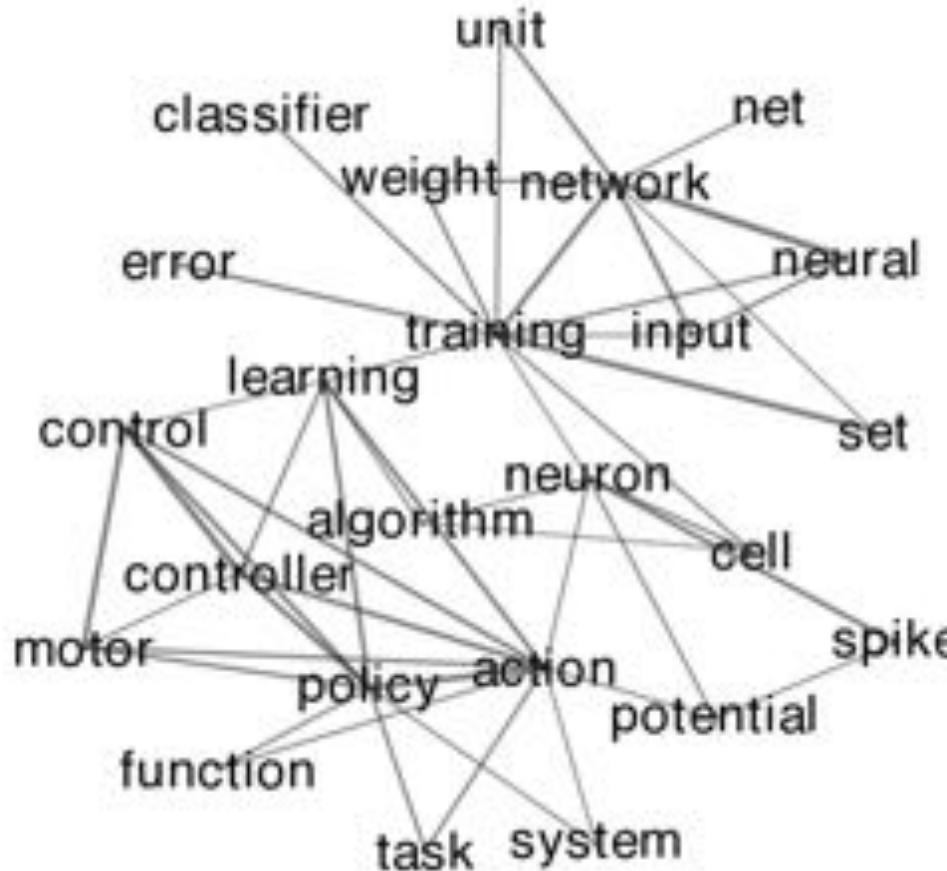
Ultimately, they may even be compiled into simple, flat, library-free code suitable for embedding in real-time applications and devices

SPNs encode polynomials over univariate distributions to build multivariate distributions. They can be fed into symbolic evaluation methods for inference

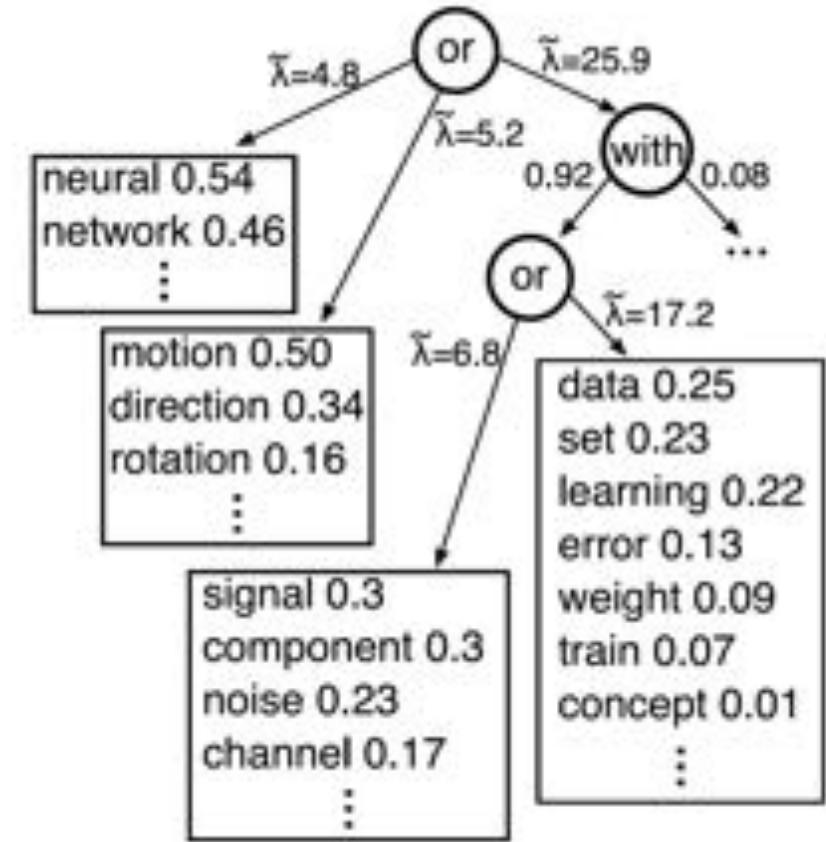
Flexible Dependencies even for challenging distributions, here multivariate Poisson distribution



Multiple views on your data due to efficient inference



Mutual Information
(NIPS corpus)



