Sum-Product Networks

STAT946 Deep Learning

Guest Lecture by Pascal Poupart
University of Waterloo

October 15, 2015

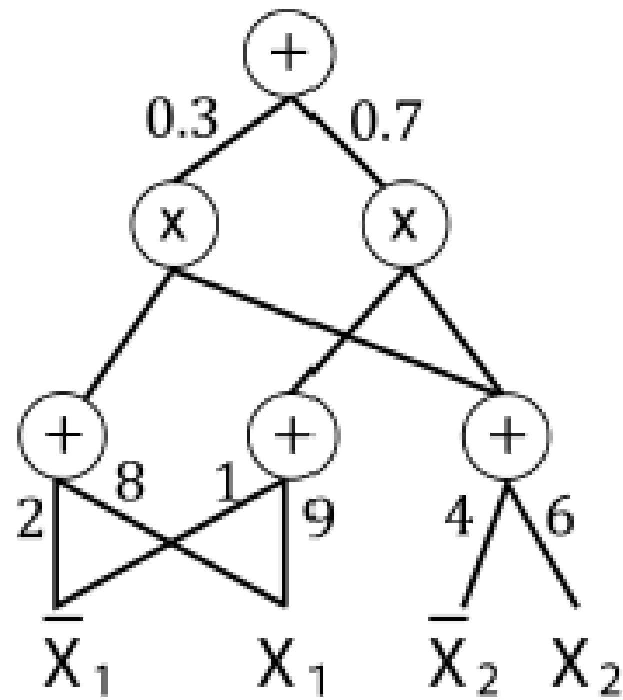
Outline

- Introduction
  - What is a Sum-Product Network?
  - Inference
  - Applications

- In more depth
  - Relationship to Bayesian networks
  - Parameter estimation
  - Online and distributed estimation
  - Dynamic SPNs for sequence data

What is a Sum-Product Network?

- Poon and Domingos, UAI 2011
- Acyclic directed graph of sums and products
- Leaves can be indicator variables or univariate distributions



Two Views

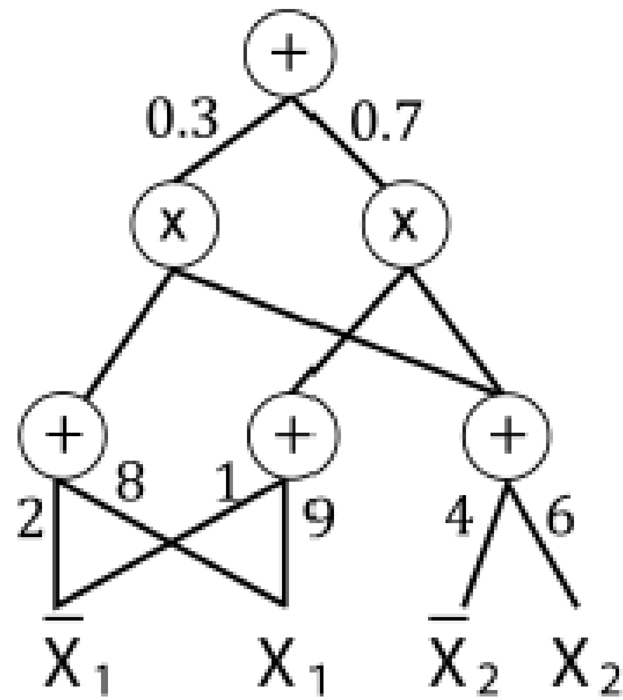
Deep architecture with clear semantics

Tractable probabilistic graphical model

Deep Architecture

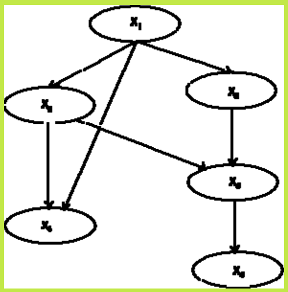
- Specific type of deep neural network
  - Activation function: product

