# Sum-Product Networks: A New Deep Architecture

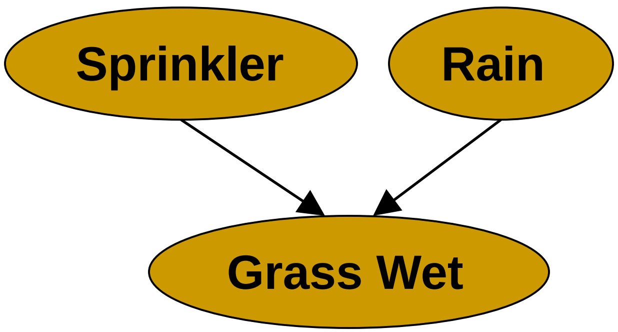
**Hoifung Poon**

Microsoft Research

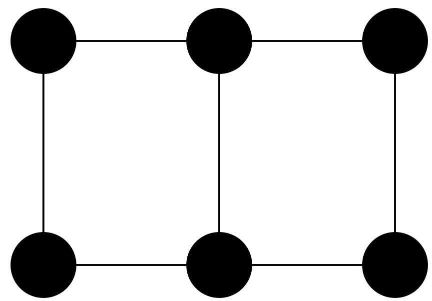
*Joint work with Pedro Domingos*

# Graphical Models: Challenges

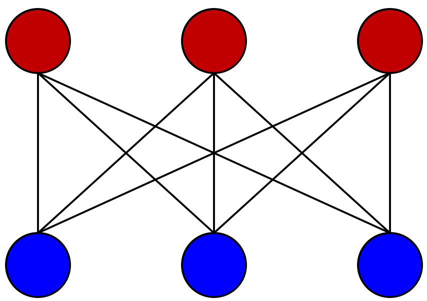
Bayesian Network




Markov Network



Restricted Boltzmann Machine (RBM)



**Advantage: Compactly represent probability**

## Problem: Inference is intractable

## Problem: Learning is difficult

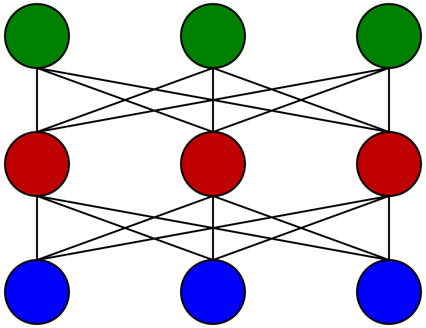
# Deep Learning

- Stack many layers

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

  CDBN [Lee et al., 2009]

  DBM [Salakhutdinov & Hinton, 2010]

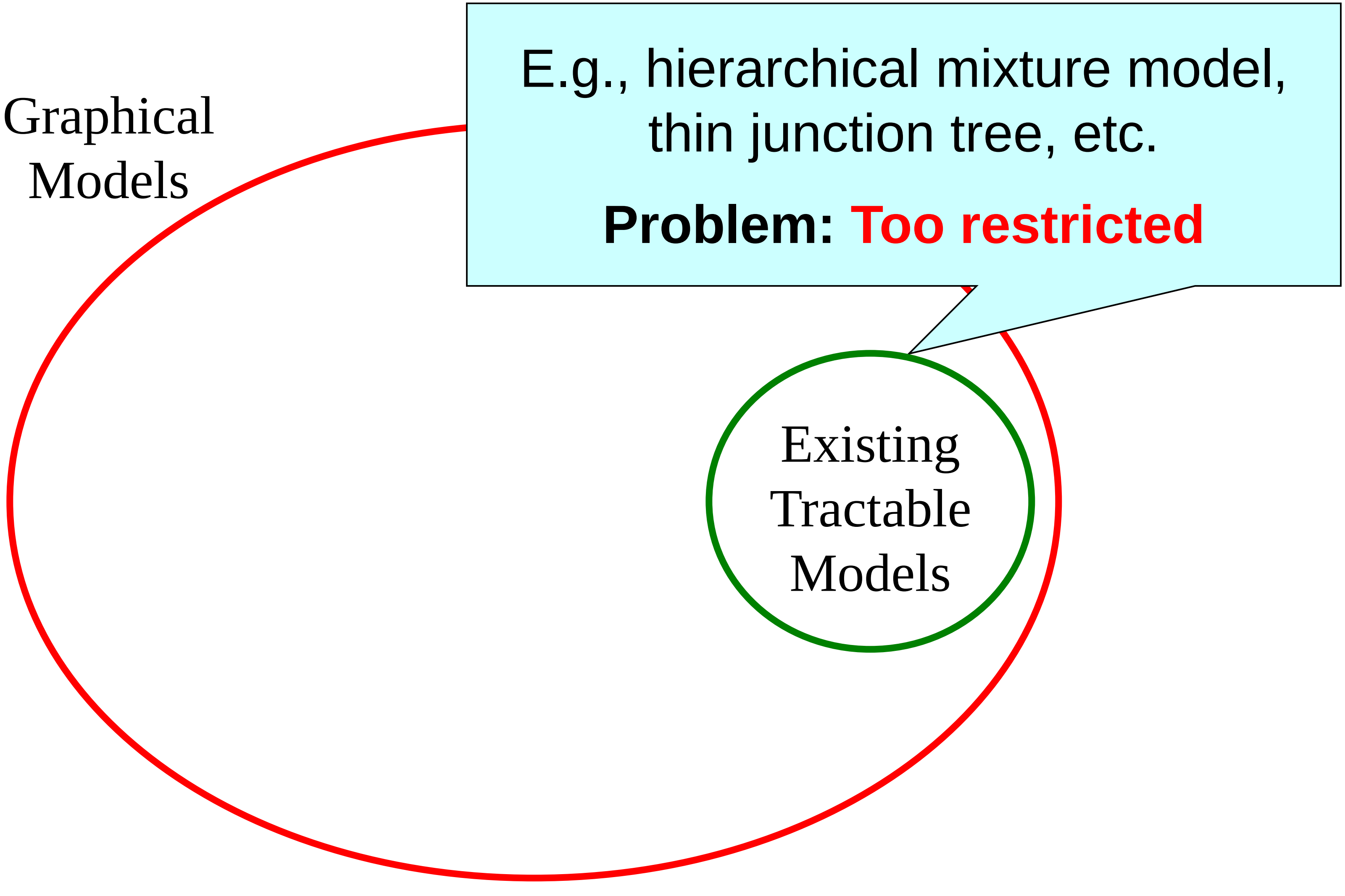


- **Potentially much more powerful than shallow architectures** [Bengio, 2009]
- But …
  - **Inference is even harder**
  - **Learning requires extensive effort**