Poisson Multinomial SPN
= hierarchical topic model



Poisson SPNs provide topics: From Topic Models to Poisson Mean Fields

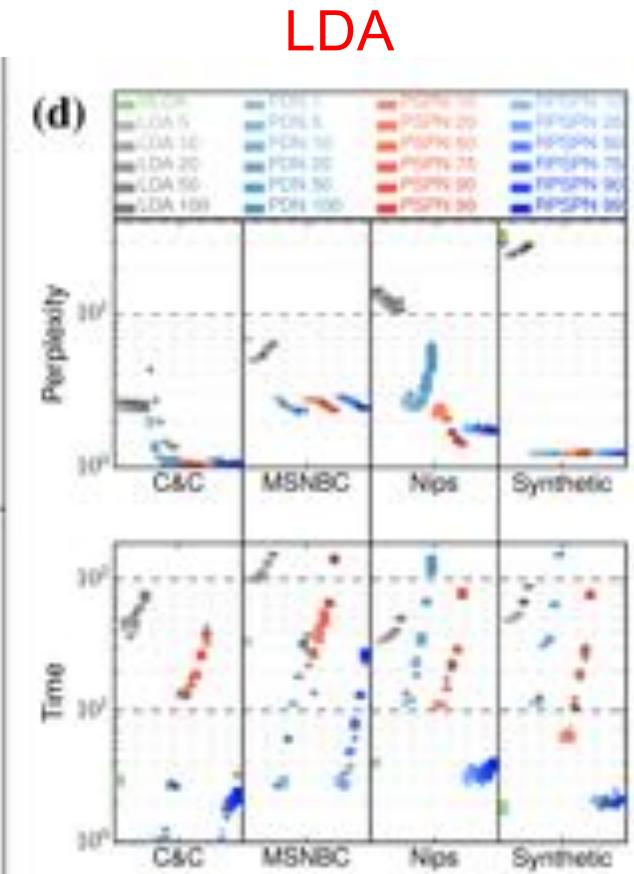
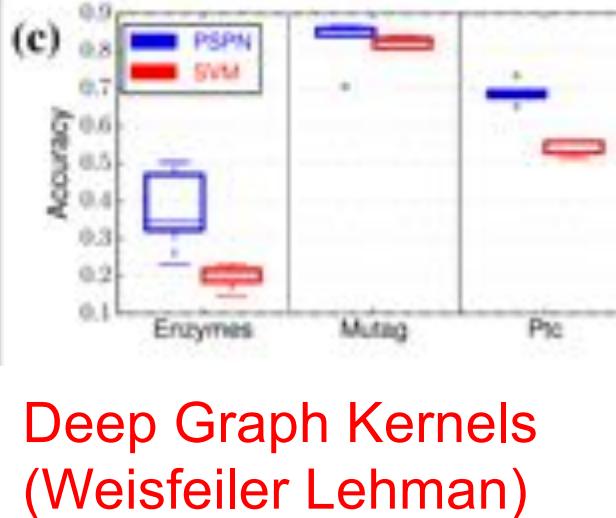
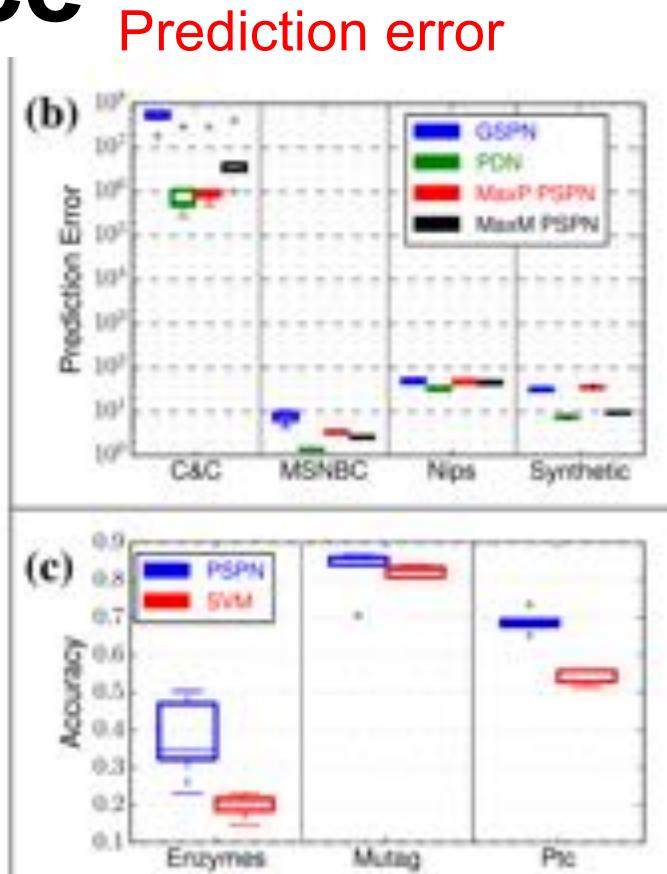
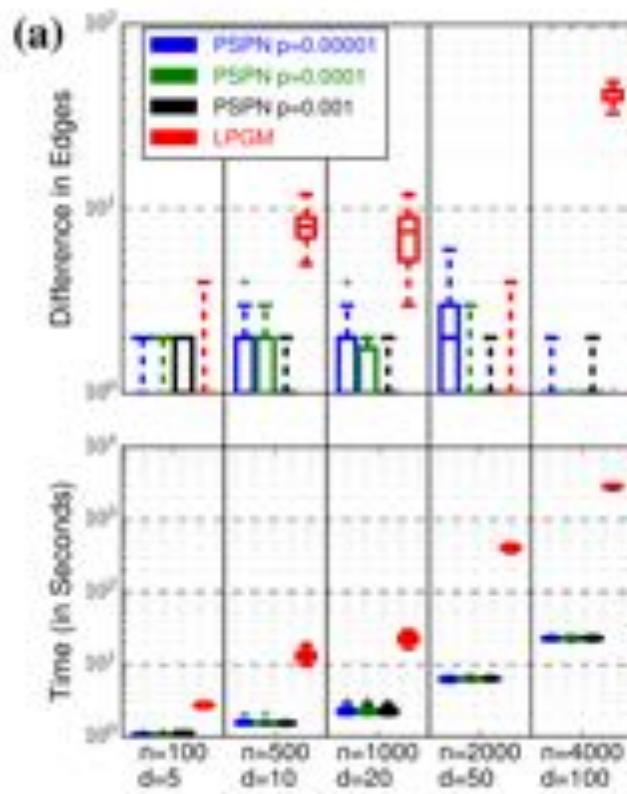
[Bishop, Fienberg, Holland. Discrete Multivariate Analysis: Theory and Practice. Springer, 2007]

Topic-word distributions of LDA can be viewed as a product of independent Poissons

$$\begin{aligned}
 & \Pr_{\text{Poiss}} \left(\tilde{x} \mid \tilde{\lambda} \right) \Pr_{\text{Mult}} \left(\mathbf{x} \mid \theta = (\lambda_1, \dots, \lambda_p) / \tilde{\lambda}, \mathbf{N} = \tilde{x} \right) \\
 &= \frac{e^{-\tilde{\lambda}}}{\tilde{x}!} \tilde{\lambda}^{\tilde{x}} \frac{\tilde{x}!}{\prod_{s=1}^p x_s!} \prod_{s=1}^p \left(\frac{\lambda_s}{\tilde{\lambda}} \right)^{x_s} \\
 &= \frac{\tilde{x}!}{\tilde{x}!} \frac{e^{-\tilde{\lambda}}}{\prod_{s=1}^p x_s!} \prod_{s=1}^p \left(\frac{\tilde{\lambda} \lambda_s}{\tilde{\lambda}} \right)^{x_s} \\
 &= \Pr_{\text{Ind. Poiss}} \left(\mathbf{x} \mid \lambda_1, \dots, \lambda_p \right) = \prod_{s=1}^p \frac{e^{-\lambda_s}}{x_s!} \lambda_s^{x_s}
 \end{aligned}$$