- Advantage:
  - Clear semantics and well understood theory



Probabilistic Graphical Models

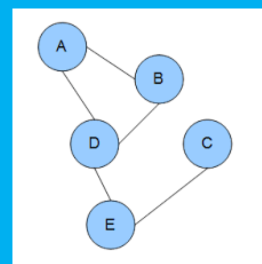
Bayesian Network



Graphical view of direct dependencies

Inference
#P: intractable

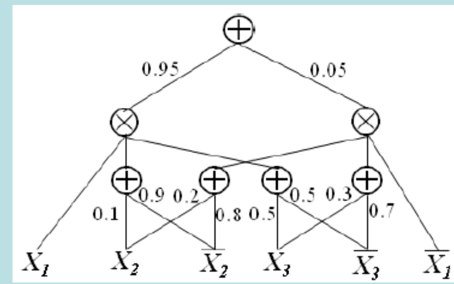
Markov Network



Graphical view of correlations

Inference
#P: intractable

Sum-Product Network

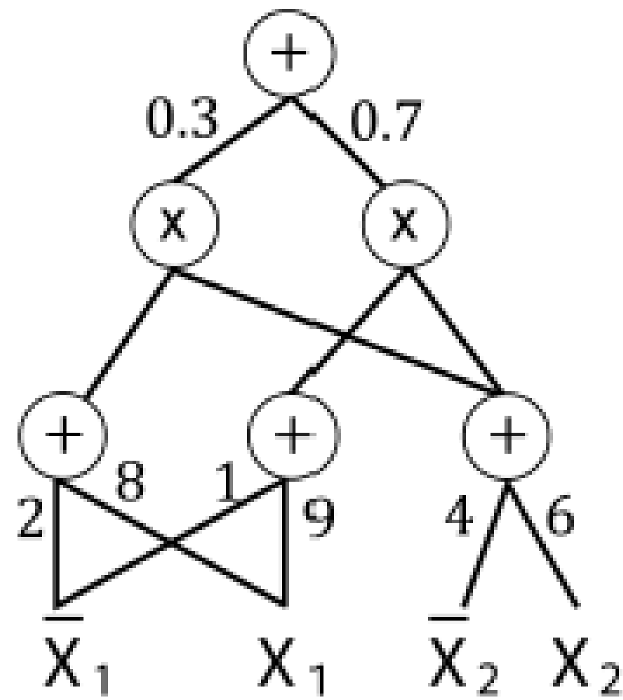


Graphical view of computation

Inference
P: tractable

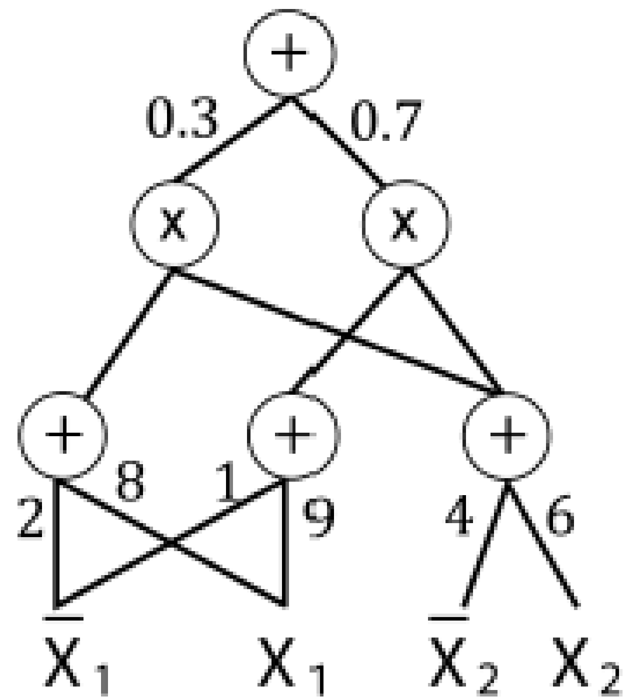
Probabilistic Inference

- SPN represents a joint distribution over a set of random variables
- Example:
  $\Pr(X_1 = true, X_2 = false)$



Marginal Inference

- Example:
  $\Pr(X_2 = false)$



Conditional Inference

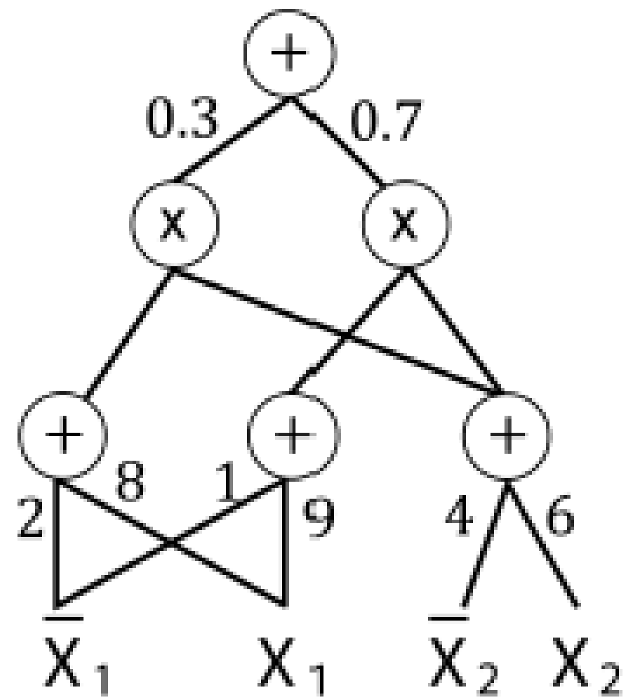
- Example:

$$\Pr(X_1 = true | X_2 = false)$$
$$= \frac{\Pr(X_1 = true, X_2 = false)}{\Pr(X_2 = false)}$$
$$=$$

- Hence any inference query can be answered in two bottom-up passes of the network
  - **Linear complexity!**

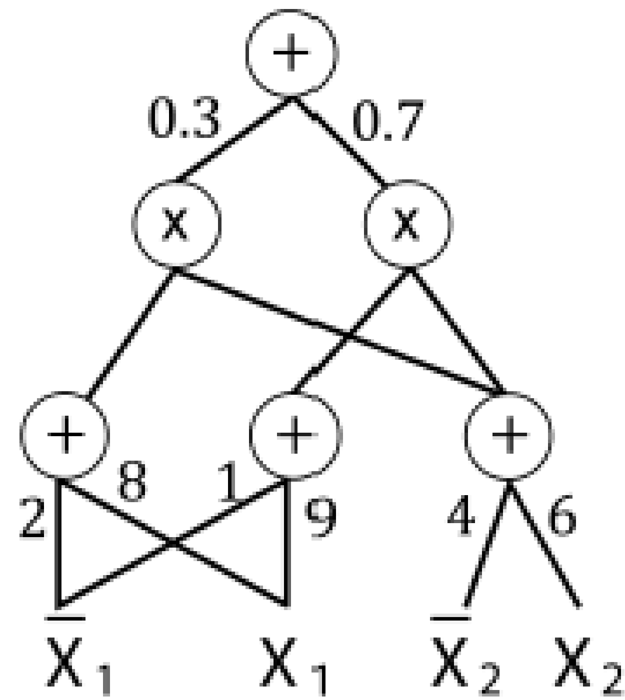
Semantics

- A **valid** SPN encodes a hierarchical mixture distribution
  - Sum nodes: hidden variables (mixture)
  - Product nodes: factorization (independence)



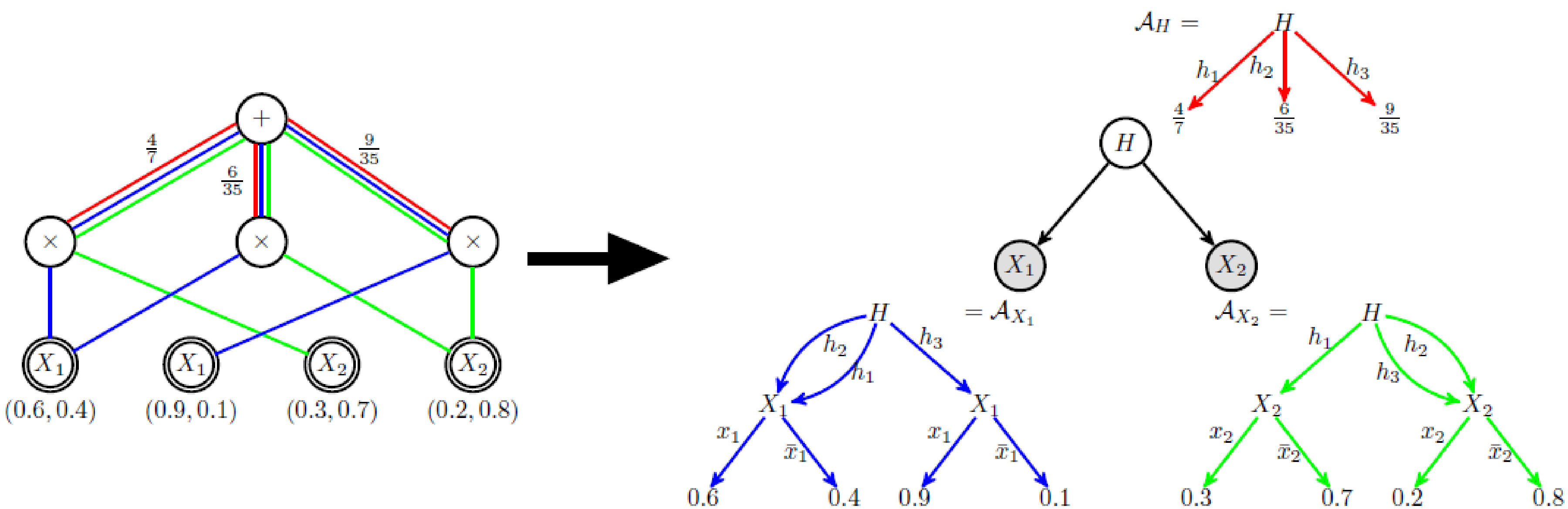
Definitions

- The scope of a node is the set of variables that appear in the sub-SPN rooted at the node
- An SPN is decomposable when each product node has children with disjoint scopes
- An SPN is complete when each sum node has children with identical scopes
- A decomposable and complete SPN is a valid SPN