**Learning:** Requires approximate inference

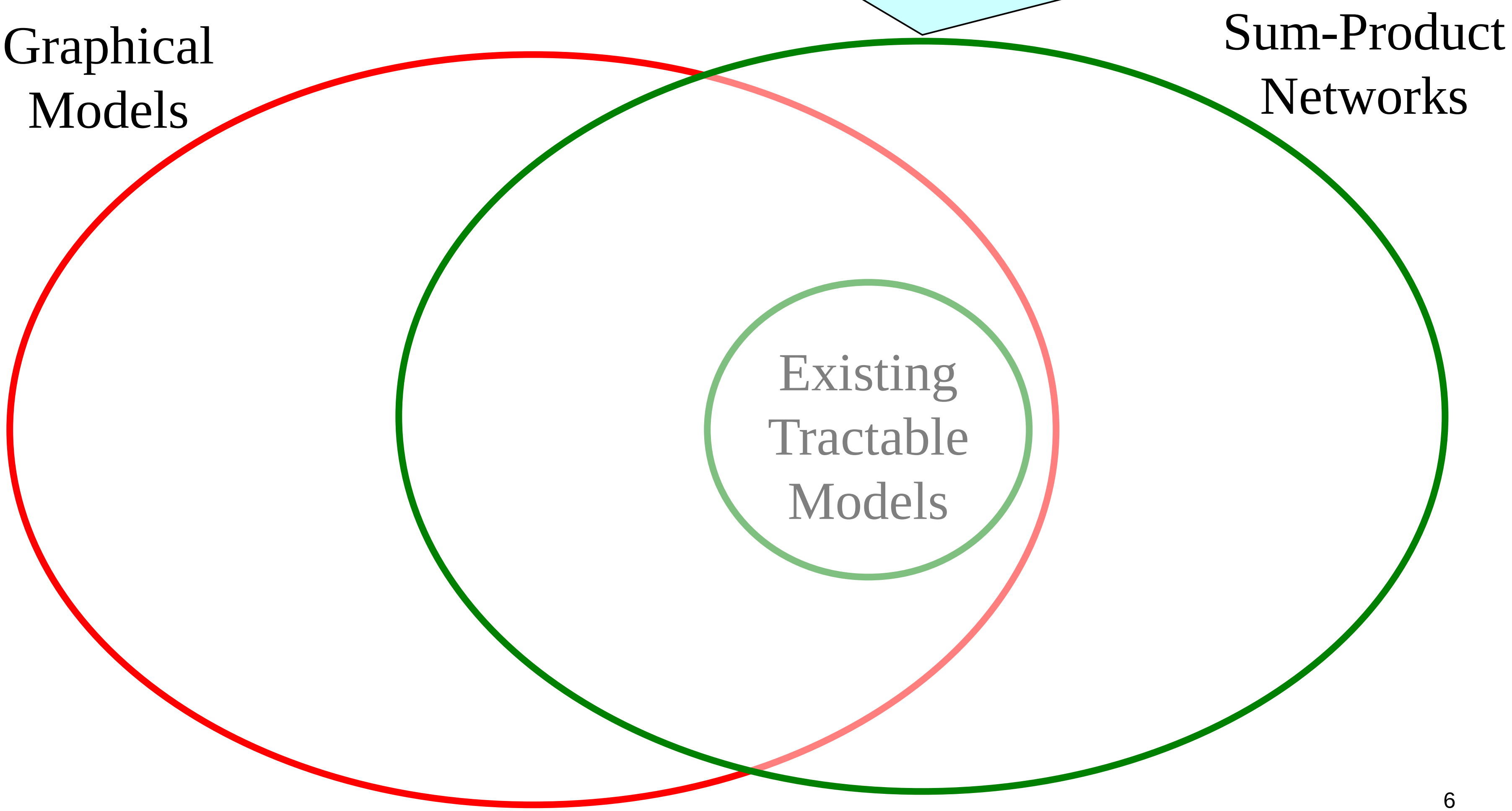
**Inference:** Still approximate

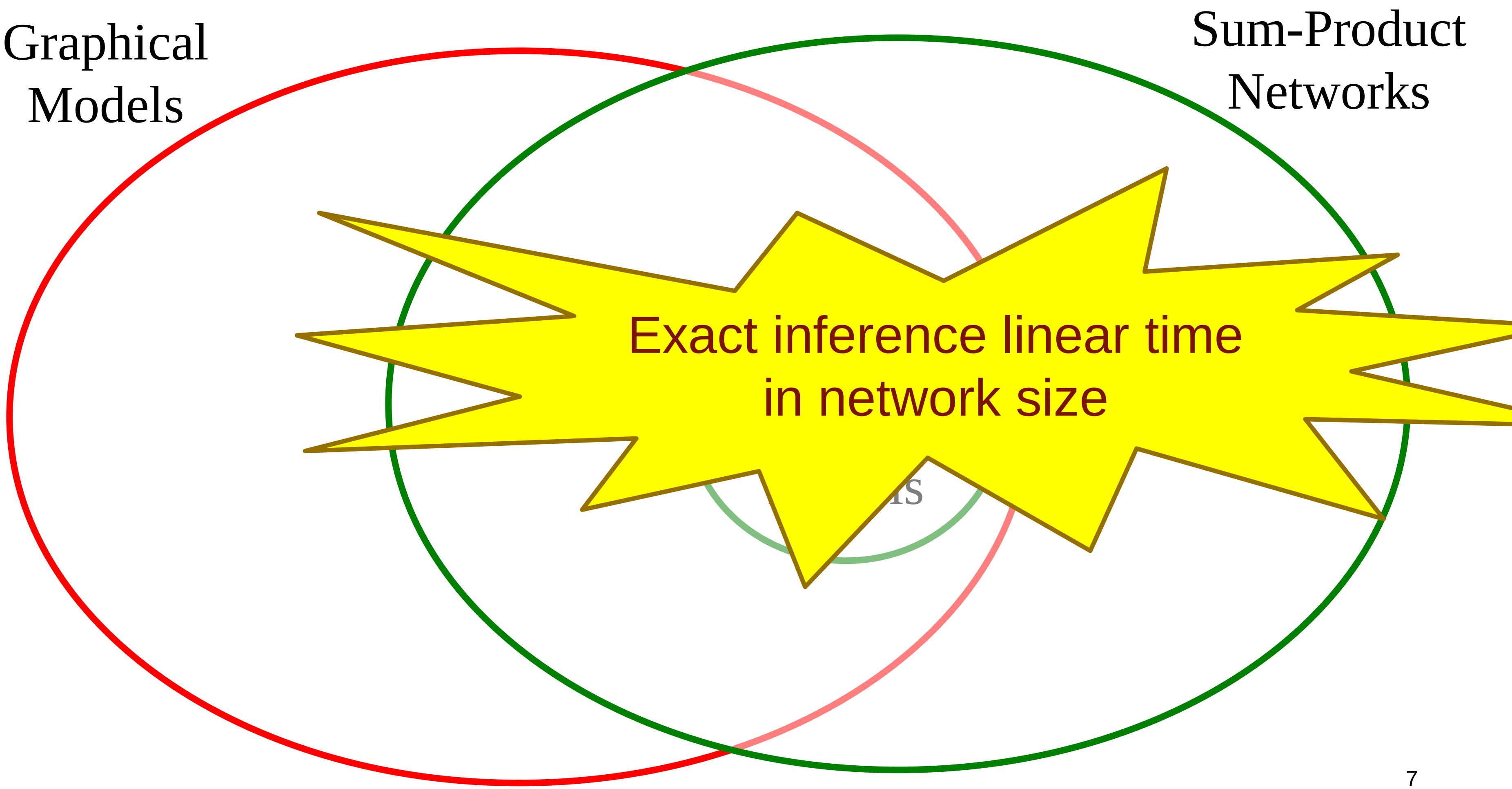
Graphical
Models

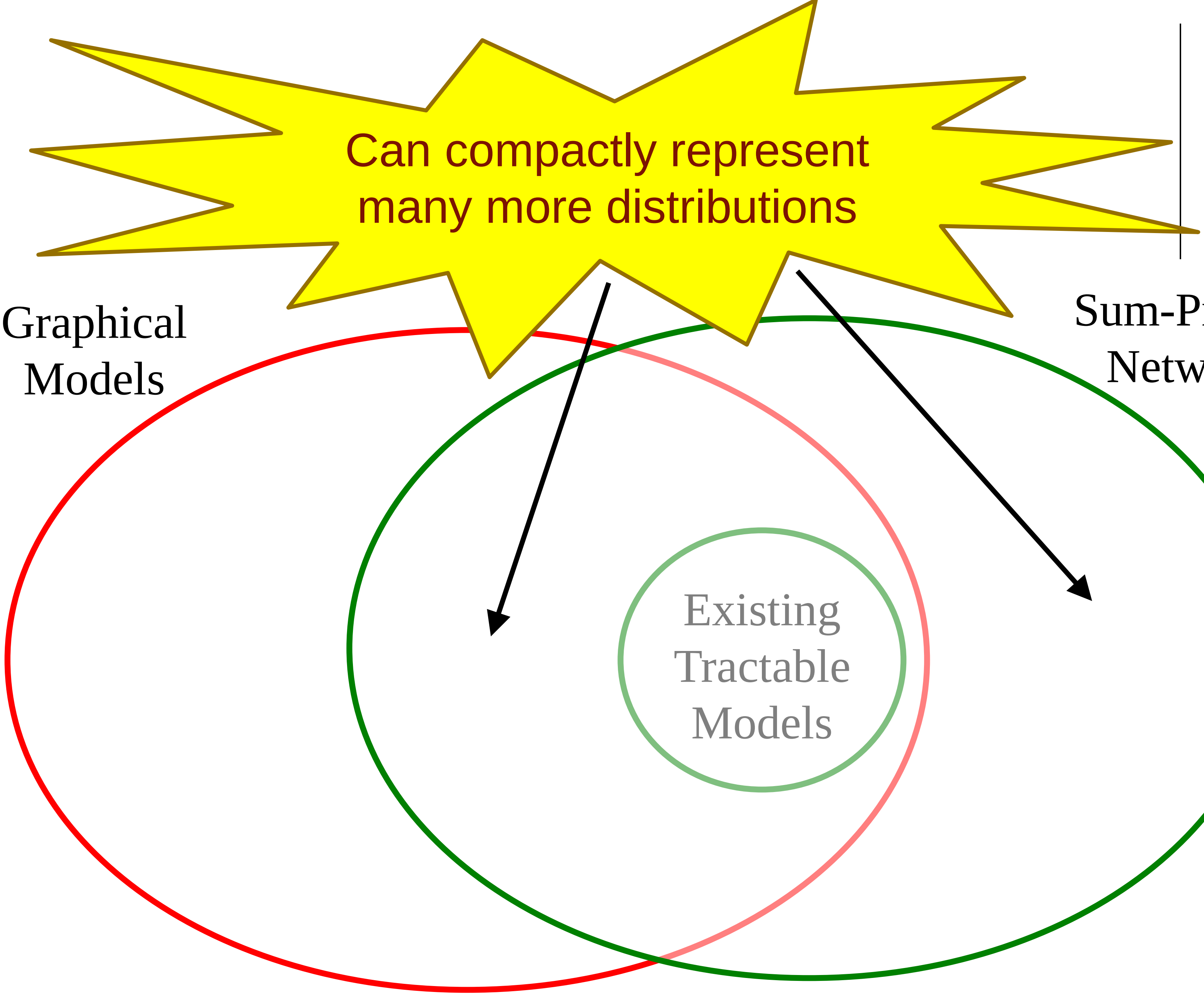

E.g., hierarchical mixture model, thin junction tree, etc.
**Problem:** **Too restricted**
Graphical Models
Existing Tractable Models