Competitive Predictive Performance



[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]

FL⁺ SPFlow: An Easy and Extensible Library ⊗W for Sum-Product Networks



UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO



Max Planck Institute for
Intelligent Systems



UNIVERSITY OF
CAMBRIDGE



VECTOR
INSTITUTE

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



Federal Ministry
of Education
and Research

195 commits

2 branches

0 releases

All 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 import Categorical
from spn.structure.Base import Sum, Product
from spn.structure.base import assign_ids, rebuild_scopes_bottom_up

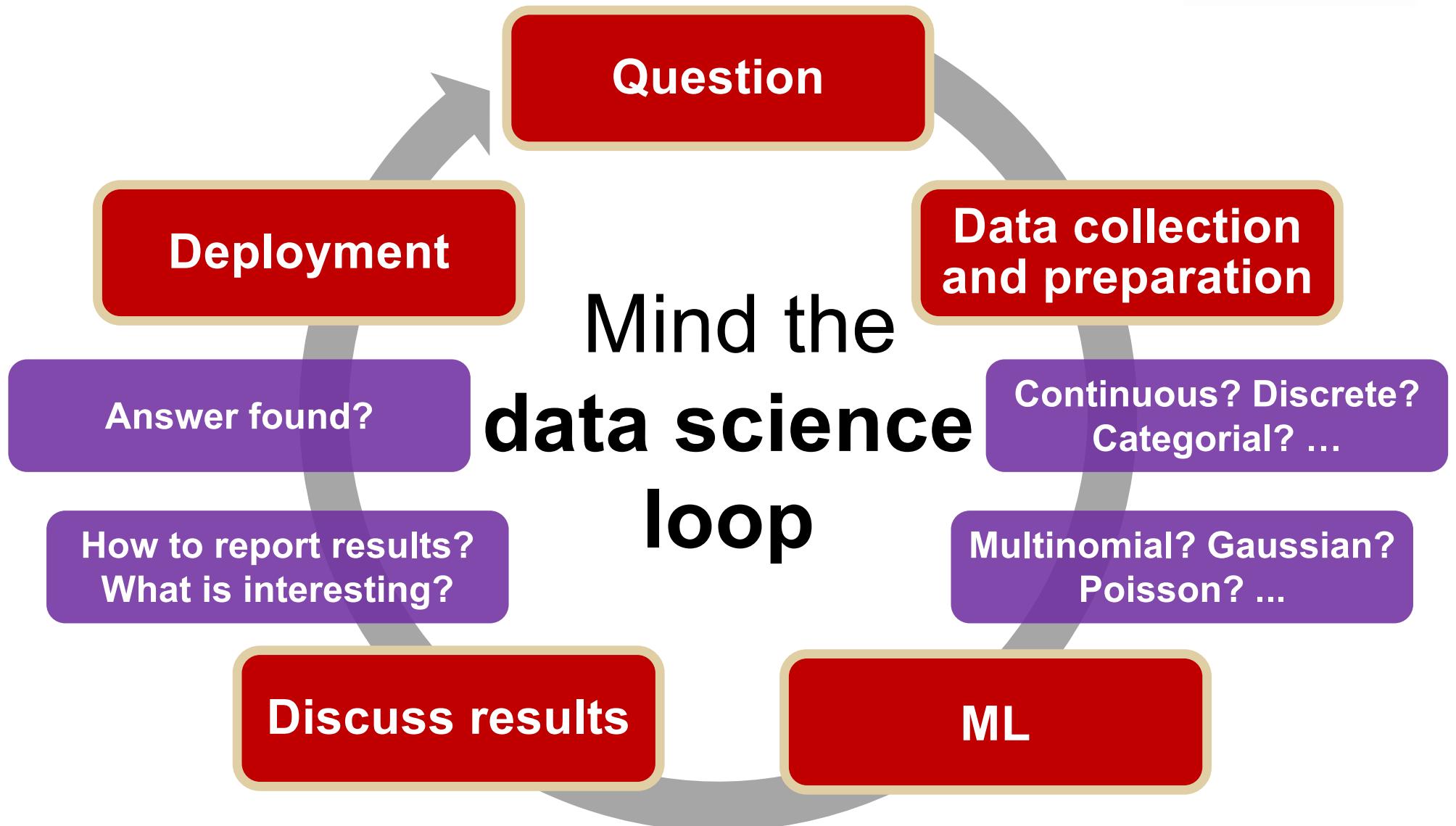
p0 = Product(children=[Categorical(p=[0.3, 0.7], scope=1), Categorical(p=[0.4, 0.6], scope=2)])
p1 = Product(children=[Categorical(p=[0.5, 0.5], scope=1), Categorical(p=[0.6, 0.4], scope=2)])
s1 = Sum(weights=[0.3, 0.7], children=[p0, p1])
p2 = Product(children=[Categorical(p=[0.2, 0.8], scope=0), s1])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