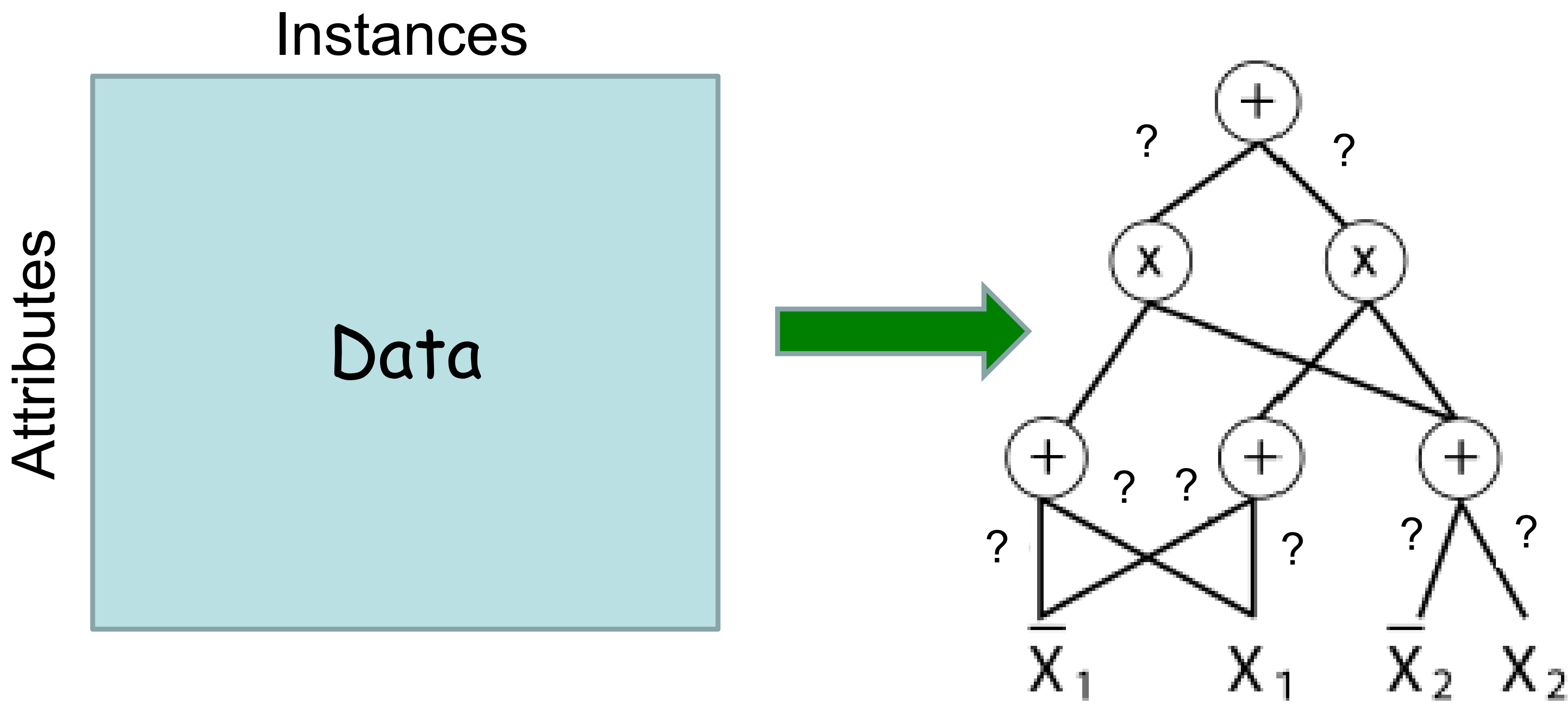


Relationship with Bayes Nets

- Any SPN can be converted into a bipartite Bayesian network (Zhao, Melibari, Poupart, ICML 2015)



Parameter Estimation



- Parameter Learning: estimate the weights
  - Expectation-Maximization, Gradient descent