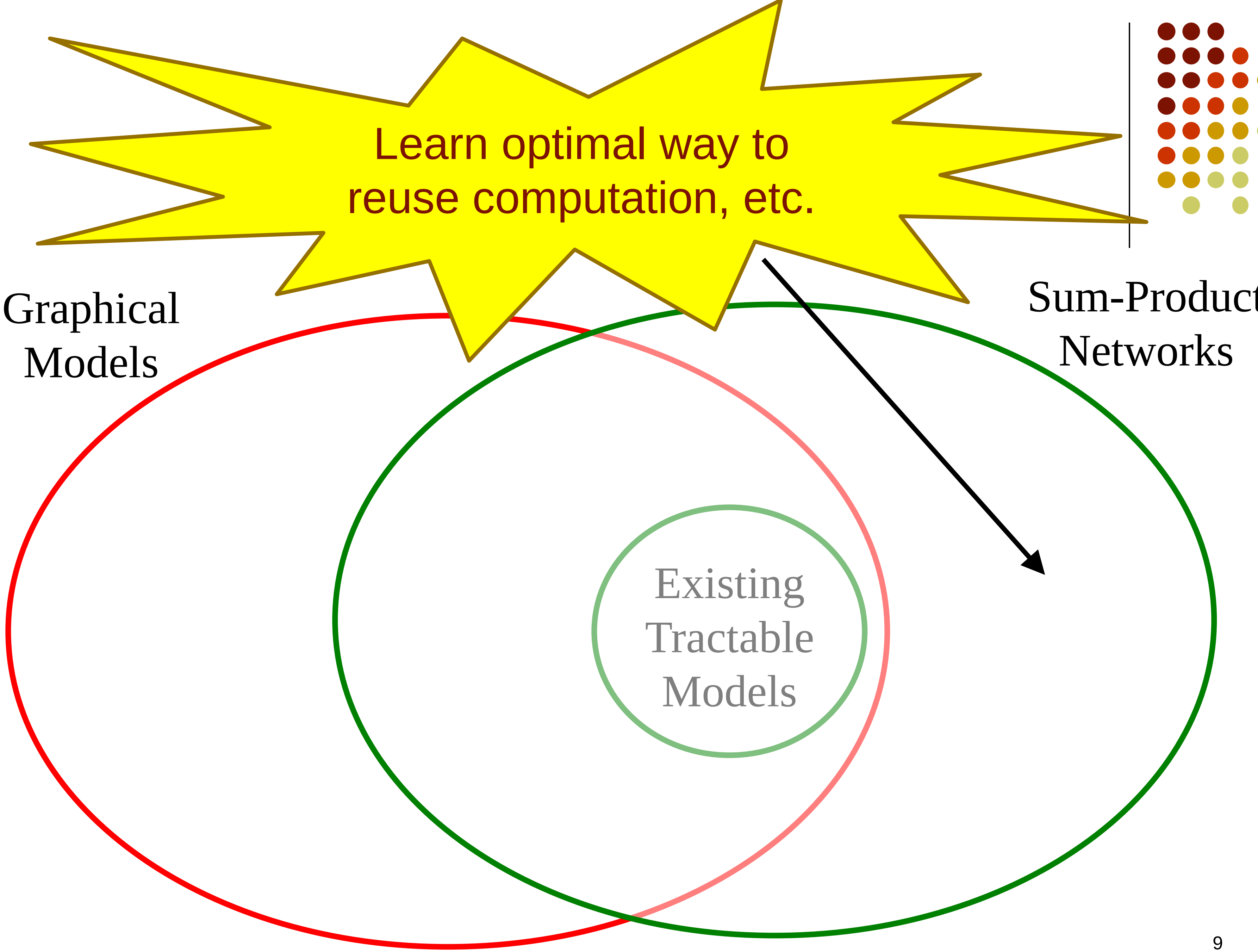
## This Talk: Sum-Product Networks

Compactly represent partition function using a deep network




Graphical Models
Sum-Product Networks
Exact inference linear time in network size


Can compactly represent
many more distributions
Graphical
Models
Sum-Product
Networks
Existing
Tractable
Models


Learn optimal way to
reuse computation, etc.
Graphical
Models
Sum-Product
Networks
Existing
Tractable
Models

# Outline

- **Sum-product networks (SPNs)**
- Learning SPN
- Experimental results
- Conclusion

# Why Is Inference Hard?

$$P(X_1, \cdots, X_N) = \frac{1}{Z} \prod_j \Phi_j(X_1, \cdots, X_N)$$

- Bottleneck: Summing out variables
- E.g.: Partition function

Sum of exponentially many products

$$Z = \sum_X \prod_j \Phi_j(X)$$

# Alternative Representation

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

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

**Network Polynomial** [Darwiche, 2003]

# Alternative Representation

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

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

**Network Polynomial** [Darwiche, 2003]

# 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 |

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

**Network Polynomial** [Darwiche, 2003]

# 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 |

$\boxed{e: X_1 = 1}$

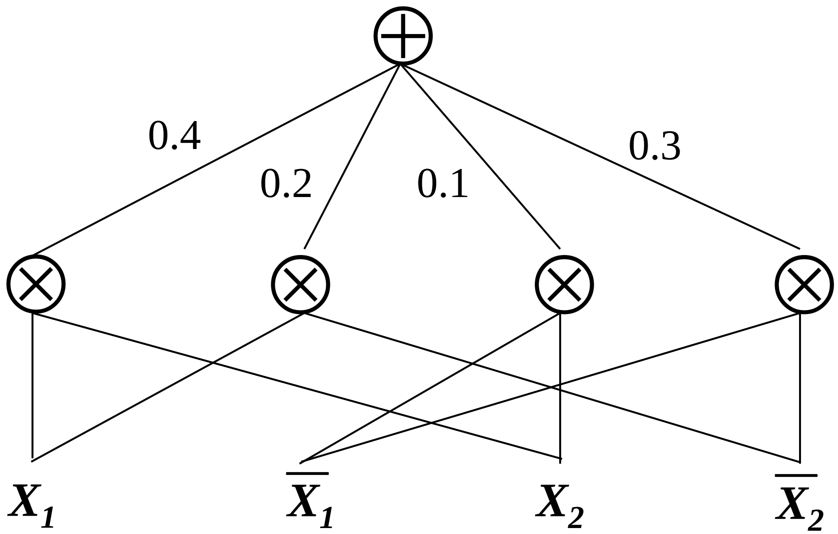
$$P(e) = \mathbf{0.4 \cdot X_1 \cdot X_2} \\ \mathbf{+ 0.2 \cdot X_1 \cdot \overline{X}_2} \\ + 0.1 \cdot \overline{X}_1 \cdot X_2 \\ + 0.3 \cdot \overline{X}_1 \cdot \overline{X}_2$$

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

**Easy: Set both indicators to 1**

# Graphical Representation

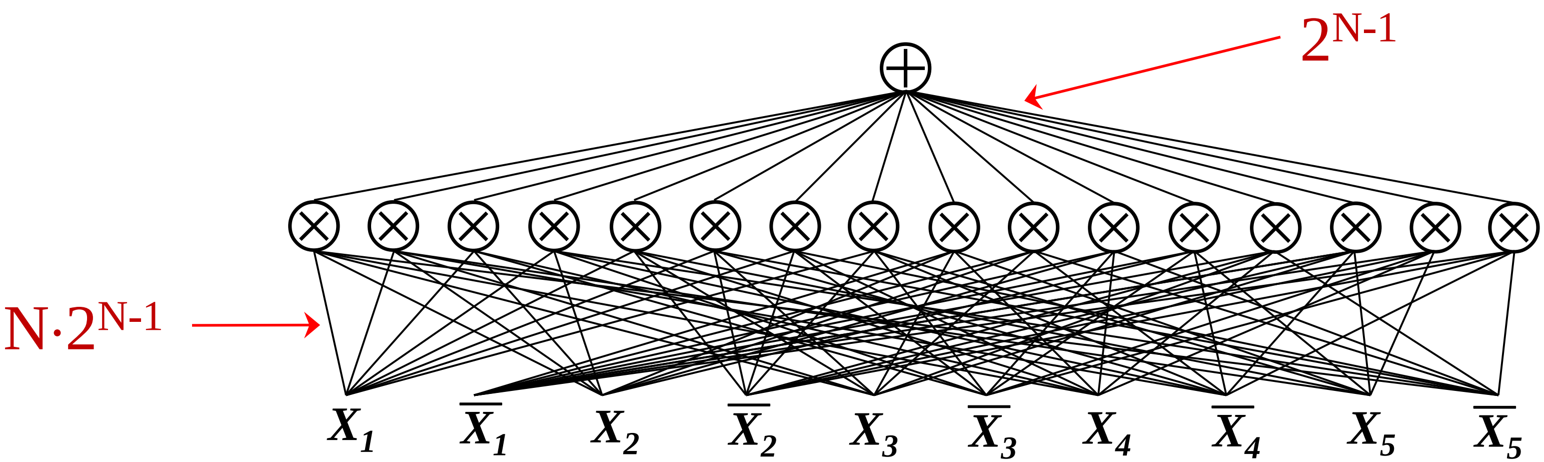
| $X_1$ | $X_2$ | $P(X)$ |
|---|---|---|
| 1 | 1 | 0.4 |
| 1 | 0 | 0.2 |
| 0 | 1 | 0.1 |
| 0 | 0 | 0.3 |