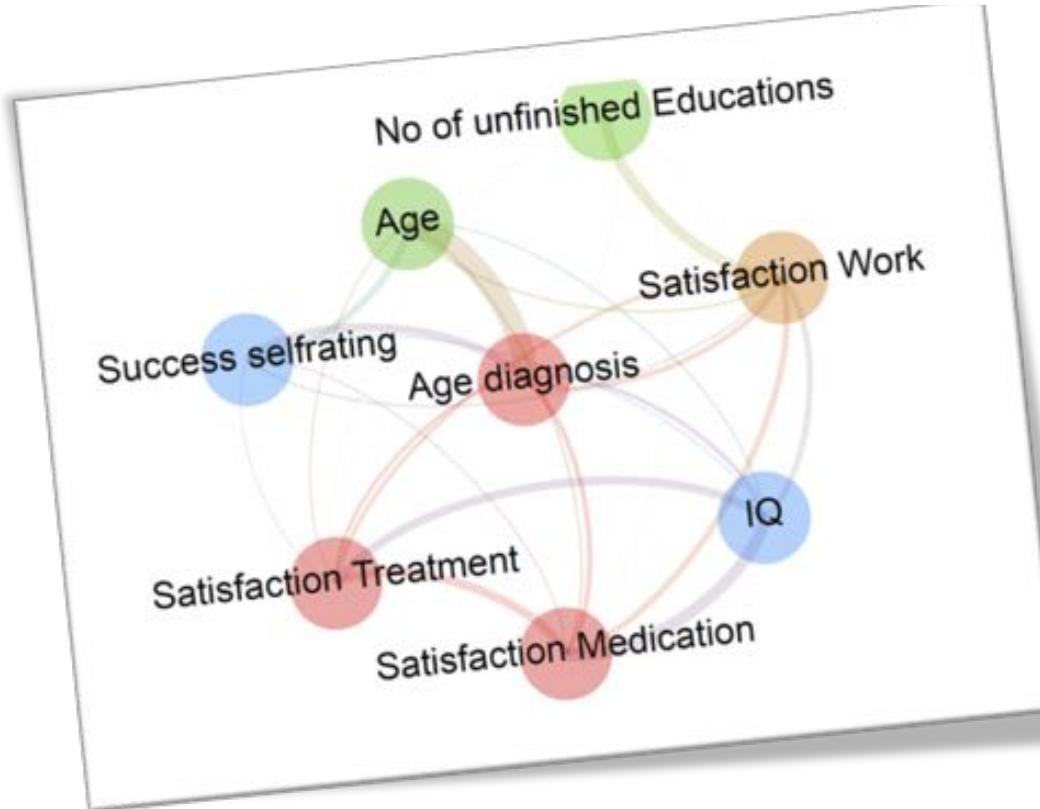
return spn
```

**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**

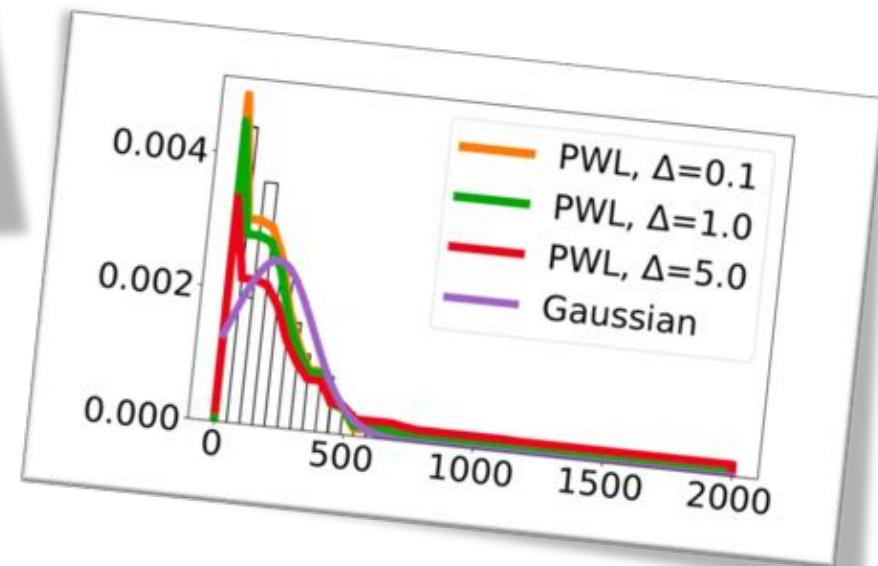
SPFlow, an open-source Python library providing a simple interface to inference, learning and manipulation routines for deep and tractable probabilistic models called Sum-Product Networks (SPNs). The library allows one to quickly create SPNs both from data and through a domain specific language (DSL). It efficiently implements several probabilistic inference engines like message-passing, conditionals and incremental most probable explanations (IMEs) along with common



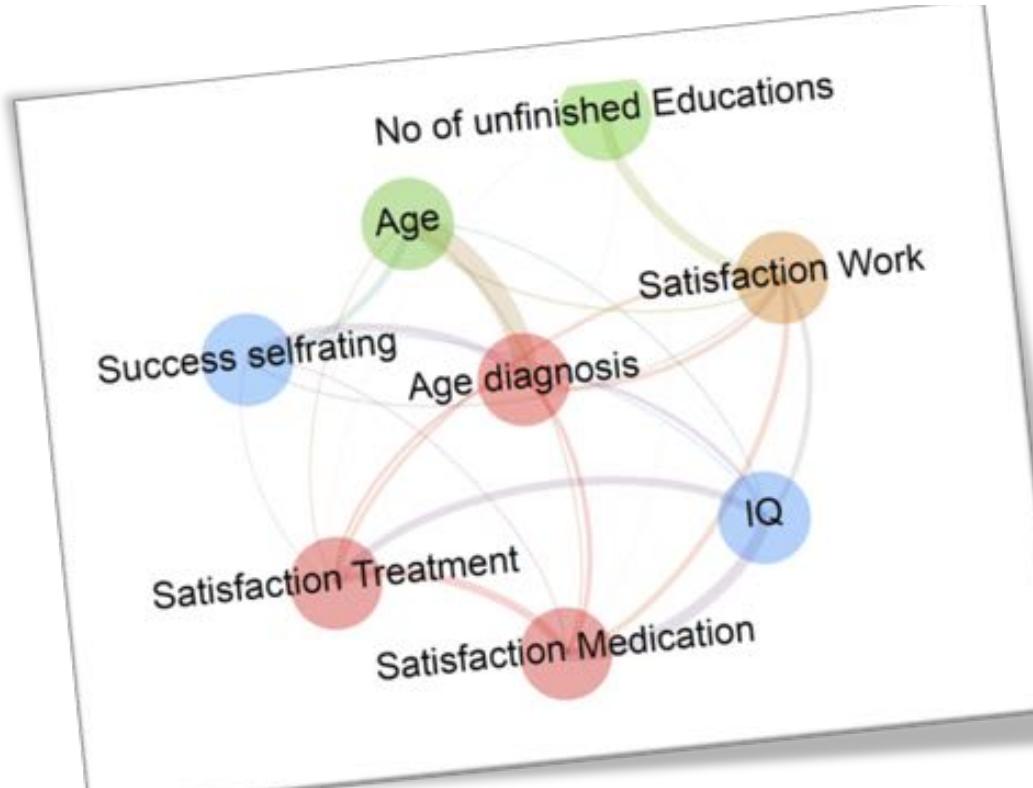
Distribution-agnostic PC



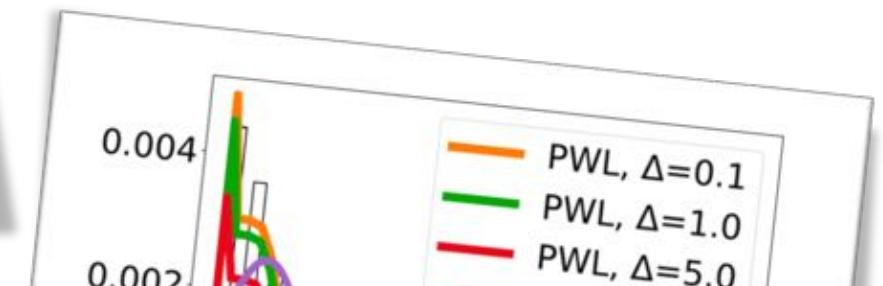
Use nonparametric
independency tests
and piece-wise linear
approximations



Distribution-agnostic PC



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? ...

The Explorative Automatic Statistician



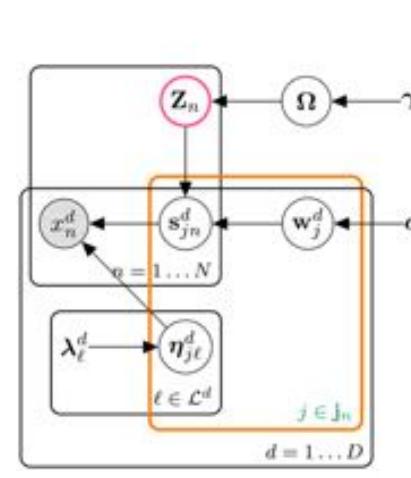
UNIVERSITY OF
CAMBRIDGE



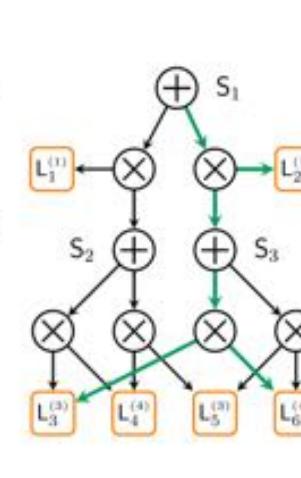
TECHNISCHE
UNIVERSITÄT
DARMSTADT

	X^1	X^2	X^3	X^4	X^5