Structure Estimation

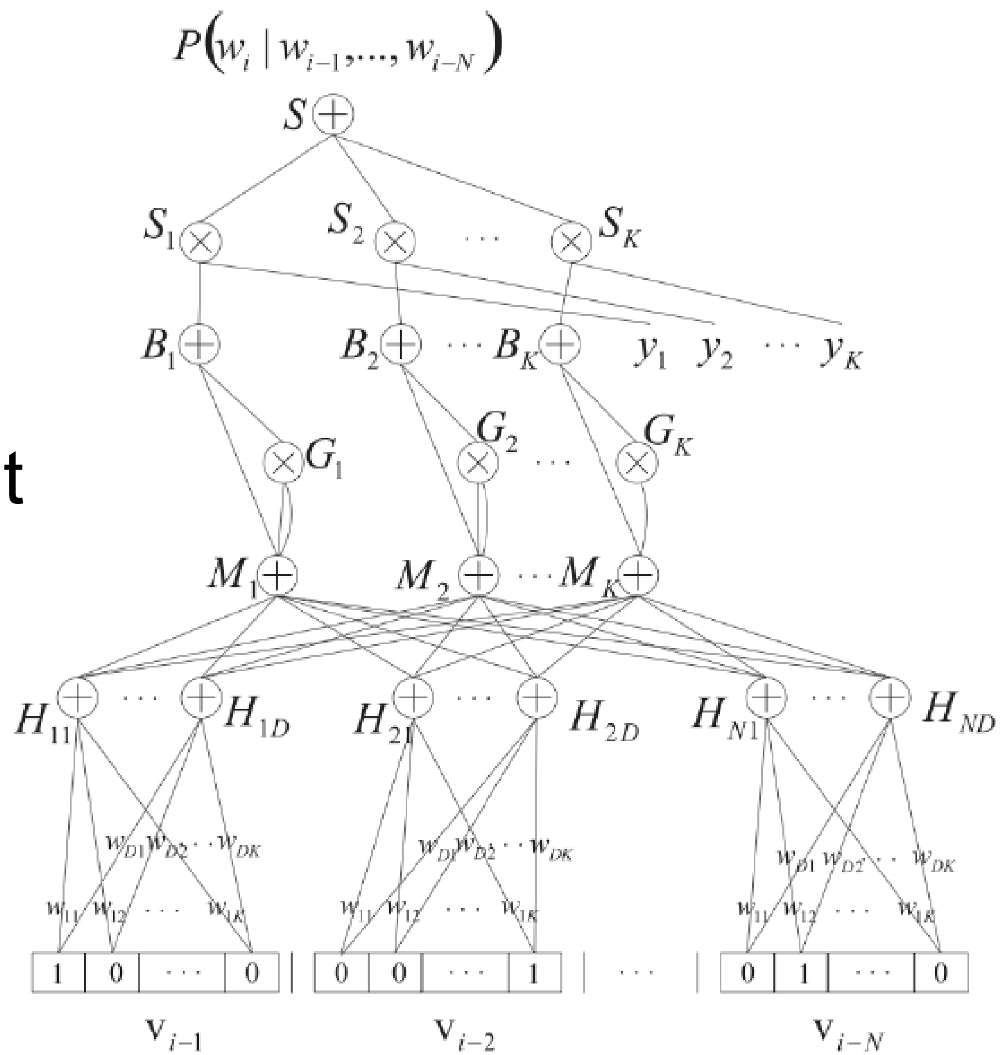
- Alternate between
  - Data Clustering: sum nodes
  - Variable partitioning: product nodes

Applications

- Image completion (Poon, Domingos; 2011)
- Activity recognition (Amer, Todorovic; 2012)
- **Language modeling (Cheng et al.; 2014)**
- Speech modeling (Perhaz et al.; 2014)

Language Model

- An SPN-based n-gram model
- Fixed structure
- Discriminative weight estimation by gradient descent



Results

- From Cheng et al. 2014

Table 1: Perplexity scores (PPL) of different language models.

Model	Individual PPL	+KN5
TrainingSetFrequency	528.4	
KN5 [3]	141.2	
Log-bilinear model [4]	144.5	115.2
Feedforward neural network [5]	140.2	116.7
Syntactical neural network [8]	131.3	110.0
RNN [6]	124.7	105.7
LDA-augmented RNN [9]	113.7	98.3
SPN-3	**104.2**	**82.0**
SPN-4	**107.6**	**82.4**
SPN-4'	**100.0**	**80.6**

Summary

- Sum-Product Networks
  - Deep architecture with clear semantics
  - Tractable probabilistic graphical model

- Going into more depth
  - SPN → BN [H. Zhao, M. Melibari, P. Poupart 2015]
  - Signomial framework for parameter learning [H. Zhao]
  - Online parameter learning: [A. Rashwan, H. Zhao]
  - SPNs for sequence data: [M. Melibari, P. Doshi]

SPN → Bayes Net

1. Normalize SPN
2. Create structure
3. Construct conditional distribution

Normal SPN

An SPN is said to be normal when

1. It is complete and decomposable
2. All weights are non-negative and the weights of the edges emanating from each sum node sum to 1.
3. Every terminal node in the SPN is a univariate distribution and the size of the scope of each sum node is at least 2.

Construct Bipartite Bayes Net

1. Create observable node for each observable variable
2. Create hidden node for each sum node
3. For each variable in the scope of a sum node, add a directed edge from the hidden node associated with the sum node to the observable node associated with the variable

Construct Conditional Distributions

1. Hidden node H: $\Pr(H = h_i) = w_i$
2. Observable node X: construct conditional distribution in the form of an algebraic decision diagram
   a. Extract sub-SPN of all nodes that contain X in their scope
   b. Remove the product nodes
   c. Replace each sum node by its corresponding hidden variable

Some Observations

- Deep SPNs can be converted into shallow BNs.

- The depth of an SPN is proportional to the height of the highest algebraic decision diagram in the corresponding BN.

Conversion Facts

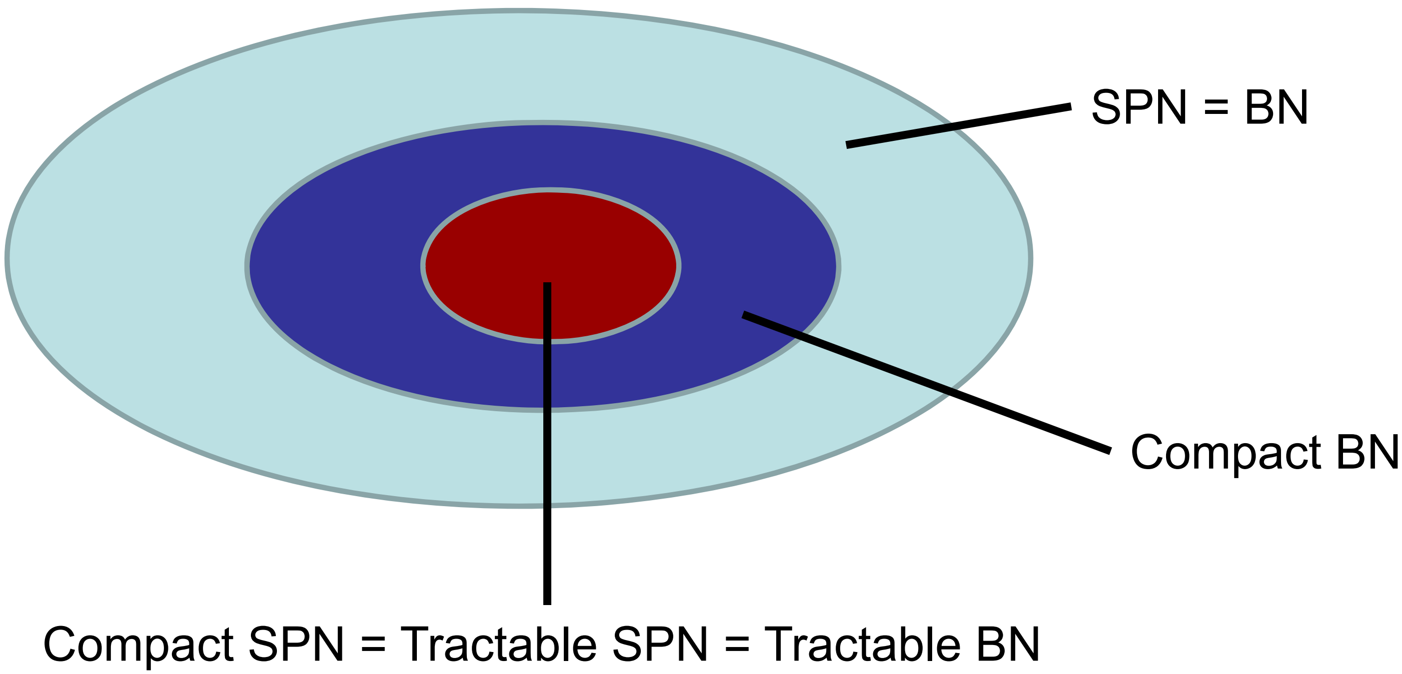
Thm 1: Any complete and decomposable SPN S over variables $X_1, \ldots, X_n$ can be converted into a BN B with ADD representation in time $O(N|S|)$. Furthermore S and B represent the same distribution and $|B| = O(N|S|)$.