# But … Exponentially Large

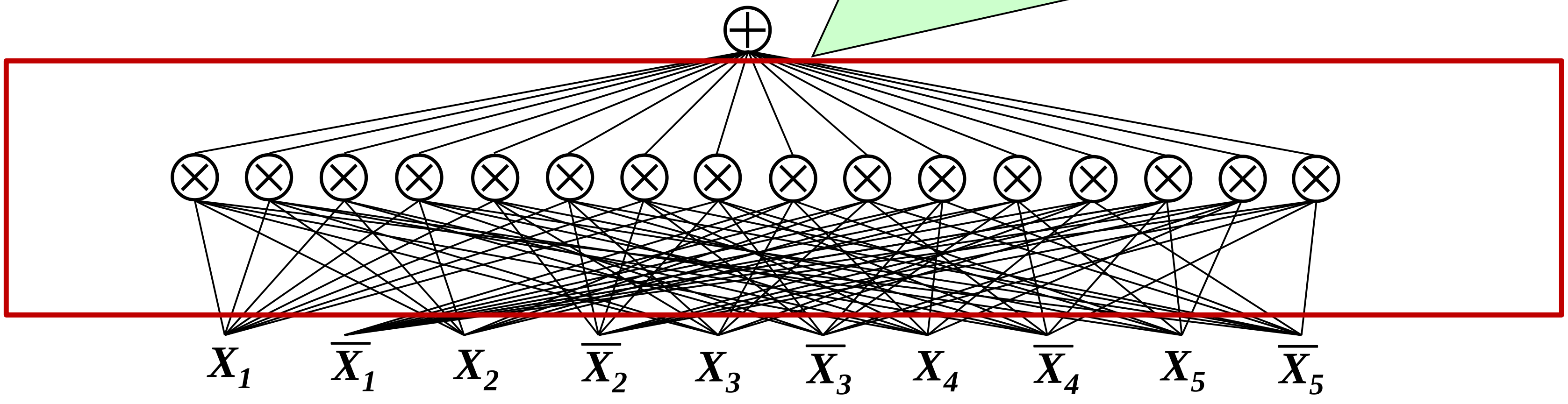
## Example: Parity

Uniform distribution over states with even number of 1's



# But … Exponentially Large

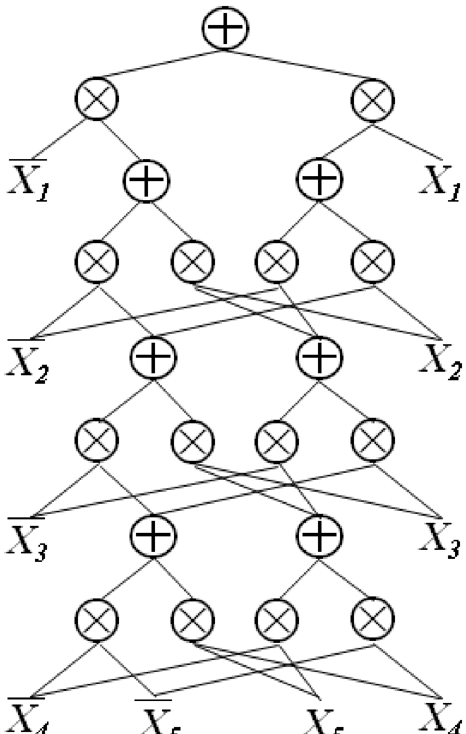
**Can we make this more compact?**



# Use a Deep Network

## Example: Parity

Uniform distribution over states with even number of 1's

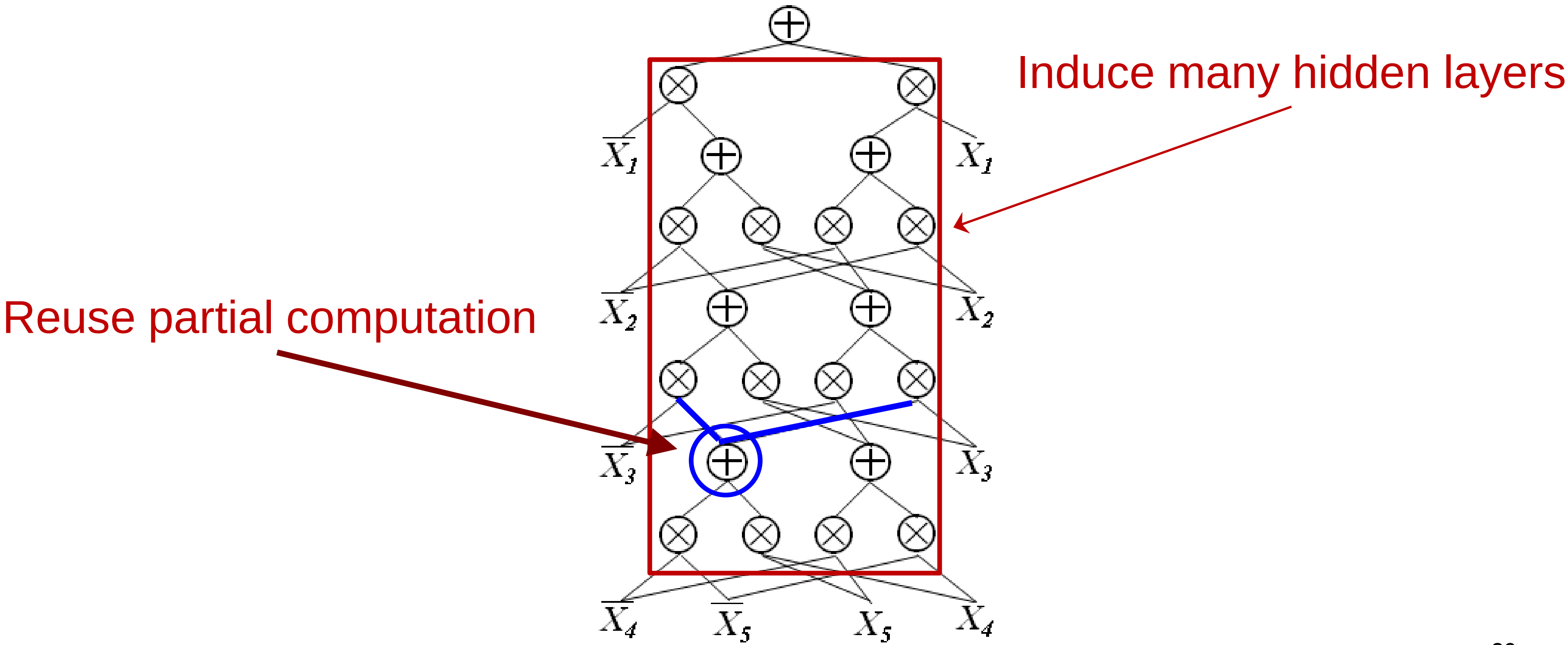



O(N)

# Use a Deep Network

## Example: Parity

Uniform distribution over states of even number of 1's

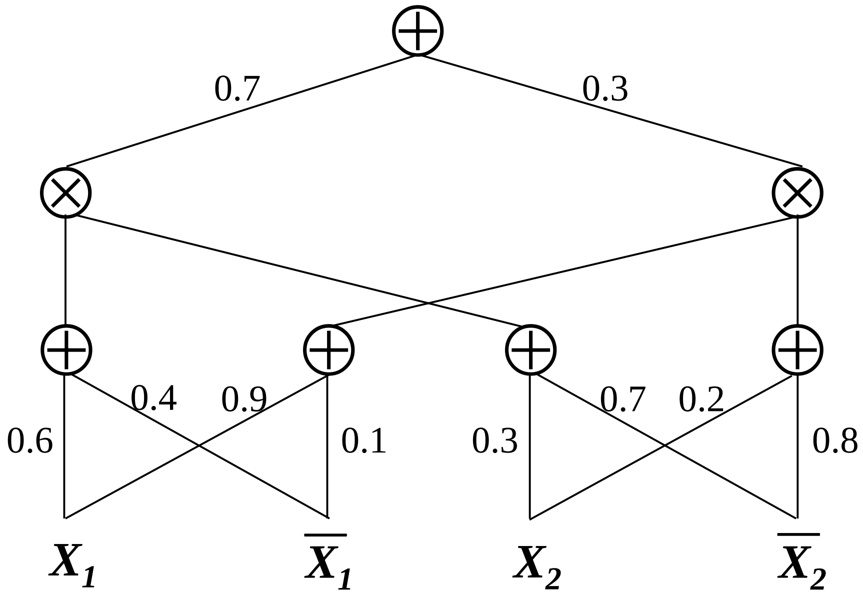


# Arithmetic Circuits (ACs)

- Data structure for efficient inference
  - Darwiche [2003]
  - Compilation target of Bayesian networks
- **Key idea:** Use ACs instead to define a new class of deep probabilistic models
- Develop new deep learning algorithms for this class of models

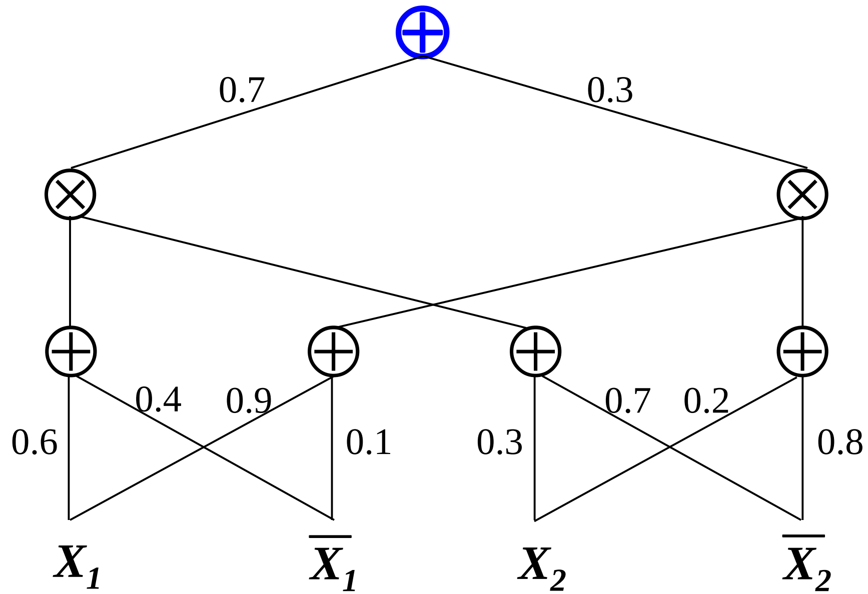
# Sum-Product Networks (SPNs)

- Rooted DAG
- Nodes: Sum, product, input indicator
- Weights on edges from sum to children



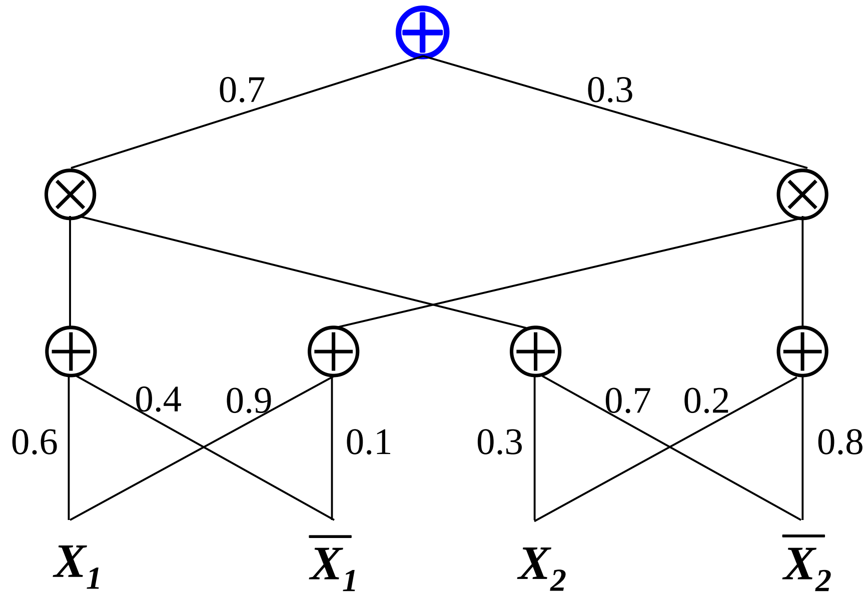
# Distribution Defined by SPN

$$P(X) \propto S(X)$$



# Can We Sum Out Variables?

$$P(e) \propto \Sigma_{X \sim e} S(X) \overset{?}{=} S(e)$$




$e$: $X_1 = 1$ 1 0 $\boxed{1 \quad 1}$

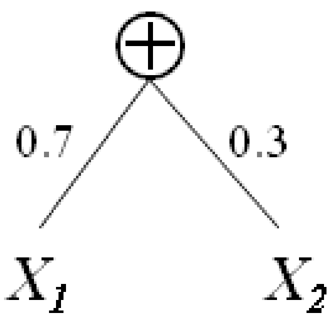
# Valid SPN

- *SPN is valid if* $S(e) = \Sigma_{X \sim e} S(X)$ *for all* $e$
- Valid $\rightarrow$ Can compute marginals efficiently
- Partition function $Z$ can be computed by setting all indicators to 1

# Valid SPN: General Conditions

**Theorem:** *SPN is valid if it is complete & consistent*

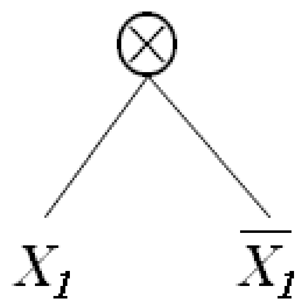
**Complete:** Under sum, children cover the same set of variables