x_8					
x_7			?		
x_6					
missing value	x_5	?			
x_4			?		
x_3					
x_2		?			
x_1					

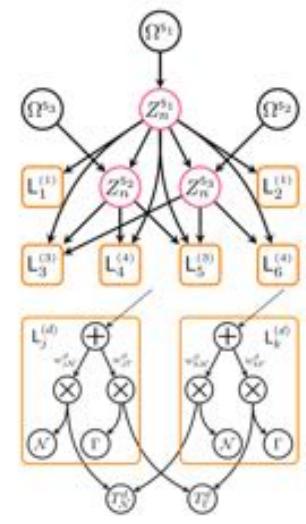
We can even automatically discovers the statistical types and parametric forms of the variables



Bayesian Type Discovery



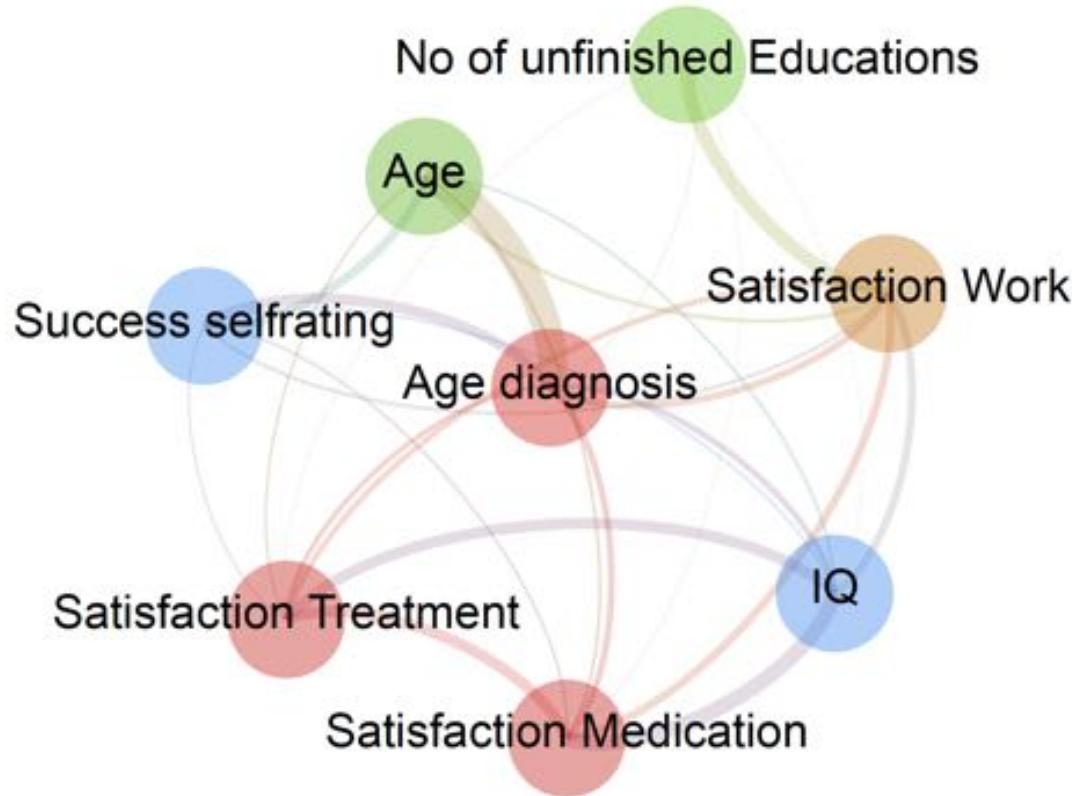
Mixed Sum-Product Network



Automatic Statistician

Towards Automated Statisticians:

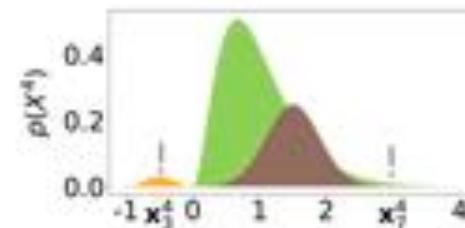
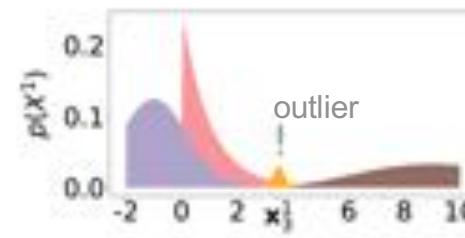
make use of nonparametric (in)dependency tests and distributions to build nonparametric SPNs that are agnostic to the statistical type of the random variables



Visualization the gist of a Mixed SPN (learned on the hybrid Autism dataset) using normalized mutual information (the thicker, the higher). **The machine learning tool does not know anything about the data types.** Node colors encode different feature groups: Demographics (green), Psychological (blue), Social Environment (orange) and Medical (red).



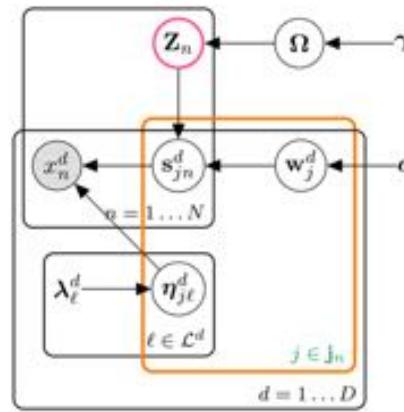
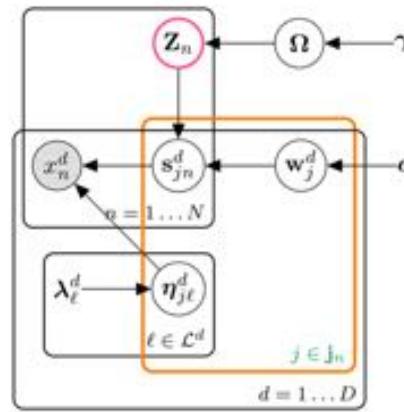
The Automatic Data Scientist