Thm 2: Given any BN B with ADD representation generated from a complete and decomposable SPN S over variables $X_1, \ldots, X_n$, the original SPN S can be recovered by applying the variable elimination algorithm B in $O(N|S|)$.

Relationships

Probabilistic distributions

- Compact: space is polynomial in # of variables
- Tractable: inference time is polynomial in # of variables



Parameter Estimation

- Maximum Likelihood Estimation
- Online Bayesian Moment Matching

Maximum Log-Likelihood

- Objective: $w^* = argmax_{w \in R_+} \log \Pr(data|w)$
$$= argmax_{w \in R_+} \sum_x \log \Pr(x|w)$$

Where $\Pr(x|w) = \frac{f(e(x)|w)}{f(\mathbf{1}|w)}$

and $f(e(x)|w) = \sum_{tree \in e(x)} \prod_{ij \in tree} w_{ij}$

Non-Convex Optimization

$$\max_{w} \sum_{x} \log \sum_{tree \in e(x)} \prod_{ij \in tree} w_{ij} - \log \sum_{tree \in 1} \prod_{ij \in tree} w_{ij}$$

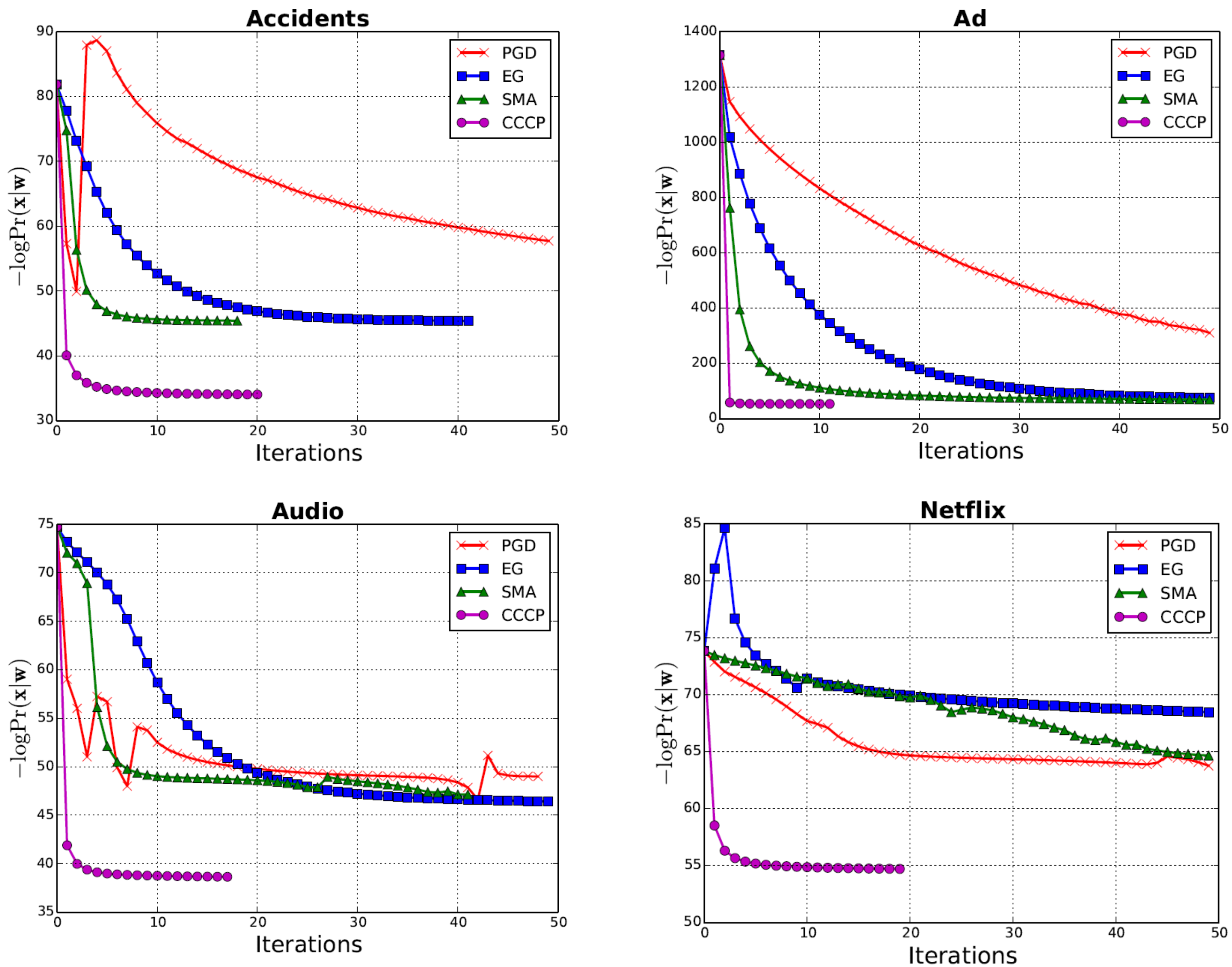
s.t. $w_{ij} \geq 0 \quad \forall ij$

- Approximations:
  - Projected gradient descent (PGD)
  - Exponential gradient (EG)
  - Sequential monomial approximation (SMA)
  - Convex concave procedure (CCCP = EM)