Incomplete

$$S(e) \leq \Sigma_{X\sim e}\, S(X)$$

**Consistent:** Under product, no variable in one child and negation in another

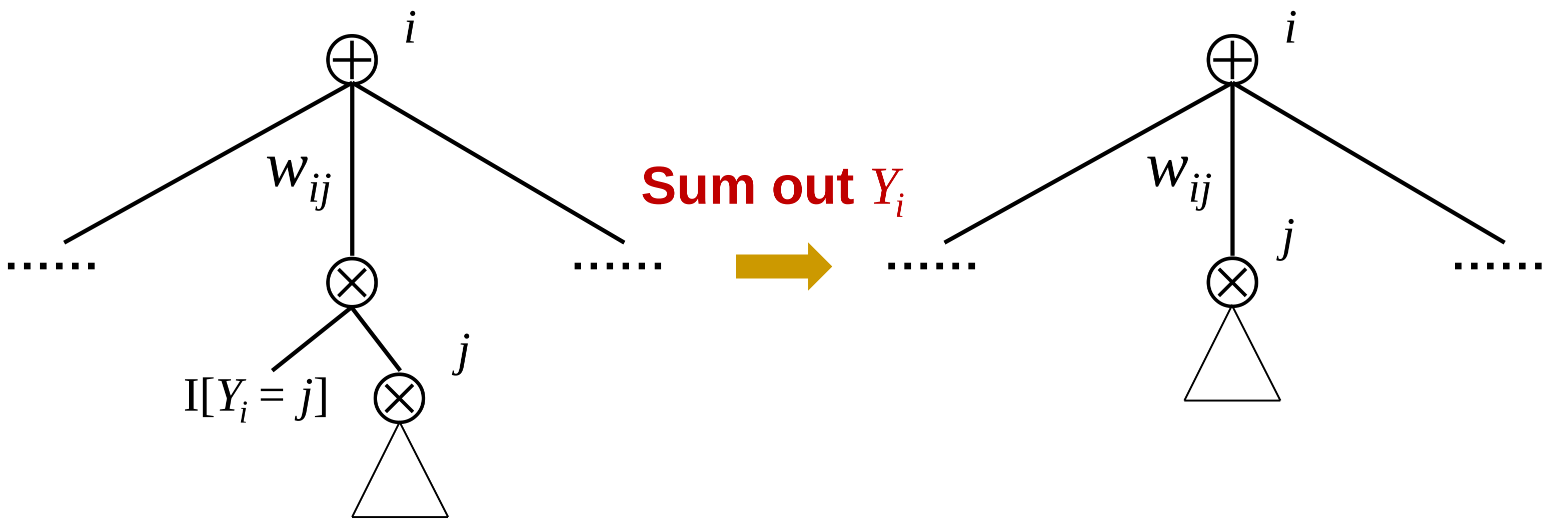



Inconsistent

$$S(e) \geq \Sigma_{X\sim e}\, S(X)$$

# Semantics of Sums and Products

- Product ~ Feature → Form feature hierarchy
- Sum ~ Mixture (with hidden var. summed out)

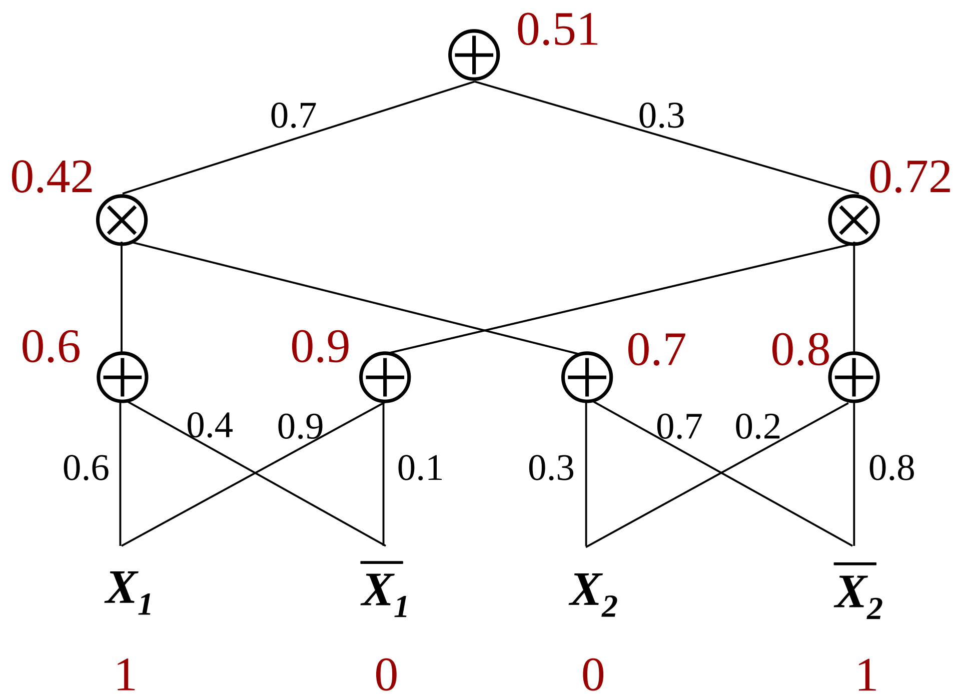


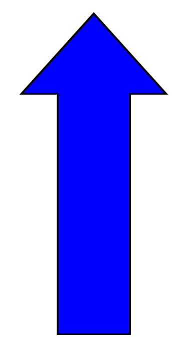
# Inference

Probability: $P(X) = S(X) / Z$

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

| | |
|---|---|
| $X_1$ | 1 |
| $\bar{X}_1$ | 0 |
| $X_2$ | 0 |
| $\bar{X}_2$ | 1 |



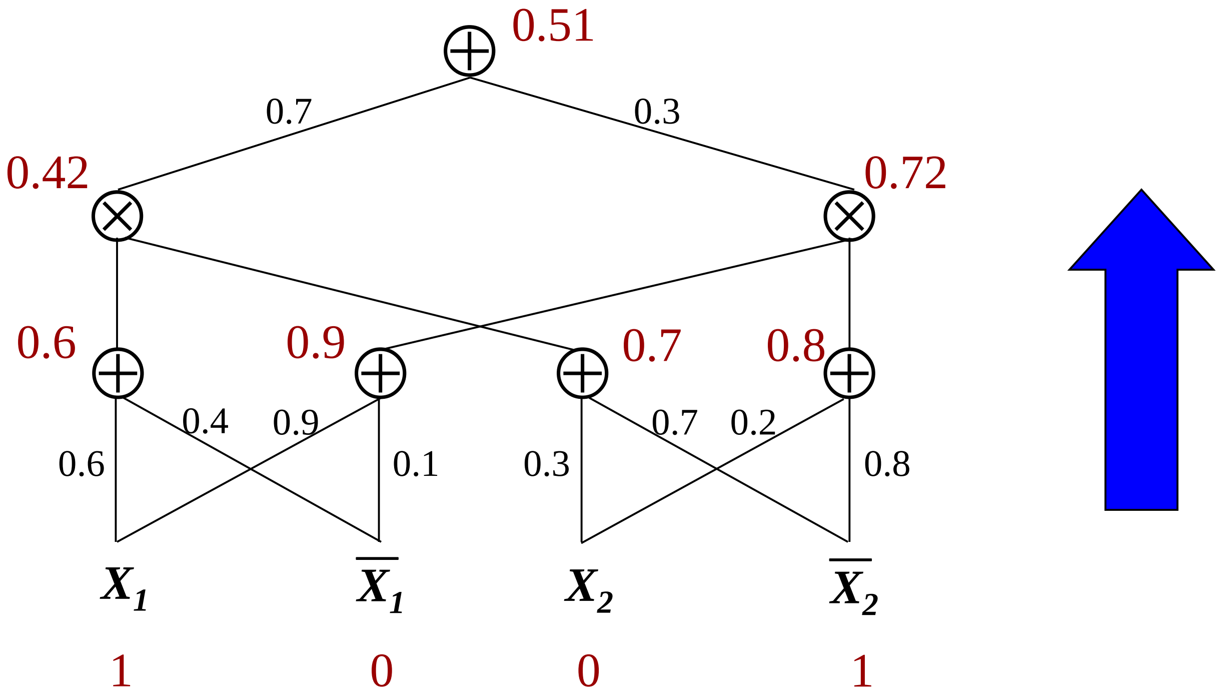



# Inference

If weights sum to 1 at each sum node
Then $Z = 1$, $P(X) = S(X)$

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

| | |
|---|---|
| $X_1$ | 1 |
| $\bar{X}_1$ | 0 |
| $X_2$ | 0 |
| $\bar{X}_2$ | 1 |

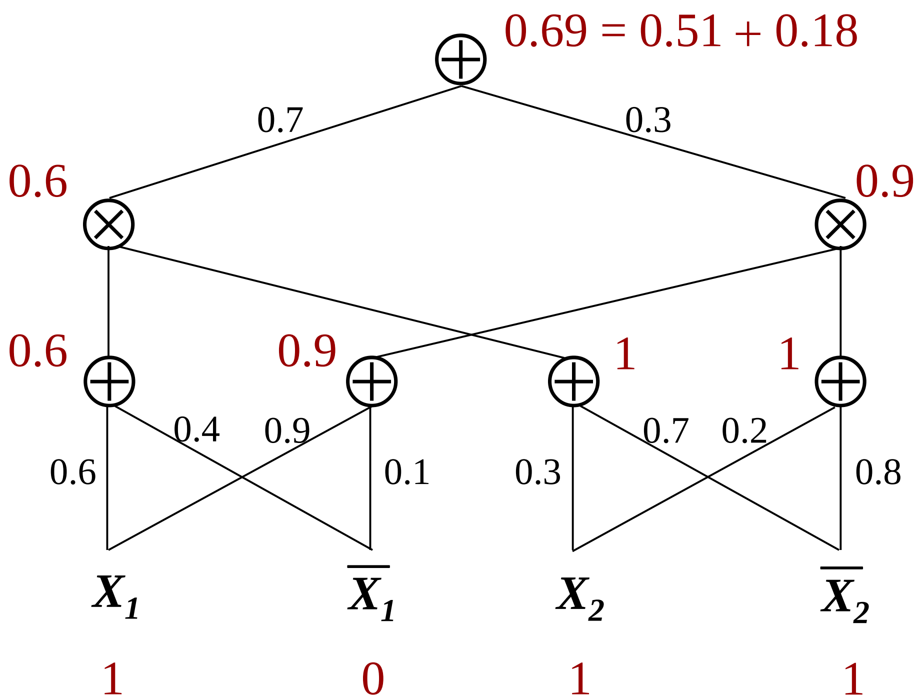


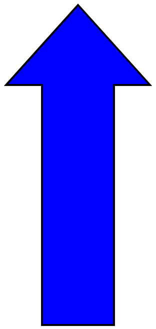
# Inference

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

$e$: $X_1 = 1$

| | |
|---|---|
| $X_1$ | 1 |
| $\overline{X}_1$ | 0 |
| $X_2$ | 1 |
| $\overline{X}_2$ | 1 |