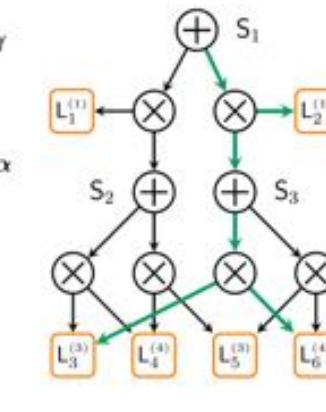
Exponential (Exp): 25.00%
Gaussian (\mathcal{N}): 37.50%
Gamma (Γ): 25.00%
Gaussian (\mathcal{N}): 12.50%

Gamma (Γ): 62.50%
Gaussian (\mathcal{N}): 12.50%
Gamma (Γ): 25.00%

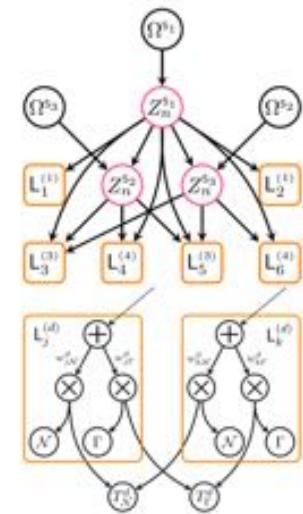
We can even automatically discovers the statistical types and parametric forms of the variables



Bayesian Type Discovery



Mixed Sum-Product Network



Automatic Statistician

That is, the machine understands the data with few expert input ...

The screenshot shows a user interface for a DeepNotebook. At the top, there are three buttons: 'Toggle Introduction', 'Toggle explanations', and 'Toggle Code'. Below these, the title 'Exploring the Titanic dataset' is displayed in a large, bold font. A detailed description of the dataset follows:

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 analyzed. 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



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Report framework created @ TU Darmstadt

...and can compile data reports automatically

Overall we need languages for Systems AI,