Summary

<table>
<tr><th>Algo</th><th>Var</th><th>Update</th><th>Approximation</th></tr>
<tr><td rowspan="2">PGD</td><td>w</td><td>additive</td><td>linear</td></tr>
<tr><td colspan="3">$w_{ij}^{k+1} \leftarrow projection\left(w_{ij}^{k} + \gamma\left[\frac{\partial \log f(e(x)|w)}{\partial w_{ij}} - \frac{\partial \log f(\mathbf{1}|w)}{\partial w_{ij}}\right]\right)$</td></tr>
<tr><td rowspan="2">EG</td><td>w</td><td>multiplicative</td><td>linear</td></tr>
<tr><td colspan="3">$w_{ij}^{k+1} \leftarrow w_{ij}^{k}\, exp\left(\gamma\left[\frac{\partial \log f(e(x)|w)}{\partial w_{ij}} - \frac{\partial \log f(\mathbf{1}|w)}{\partial w_{ij}}\right]\right)$</td></tr>
<tr><td rowspan="2">SMA</td><td>$\log w$</td><td>multiplicative</td><td>monomial</td></tr>
<tr><td colspan="3">$w_{ij}^{k+1} \leftarrow w_{ij}^{k} \exp\left(\gamma\left[\frac{\partial \log f(e(x)|w)}{\partial \log w_{ij}} - \frac{\partial \log f(\mathbf{1}|w)}{\partial \log w_{ij}}\right]\right)$</td></tr>
<tr><td rowspan="2">CCCP (EM)</td><td>$\log w$</td><td>multiplicative</td><td>Concave lower bound</td></tr>
<tr><td colspan="3">$w_{ij}^{k+1} \propto w_{ij}^{k} \frac{f_{v_j}(x|w^k)}{f(x|w^k)} \frac{\partial f(x|w^k)}{\partial f_{v_i}(x|w^k)}$</td></tr>
</table>

Results


Accidents
Ad
Audio
Netflix
PGD
EG
SMA
CCCP
−logPr(x|w)
Iterations

Scalability

- Online: process data sequentially once only
- Distributed: process subsets of data on different computers

- Mini-batches: online PGD, online EG, online SMA, online EM
- Problems: loss of information due to mini-batches, local optima, overfitting

- Can we do better?

Thomas Bayes

Bayesian Learning

- Bayes' theorem (1764)

$$\boxed{\Pr(\theta|X_{1:n}) \propto \Pr(\theta)\Pr(X_1|\theta)\Pr(X_2|\theta)\ldots\Pr(X_n|\theta)}$$

- Broderick et al. (2013): facilitates
  - **Online learning (streaming data)**

$$\Pr(\theta|X_{1:n}) \propto \Pr(\theta)\Pr(X_1|\theta)\Pr(X_2|\theta)\ldots\Pr(X_n|\theta)$$

  - **Distributed computation**

$$\underbrace{\Pr(\theta)\Pr(X_1|\theta)}_{\text{core \#1}}\underbrace{\Pr(X_2|\theta)\Pr(X_3|\theta)}_{\text{core \#2}}\underbrace{\Pr(X_4|\theta)\Pr(X_5|\theta)}_{\text{core \#3}}$$

Exact Bayesian Learning

- Assume a normal SPN where the weights $w_{i\cdot}$ of each sum node i form a discrete distribution.

- Prior: $\Pr(w) = \prod_{i\cdot} Dir(w_{i\cdot}|\alpha_{i\cdot})$

  where $Dir(w_{i\cdot}|\alpha_{i\cdot}) \propto \prod_j \left(w_{ij}\right)^{\alpha_{ij}}$
- Likelihood:
  $\Pr(x|w) = f(e(x)|w) = \sum_{tree \in e(x)} \prod_{ij \in tree} w_{ij}$

- Posterior:

Karl Pearson

Method of Moments (1894)

- Estimate model parameters by matching a subset of moments (i.e., mean and variance)

- Performance guarantees
  - Break through: First provably consistent estimation algorithm for several mixture models
    - HMMs: Hsu, Kakade, Zhang (2008)
    - MoGs: Moitra, Valiant (2010), Belkin, Sinha (2010)
    - LDA: Anandkumar, Foster, Hsu, Kakade, Liu (2012)

Bayesian Moment Matching for Sum Product Networks

Bayesian Learning
+
Method of Moments

Online, distributed and **tractable** algorithm for **SPNs**

Approximate **mixture of products of Dirichlets**
by a **single product of Dirichlets**
that **matches first and second order moments**

Moments

- Moment definition: $M_P\left(w_{ij}^k\right) = \int_w w_{ij}^k P(w)dw$
- Dirichlet: $Dir(w_{i\cdot}|\alpha_{i\cdot}) \propto \prod_{ij}\left(w_{ij}\right)^{\alpha_{ij}}$
  - Moments: $M_{Dir}\left(w_{ij}\right) = \frac{\alpha_{ij}}{\sum_j \alpha_{ij}}$

$$M_{Dir}\left(w_{ij}^2\right) = \left(\frac{\alpha_{ij}}{\sum_j \alpha_{ij}}\right)\left(\frac{\alpha_{ij}+1}{\sum_j \alpha_{ij}+1}\right)$$

  - Hyperparameters:

$$\alpha_{ij} = M_{Dir}\left(w_{ij}\right)\frac{M_{Dir}\left(w_{ij_1}\right)-M_{Dir}(w_{ij}^2)}{M_{Dir}\left(w_{ij_1}^2\right)-\left(M_{Dir}\left(w_{ij}\right)\right)^2}$$

Moment Matching

Recursive moment computation

- Compute $M_P(w_{ij}^k)$ of posterior $P(w|x)$ after observing x

$M_P(w_{ij}^k) \leftarrow computeMoment(node)$

If $isLeaf(node)$ then

Return leaf value

Else if $isProduct(node)$ then

Return $\prod_{child} computeMoment(child)$

Else if $isSum(node)$ and $node == i$ then

Return $\sum_{child} M_{Dir}(w_{ij}^k w_{i,child})\ computeMoment(child)$

Else

Return $\sum_{child} w_{node,child}\ computeMoment(child)$

Results (benchmarks)