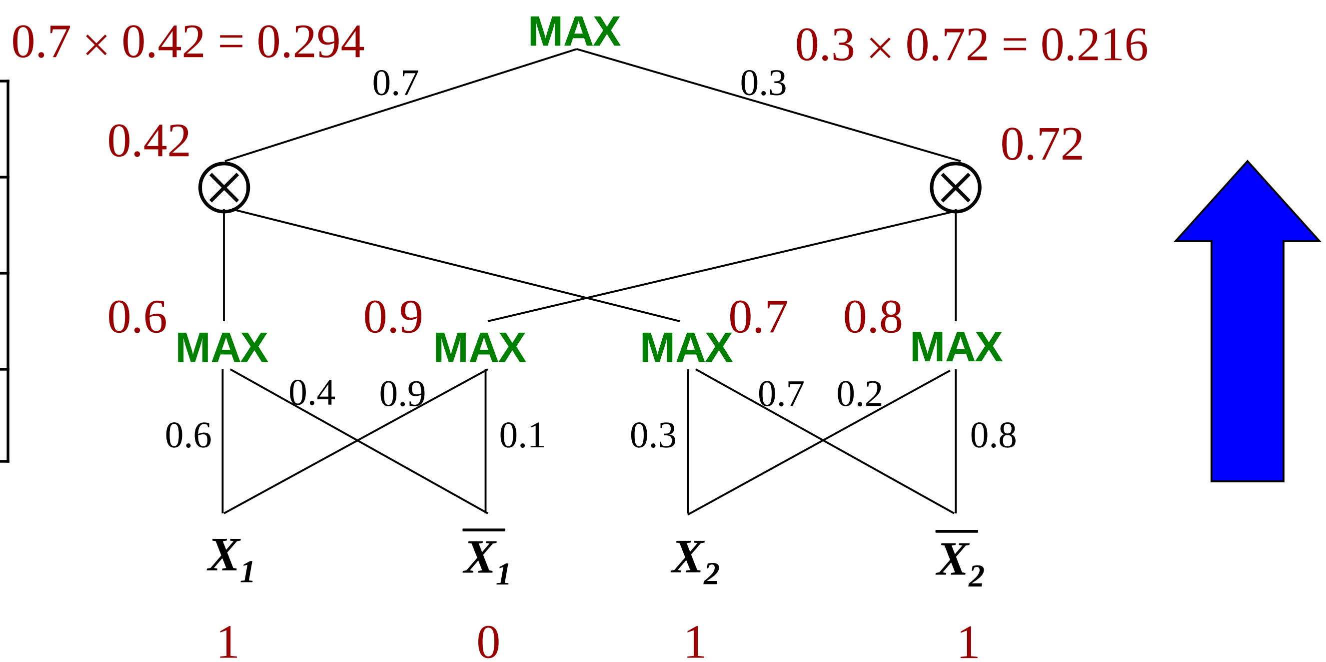



# Inference

MPE: Replace sums with maxes

$e$: $X_1 = 1$

| | |
|---|---|
| $X_1$ | 1 |
| $\overline{X}_1$ | 0 |
| $X_2$ | 1 |
| $\overline{X}_2$ | 1 |



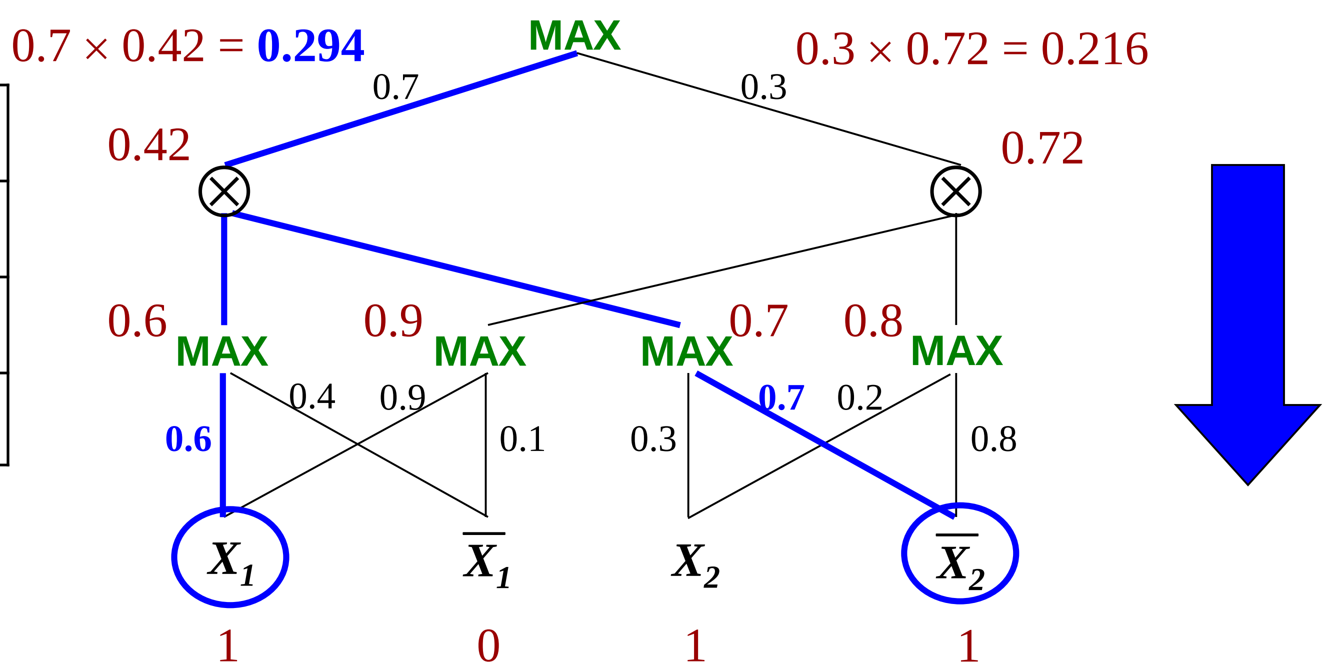

Darwiche [2003]

# Inference

MAX: Pick child with highest value

$e$: $X_1 = 1$

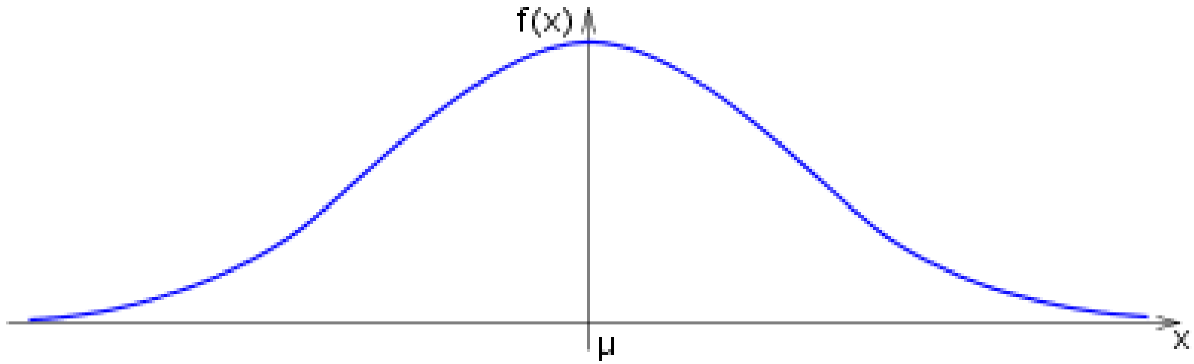
| | |
|---|---|
| $X_1$ | 1 |
| $\overline{X}_1$ | 0 |
| $X_2$ | 1 |
| $\overline{X}_2$ | 1 |




Darwiche [2003]

# Handling Continuous Variables

- Sum $\rightarrow$ Integral over input




- Simplest case: Indicator $\rightarrow$ Gaussian

  SPN compactly defines a very large mixture of Gaussians

# SPNs Everywhere

- Graphical models
  - Existing tractable mdls. & inference mthds.
  - Determinism, context-specific indep., etc.
  - Can potentially learn the optimal way

# SPNs Everywhere

- Graphical models
- Methods for efficient inference

E.g., arithmetic circuits,
AND/OR graphs, case-factor diagrams

**SPNs are a class of probabilistic models**

**SPNs have validity conditions**

**SPNs can be learned from data**

# SPNs Everywhere

- Graphical models
- Models for efficient inference
- General, probabilistic convolutional network

**Sum: Average-pooling**

**Max: Max-pooling**

# SPNs Everywhere

- Graphical models
- Models for efficient inference
- General, probabilistic convolutional network
- Grammars in vision and language

E.g., object detection grammar,
probabilistic context-free grammar

**Sum: Non-terminal**
**Product: Production rule**

# Outline

- Sum-product networks (SPNs)
- **Learning SPN**
- Experimental results
- Conclusion

# General Approach

- Start with a dense SPN
- Find the structure by learning weights

  Zero weights signify absence of connections
- Can learn with gradient descent or EM

# The Challenge

- **Gradient diffusion:** Gradient quickly dilutes
- Similar problem with EM
- Hard EM overcomes this problem

# Our Learning Algorithm

- Online learning + Hard EM
- Sum node maintains counts for each child
- For each example
  - Find MPE instantiation with current weights
  - Increment count for each chosen child
  - Renormalize to set new weights
- Repeat until convergence