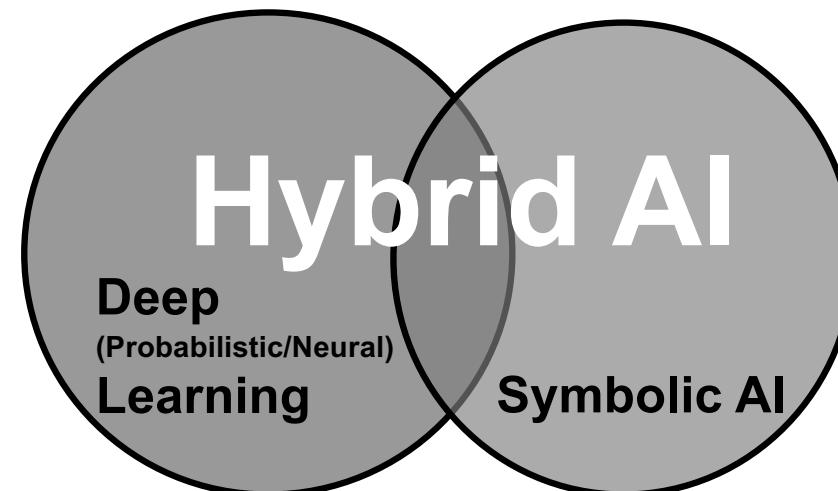
the computational and mathematical modeling of complex AI systems.

[Kordjamshidi, Roth, Kersting: “Systems AI: A Declarative Learning Based Programming Perspective.” IJCAI-ECAI 2018] following the motivation of Christopher Re



The next breakthrough in AI may not just be a new ML/AI algorithm...

...but may be in the ability to rapidly combine, deploy, and maintain existing AI algorithms



Hybrid AI: Unsupervised scene understanding

[Stelzner, Peharz, Kersting ICML 2019, Best Paper Award at TPM@ICML2019] <https://github.com/stelzner/supair>

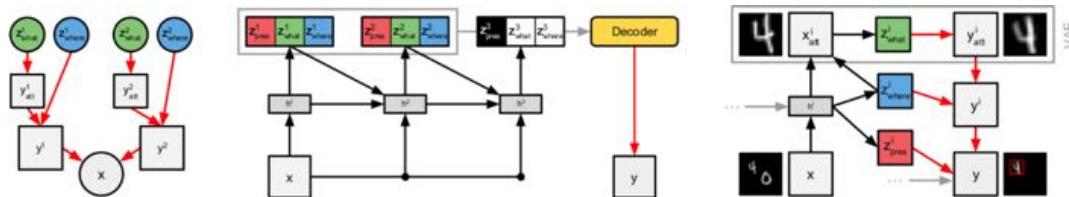
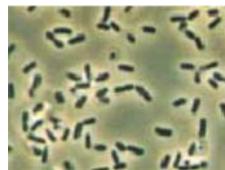


TECHNISCHE
UNIVERSITÄT
DARMSTADT

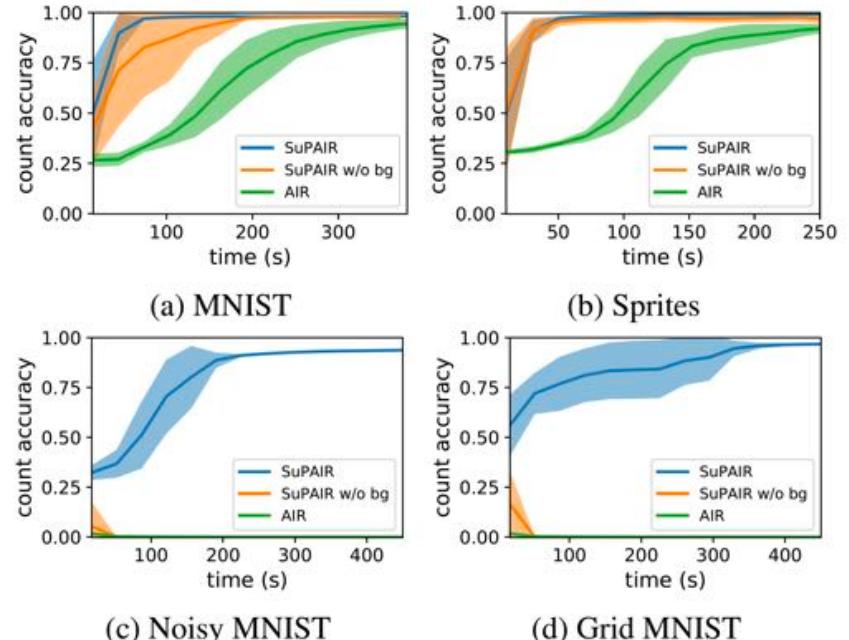
ICML | 2019

Thirty-sixth International Conference on
Machine Learning

Consider e.g. unsupervised scene
understanding using a generative model
implemented in a neural fashion



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



(a) MNIST

(b) Sprites

(c) Noisy MNIST

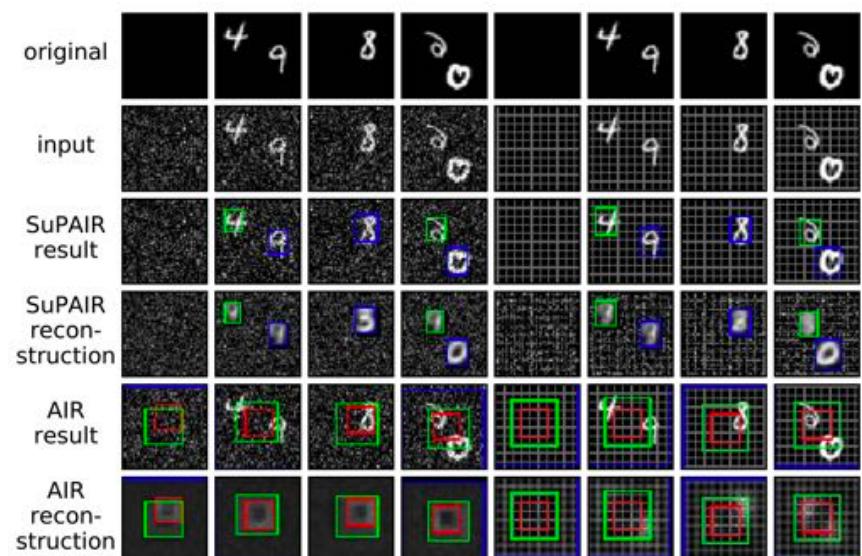
(d) Grid MNIST



Replace VAE
by SPN as
object model