Dataset	Var#	LearnSPN	oBMM	SGD	oEM	oEG
NLTCS	16	-6.11	**-6.07**	↓-8.76	↓-6.31	↓-6.85
MSNBC	17	-6.11	**-6.03**	↓-6.81	↓-6.64	↓-6.74
KDD	64	-2.18	**-2.14**	↓-44.53	↓-2.20	↓-2.34
PLANTS	69	-12.98	**-15.14**	↓-21.50	↓-17.68	↓-33.47
AUDIO	100	-40.50	**-40.7**	↓-49.35	↓-42.55	↓-46.31
JESTER	100	-53.48	**-53.86**	↓-63.89	↓-54.26	↓-59.48
NETFLIX	100	-57.33	**-57.99**	↓-64.27	↓-59.35	↓-64.48
ACCIDENTS	111	-30.04	**-42.66**	↓-53.69	-43.54	↓-45.59
RETAIL	135	-11.04	**-11.42**	↓-97.11	↓-11.42	↓-14.94
PUMSB-STAR	163	-24.78	**-45.27**	↓-128.48	↓-46.54	↓-51.84
DNA	180	-82.52	**-99.61**	↓-100.70	↓-100.10	↓-105.25
KOSAREK	190	-10.99	**-11.22**	↓-34.64	↓-11.87	↓-17.71
MSWEB	294	-10.25	**-11.33**	↓-59.63	↓-11.36	↓-20.69
BOOK	500	-35.89	**-35.55**	↓-249.28	↓-36.13	↓-42.95
MOVIE	500	-52.49	**-59.50**	↓-227.05	↓-64.76	↓-84.82
WEBKB	839	-158.20	**-165.57**	↓-338.01	↓-169.64	↓-179.34
REUTERS	889	-85.07	**-108.01**	↓-407.96	-108.10	↓-108.42
NEWSGROUP	910	-155.93	**-158.01**	↓-312.12	↓-160.41	↓-167.89
BBC	1058	-250.69	-275.43	↓-462.96	**-274.82**	↓-276.97
AD	1556	-19.73	**-63.81**	↓-638.43	↓-63.83	↓-64.11

Results (Large Datasets)

- Log likelihood

Dataset	Var#	LearnSPN	oBMM	oDMM	SGD	oEM	oEG
KOS	6906	-444.55	**-422.19**	-437.30	-3581.72	-452.02	-452.02
NIPS	12419	-	-1691.87	-1709.04	-6254.22	**-1495.63**	-3142.09
ENRON	28102	-	**-518.842**	-522.45	-	-	-
NYTIMES	102660	-	**-1503.65**	-1559.39	-	-	-

- Time (minutes)

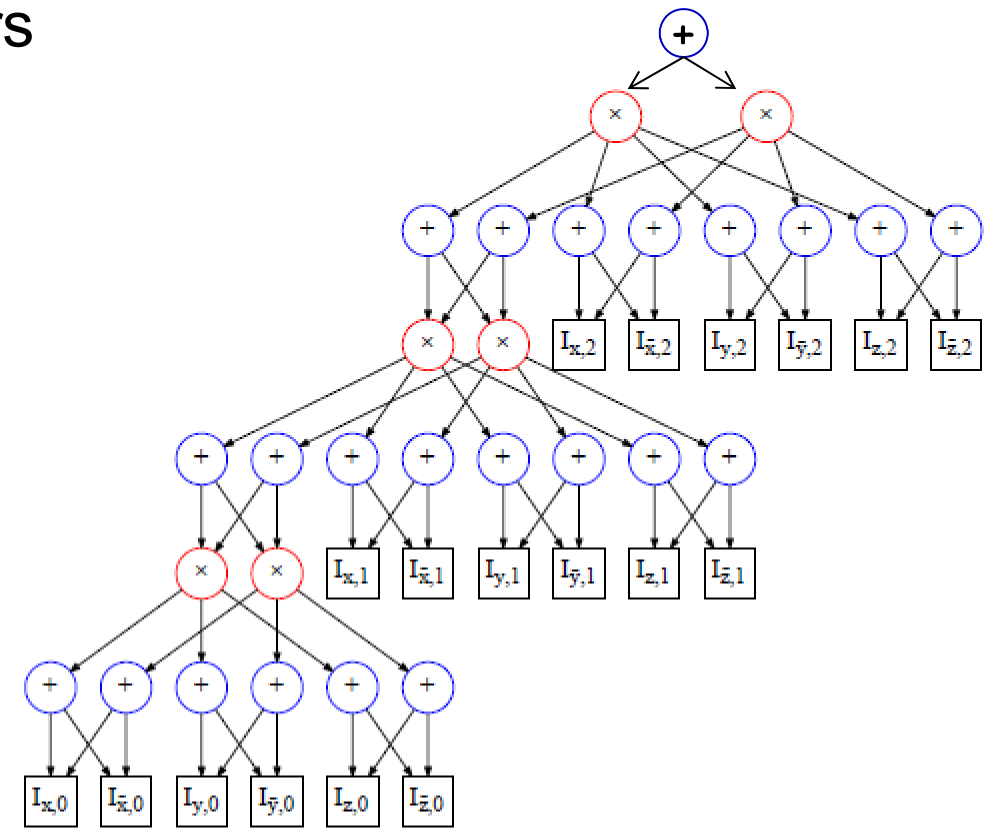
Dataset	Var#	LearnSPN	oBMM	oDMM	SGD	oEM	oEG
KOS	6906	1439.11	89.40	**8.66**	162.98	59.49	155.34
NIPS	12419	-	139.50	**9.43**	180.25	64.62	178.35
ENRON	28102	-	2018.05	**580.63**	-	-	-
NYTIMES	102660	-	12091.7	**1643.60**	-	-	-

Sequence Data

- How can we train an SPN with data sequences of varying length?
- Examples
  - Sentence modeling: sequence of words
  - Activity recognition: sequence of measurements
  - Weather prediction: time-series data
- Challenge: need structure that adapts to the length of the sequence while keeping # of parameters fixed

Dynamic SPN

- Idea: stack template networks with identical structure and parameters



Definitions

- **Dynamic Sum-Product Network: bottom network**, a stack of **template networks** and a **top network**

- **Bottom network:** directed acyclic graph with $2n$ indicator leaves and k roots that interface with the network above.

- **Top network:** rooted directed acyclic graph with k leaves that interface with the network below

- **Template network:** directed acyclic graph of k roots that interface with the network above, $2n$ indicator leaves and k additional leaves that interface with the network below.

Invariance

Let f be a bijective mapping that associates inputs to corresponding outputs in a template network

Invariance: a template network over $X_1, \dots, X_n$ is invariant when the scope of each interface node excludes $X_1, \dots, X_n$ and for all pairs of interface nodes i and j, the following properties hold:

- $scope(i) = scope(j)$ or $scope(i) \cap scope(j) = \emptyset$
- $scope(i) = scope(j) \Leftrightarrow scope(f(i)) = scope(f(j))$
- $scope(i) \cap scope(j) = \emptyset \Leftrightarrow scope(f(i)) \cap scope(f(j)) = \emptyset$
- All interior and output sum nodes are complete
- All interior and output product nodes are decomposable