# Outline

- Sum-product networks (SPNs)
- Learning SPN
- **Experimental results**
- Conclusion

# Task: Image Completion

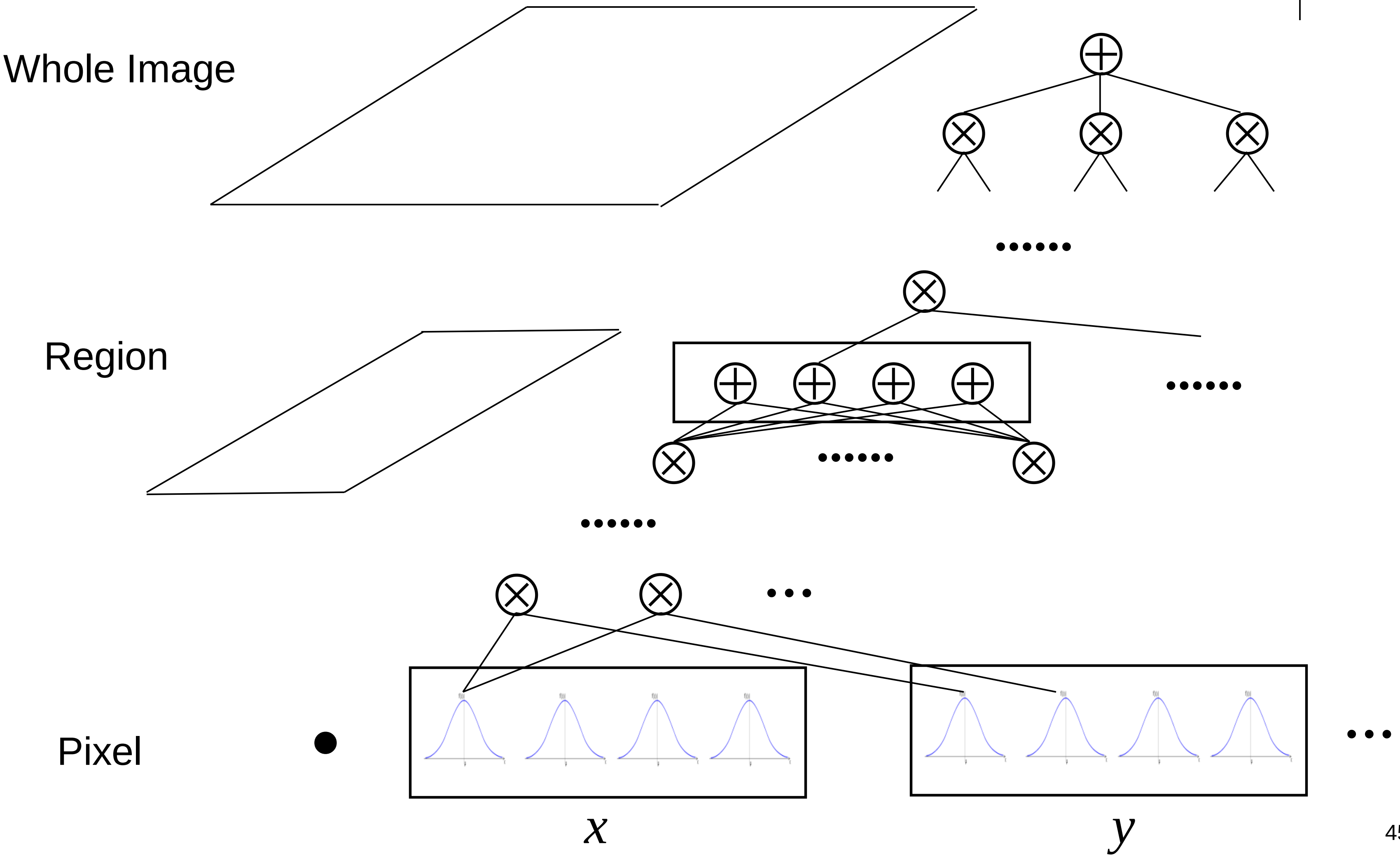
- Methodology:
  - Learn a model from training images
  - Complete unseen test images
  - Measure mean square errors
- Very challenging
- Good for evaluating deep models

# Datasets

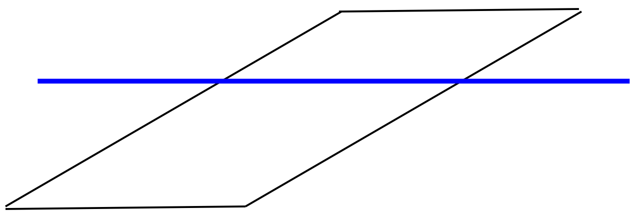
- **Main evaluation: Caltech-101** [Fei-Fei et al., 2004]
  - 101 categories, e.g., faces, cars, elephants
  - Each category: 30 – 800 images
- Also, Olivetti [Samaria & Harter, 1994] (400 faces)
- Each category: Last third for test