- 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]



Hybrid AI: Unsupervised physics learning

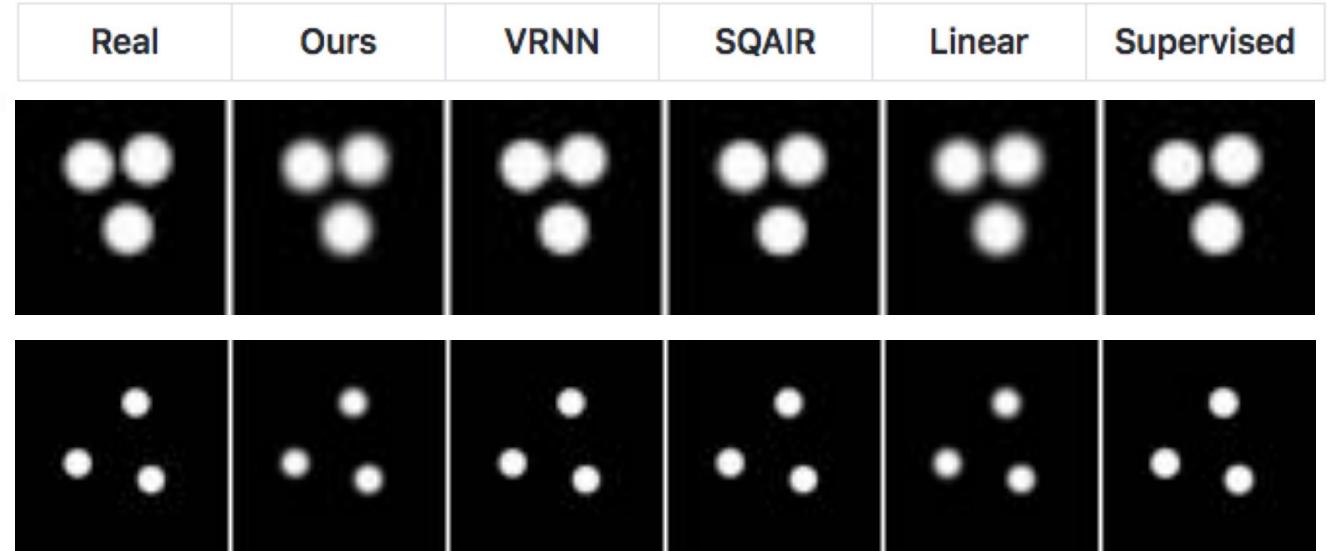
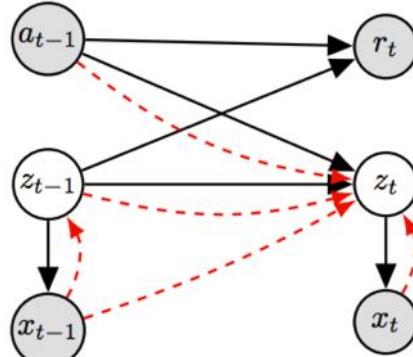
[Kossa, Stelzner, Hussing, Voelcker, Kersting ICLR 2020]

ICLR | 2020

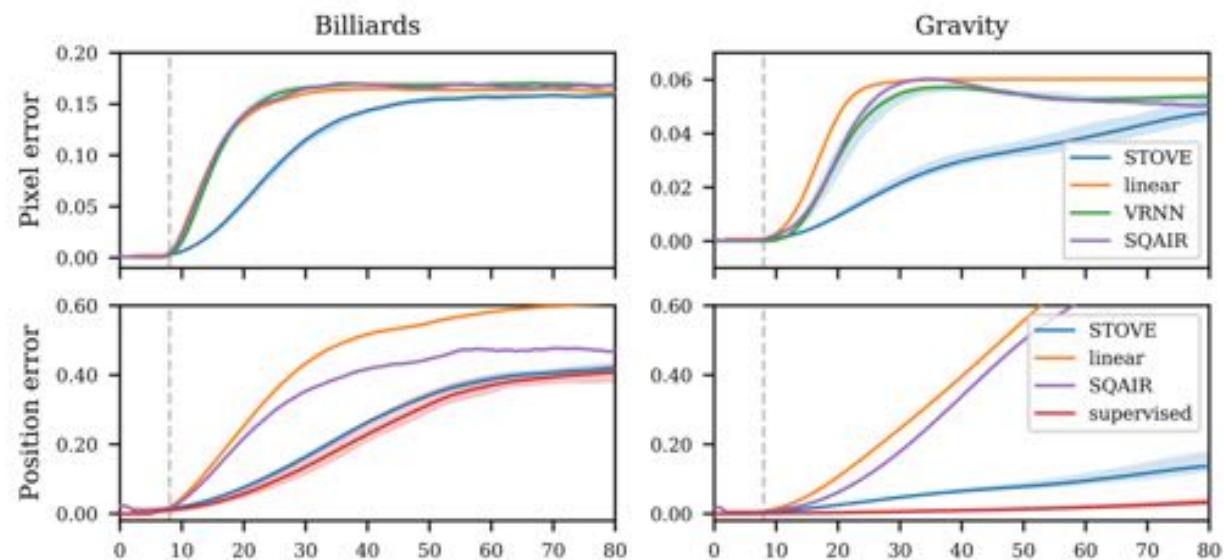
Eighth International Conference on
Learning Representations



TECHNISCHE
UNIVERSITÄT
DARMSTADT

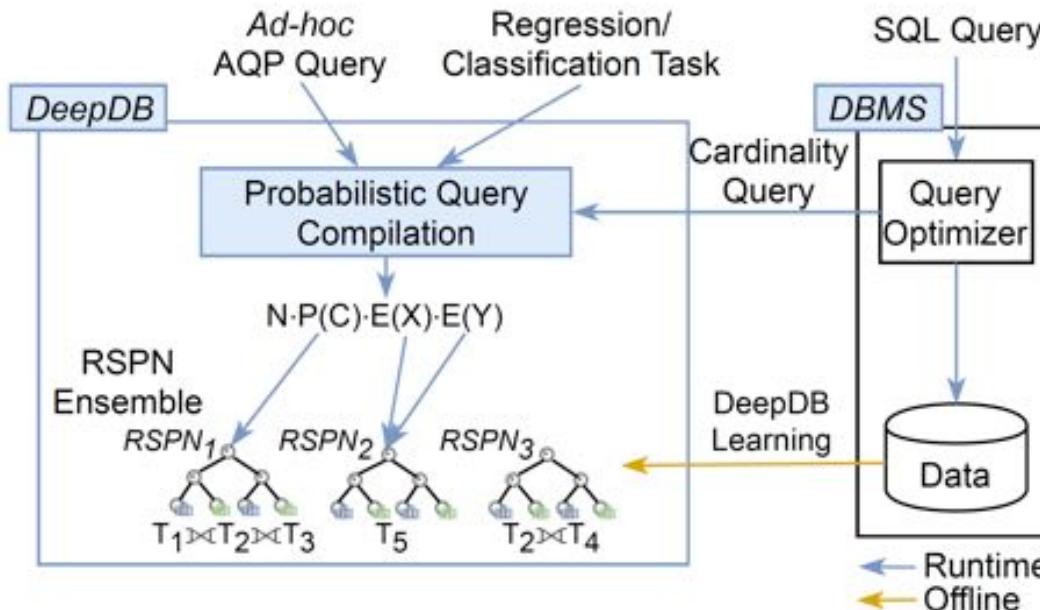


putting
structure and
tractable
inference into
deep models

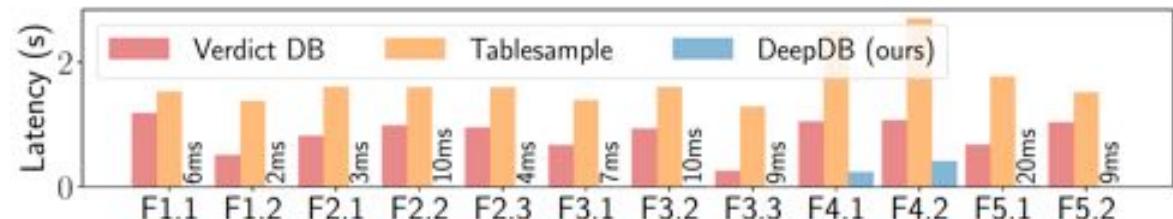
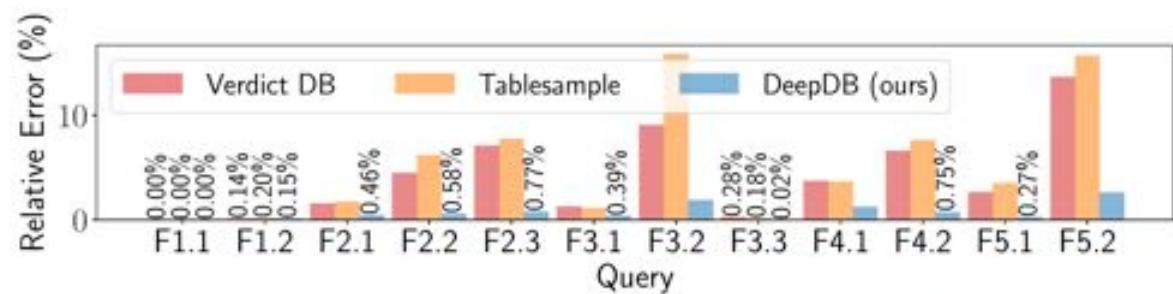


Hybrid AI: Deep Databases

[Hilprecht, Schmidt, Kulessa, Molina, Kersting, Binnig VLDB 2020]



46TH INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES



What have we learnt about SPNs?



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Sum-product networks (SPNs)

- DAG of sums and products
- They are instances of Arithmetic Circuits (ACs)
- Compactly represent partition function
- Learn many layers of hidden variables

Efficient marginal inference

Easy learning

Can outperform well-known alternatives