Completeness and Decomposability

Theorem 1: If

a. the bottom network is complete and decomposable,
b. the scopes of all pairs of output interface nodes of the bottom network are either identical or disjoint,
c. the scopes of the output interface nodes of the bottom network can be used to assign scopes to the input interface nodes of the template and top networks in such a way that the template network is invariant and the top network is complete and decomposable,

then the **DSPN is complete and decomposable**

Structure Learning

Anytime search-and-score framework

Input: data, variables $X_1, \dots, X_n$

Output: $templateNet$

$templateNet \leftarrow initialStructure(data, X_1, \dots, X_n)$

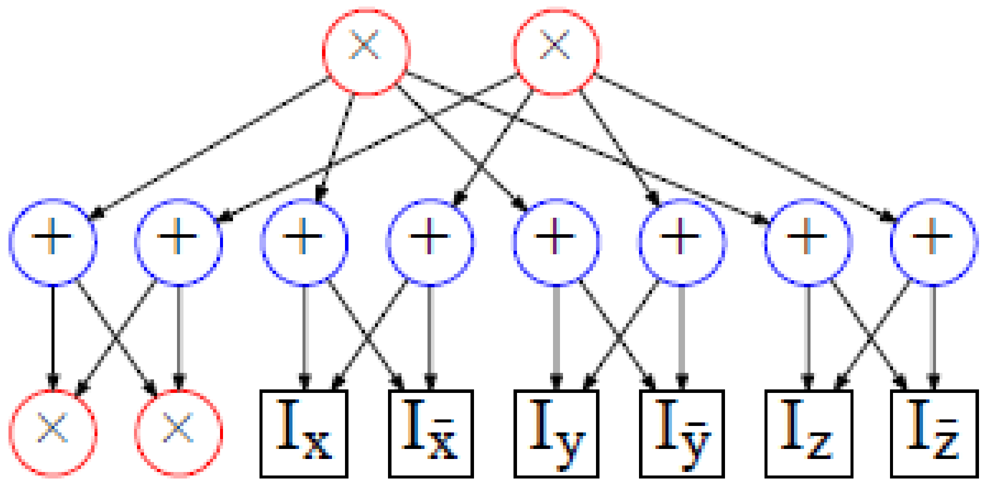
Repeat

$templateNet \leftarrow neighbour(templateNet, data)$

Until stopping criterion is met

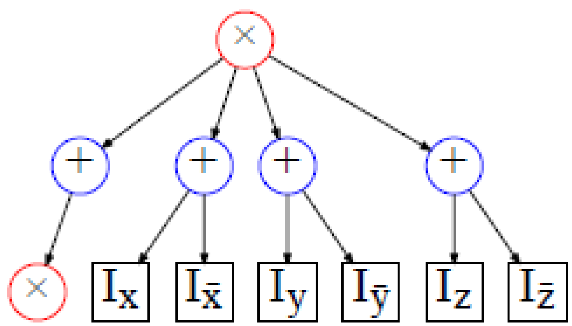
Initial Structure

- Factorized model of univariate distributions

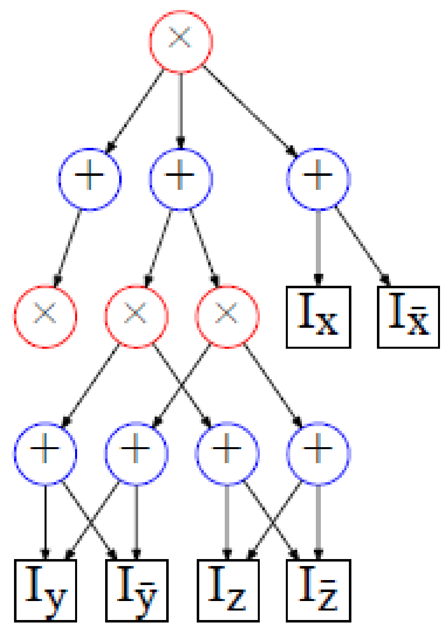


Neighbour generation

- Replace sub-SPN rooted at a product node by a product of Naïve Bayes modes



(a)



(b)

Results

Table 1: Statistics of the datasets used in our experiments.

Dataset	# Instances	Sequence length	# of Obs. variables
HMM-Samples	100	100	1
Water	100	100	4
BAT	100	100	10
Pen-Based Digits	10992	16	7
EEG Eye State	14980	15	1
Spoken Arabic Digit	8800	40	13
Hill-Valley	606	100	1
Japanese Vowels	640	16	12

Table 2: Mean log-likelihood and standard error for the synthetic datasets.

Dataset	True Model LL	LearnSPN LL	DSPN LL
HMM-Samples	-62.2015 $\pm$ 0.8449	-65.3996 $\pm$ 0.7081	**-62.5982** $\pm$ 0.7362
Water	-249.5736 $\pm$ 1.0241	-270.3871 $\pm$ 0.9422	**-252.3607** $\pm$ 0.8958
BAT	-628.1721 $\pm$ 1.9802	-684.3833 $\pm$ 1.3088	**-641.5974** $\pm$ 1.1176

Results

Dataset	HMM Training	Reveal Training	DSPN Training
Pen-Based Digits	-74.3763 $\pm$ 0.1493	-74.1533 $\pm$ 0.2643	**-63.2376** $\pm$ 0.6727
EEG Eye State	-8.1381 $\pm$ 0.1265	-7.8332 $\pm$ 0.0134	**-7.5216** $\pm$ 0.1774
Spoken Arabic Digit	-323.4032$\pm$ 0.4752	-256.6012$\pm$ 0.2028	**-252.2177** $\pm$ 0.3404
Hill-Valley	-69.7490 $\pm$ 0.2071	-67.7216 $\pm$ 0.0135	**-63.2722** $\pm$ 0.1614
Japanese Vowels	-94.8432 $\pm$ 0.3931	-69.7882 $\pm$ 0.1023	**-66.3305** $\pm$ 0.2942

Dataset	HMM Testing	Reveal Testing	DSPN Testing
Pen-Based Digits	-74.1607 $\pm$ 0.1208	-74.3826 $\pm$ 0.2425	**-63.4597** $\pm$ 0.2794
EEG Eye State	-8.4959 $\pm$ 0.2579	-7.8433 $\pm$ 0.0252	**-7.2508** $\pm$ 0.1031
Spoken Arabic Digit	-327.4504$\pm$ 0.4342	-260.2027$\pm$ 0.9617	**-257.8612** $\pm$ 0.5031
Hill-Valley	-69.7613 $\pm$ 0.1755	-67.7253 $\pm$ 0.0741	**-63.3698** $\pm$ 0.3068
Japanese Vowels	-94.2505 $\pm$ 0.2981	-71.3435 $\pm$ 1.2324	**-68.7529** $\pm$ 0.2688

Conclusion

- Sum-Product Networks
  - Deep architecture with clear semantics
  - Tractable probabilistic graphical model
- Future work
  - Decision SPNs: M. Melibari and P. Doshi
- Open problem:
  - Thorough comparison of SPNs to other deep networks