  Test images: Unseen objects

# SPN Architecture

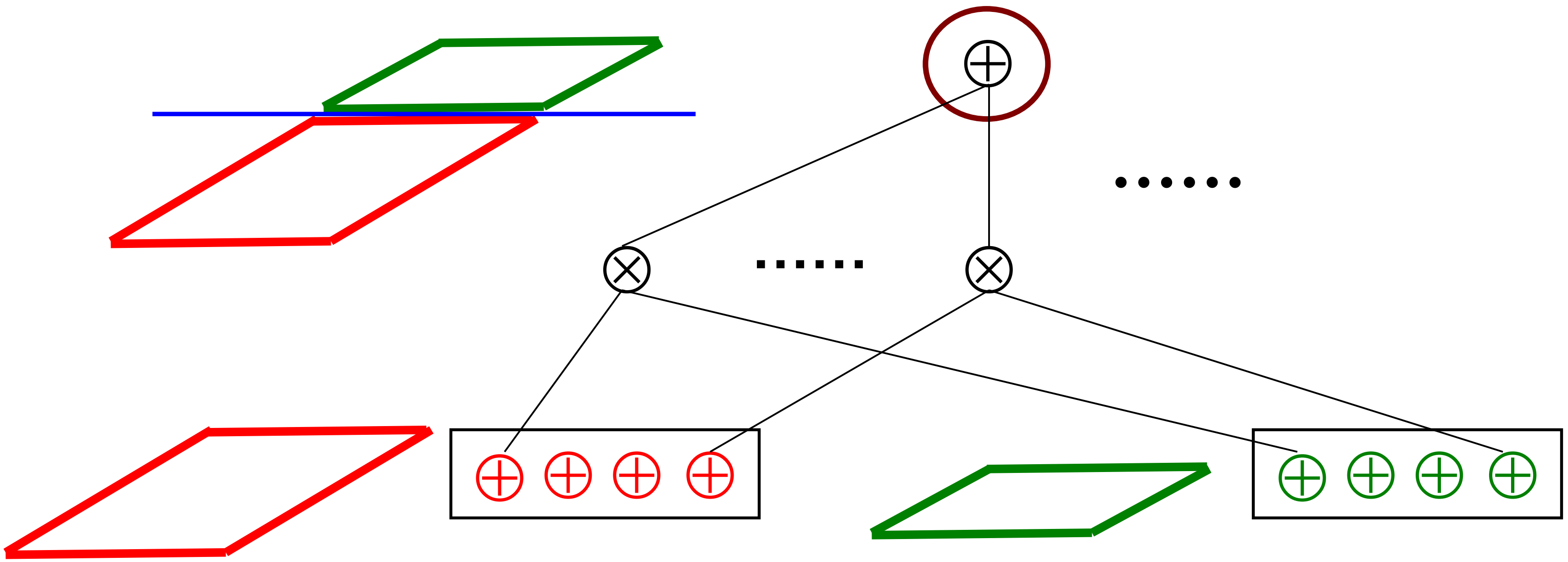


# Decomposition



$\oplus$

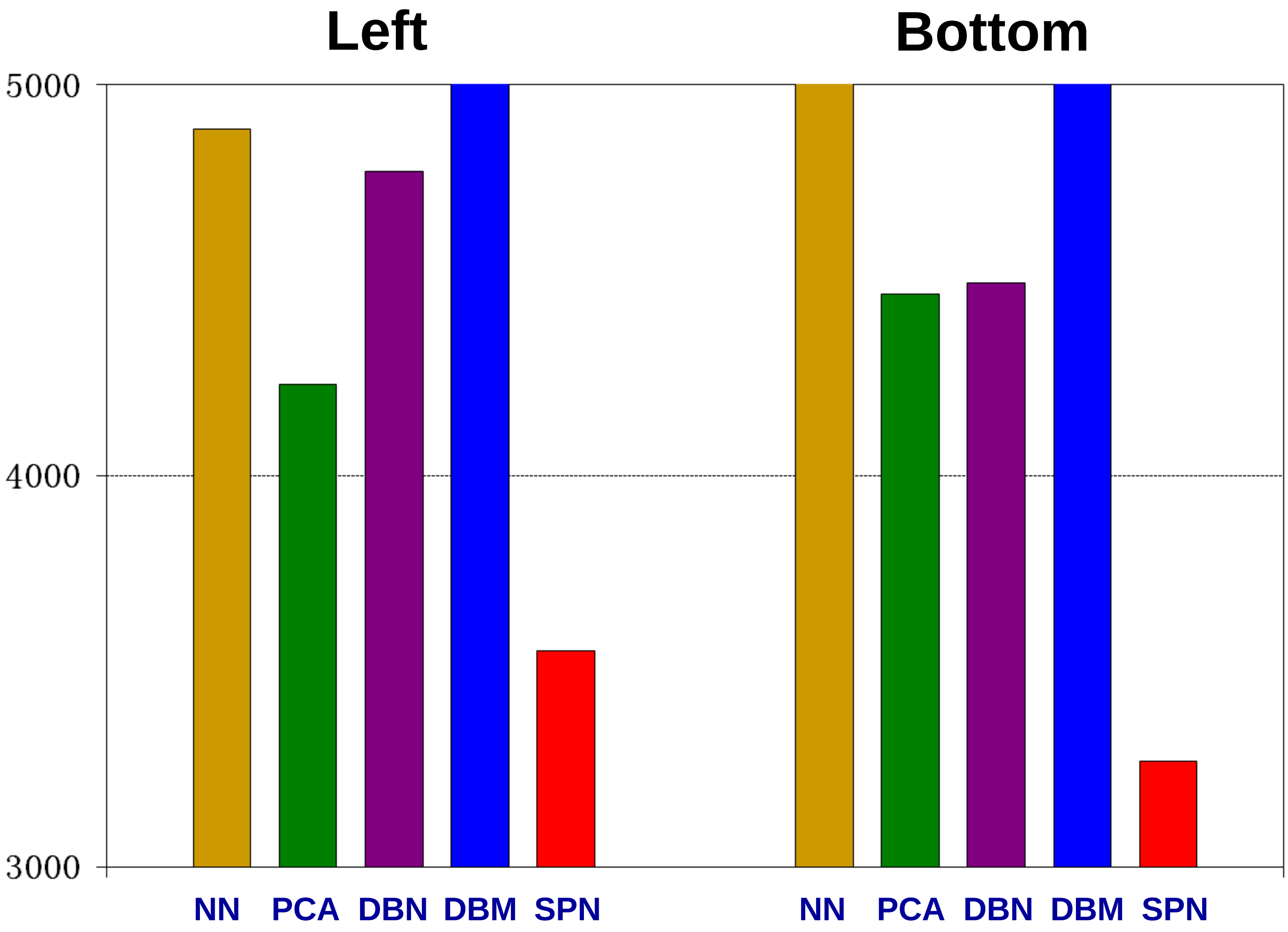
# Decomposition



# Systems

- SPN
- DBM [Salakhutdinov & Hinton, 2010]
- DBN [Hinton & Salakhutdinov, 2006]
- PCA [Turk & Pentland, 1991]
- Nearest neighbor [Hays & Efros, 2007]

# Caltech: Mean-Square Errors



# 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

Original

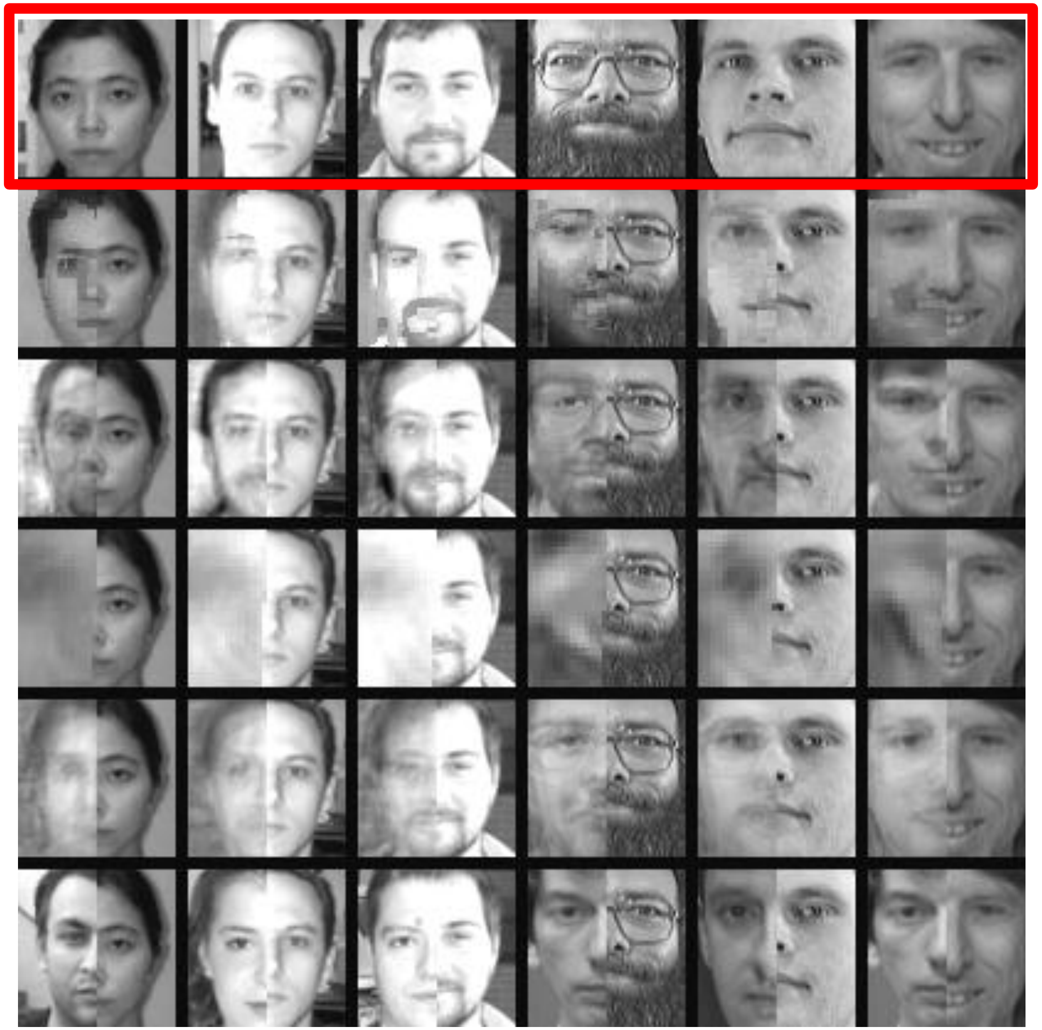
SPN

DBM

DBN

PCA

Nearest Neighbor



# Example Completions

Original

SPN

DBM

DBN

PCA

Nearest Neighbor

# Example Completions

Original

SPN

DBM

DBN

PCA

Nearest Neighbor

# Example Completions

Original

SPN

DBM

DBN

PCA

Nearest Neighbor

# Example Completions

Original

SPN

DBM

DBN

PCA

Nearest Neighbor

# Example Completions

# Open Questions

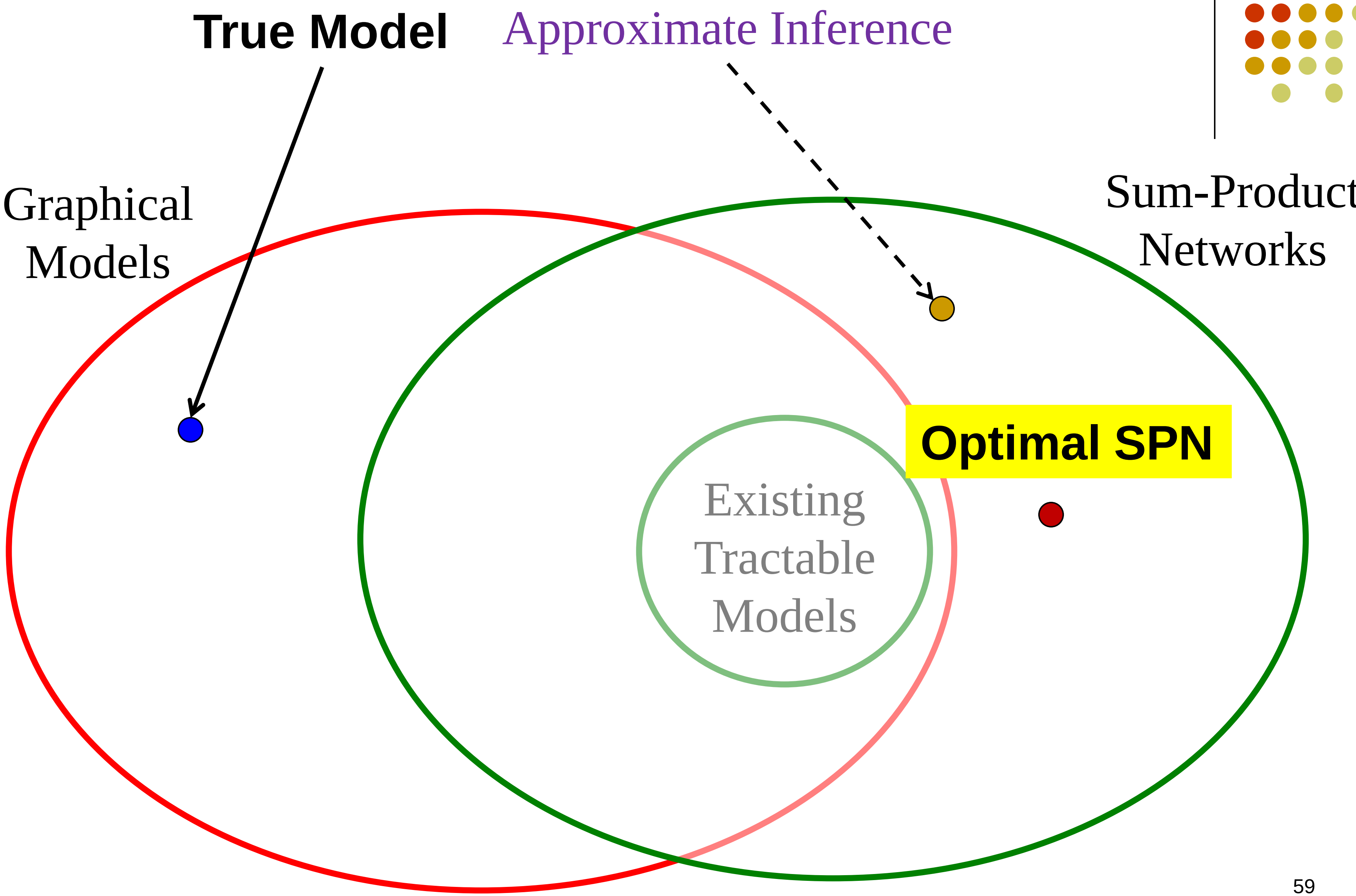
- Other learning algorithms
- Discriminative learning
- Architecture
- Continuous SPNs
- Sequential domains
- Other applications

# End-to-End Comparison

Data $\xrightarrow{\text{Approximate}}$ **General Graphical Models** $\xrightarrow{\text{Approximate}}$ Performance

**Given same computation budget, which approach has better performance?**

Data $\xrightarrow{\text{Approximate}}$ **Sum-Product Networks** $\xrightarrow{\text{Exact}}$ Performance


True Model
Approximate Inference
Graphical Models
Sum-Product Networks
Optimal SPN
Existing Tractable Models

# Conclusion

- Sum-product networks (SPNs)
  - DAG of sums and products
  - Compactly represent partition function
  - Learn many layers of hidden variables
- Exact inference: Linear time in network size
- Deep learning: Online hard EM
- Substantially outperform state of the art on image completion