A Co-Clustering approach for Sum-Product Network Structure Learning

Antonio Vergari
Nicola Di Mauro
Floriana Esposito

December 8, 2014
Outline

- Introducing **Sum-Product Networks (SPNs)**
  - definition and *properties*
  - inference

- **Learning the structure** of SPNs
  - current state of the art
  - affinities with *hierarchical co-clustering*
  - sketching a proposal

- Experimentation
  - experimental setting
  - results

- Further Works and Conclusions
Probabilistic Graphical Models

They can compactly represent joint probability distributions...

...but inference is potentially intractable, *exponential in the treewidth*. One of the bottlenecks is the computation of the *partition function*:

\[
Z = \sum_{x \sim X} \prod_{C \in \mathcal{C}} \phi_C(x_C)
\]
Sum-Product Networks

A tractable distribution is an SPN.

A tractable distribution is an SPN.
A product of SPNs over different scopes is an SPN (*decomposability*).

---

Sum-Product Networks

A tractable distribution is an SPN.

A product of SPNs over different scopes is an SPN (*decomposability*).

A weighted sum (*w*₂ ≥ 0) of SPN over the same scope is an SPN (*completeness*).

Nothing else is an SPN.

Sum-Product Networks

A tractable distribution is an SPN. A product of SPNs over different scopes is an SPN (*decomposability*). A weighted sum \( (w_i \geq 0) \) of SPN over the same scope is an SPN (*completeness*). Nothing else is an SPN. They compactly encode the *network polynomial* over \( \mathbf{X} \) (multilinear function).

Inference

Given an SPN $S$ over r.v.s $\mathbf{X}$, these measures are computable in time linear to $|\text{edges}(S)|$:

- the partition function of the distribution over $\mathbf{X}$:
  \[ Z = S(\ast) \]

- exact marginal probabilities for an evidence $e$:
  \[ Pr(e) = \frac{S(e)}{S(\ast)} \]

- MPE probability for an evidence $e$ and query $q$ (converting $S$ into $S^{\max}$ by replacing sum nodes with max nodes)
  \[ \text{MPE}(q, e) = \max_q Pr(q, e) = S^{\max}(e) \]
Inference (example)

To compute the *marginal* probability $Pr(X) = \sum_{y \sim Y} Pr(X, y)$:

![Diagram representing the computation of marginal probability in a sum-product network]
Inference (example)

To compute the *marginal* probability $Pr(X) = \sum_{y \sim Y} Pr(X, y)$:

- Set the leaf values for $X$ and marginalize over $Y$ by setting their probabilities to 1.
Inference (example)

To compute the *marginal* probability \( Pr(X) = \sum_{y \sim Y} Pr(X, y) \):

- Set the leaf values for \( X \) and marginalize over \( Y \) by setting their probabilities to 1
- Propagate probabilities (computing sums and products *bottom-up*)
Inference (example)

To compute the *marginal* probability $Pr(X) = \sum_{y \sim Y} Pr(X, y)$:

- Set the leaf values for $X$ and marginalize over $Y$ by setting their probabilities to 1.
- Propagate probabilities (computing sums and products *bottom-up*).
- The exact values is the root value.
Deep Architectures

Exploit *local interactions* to save computations by layering a *deep architecture*.

Learning a compact structure equals to discover these interactions.
Structure Learning

Building the network *top down* or *bottom up* by clustering features and/or instances:

- **KMeans** on features ($k = 2$, euclidean distance) to discover similarities, adding layers of sum nodes (fixed length, fully connected) [Dennis and Ventura 2012]

- Merging feature regions bottom-up by a **Bayesian-Dirichlet independence test**, adding sum layers (fixed length) and reducing edges by maximizing Mutual Information (Information Bottleneck) [Peharz, Geiger, and Pernkopf 2013]

- LearnSPN: alternating **splitting** instances (clustering by similarity) and features (independence checking) in a **greedy way**.
  Builds a *tree-like* SPN with univariate distributions at the leaves estimated from data [Gens and Domingos 2013]
LearnSPN

Build a tree-like SPN that maximizes the log-likelihood of the data by alternating splits on instances and features.

**Online Hard-EM** with restarts to cluster instances $T$ with an exp prior ($Pr(C_i) \propto e^{-\lambda |C_i| \cdot |X|}$) on clusters to avoid overfitting:

$$
Pr(X) = \sum_{C_i \in C} \prod_{X_j \in X} Pr(X_j | C_i) Pr(C_i)
$$

found clusters become *sum node children* with parameters $w_i = |C_i|/|T|$.

Features $X$ are clustered into independent components via a greedy procedure based on a **G Test** over pairs $X_i, X_j$:

$$
G(X_i, X_j) = 2 \sum_{x_i \sim X_i} \sum_{x_j \sim X_j} c(x_i, x_j) \cdot \log \frac{c(x_i, x_j) \cdot |T|}{c(x_i) c(x_j)}
$$

which are associated to **product node children**.

If $|T| < m$ consider features independent and create *leaves* by **smooth laplacian frequency estimation** (with parameter $\alpha$).
LearnSPN (example)
LearnSPN (example)
LearnSPN (example)
A Co-Clustering Analogy

In the end, LearnSPN builds a network by partitioning the data matrix into blocks by highlighting local interactions.

One could discover both row and column interactions by the means of a co-clustering technique (co-clusters as blocks in the partitioning).

To preserve the decomposability and completeness of the resulting SPN:

- co-clusters shall be non overlapping
- the union of the co-clusters shall reconstruct the whole data matrix

To have a deep architecture one needs a hierarchy of co-clusters.

Such several out-of-the box co-clustering algorithms could be used to build SPNs... ...but rows are have not the same meaning of columns! (see caveat).
From a Hierarchical Co-Clustering to an SPN

Considering a co-cluster hierarchy as represented by two cluster hierarchies (on rows and columns), the construction uses the two hierarchies and the matrix decomposition intuition from LearnSPN.

Traverse the hierarchies top-down for at most $k_{max}$ levels (to avoid overfitting) and:

- add a *sum node* over a split in the row hierarchy estimating the child weights as the proportions of instances in the split
- add a *product node* over a split in the column hierarchy
- if a split contains less than $m$ instances add a product node over single column splits
- if a split contains only a feature, add a *leaf node* estimating the univariate distribution with a Laplacian smoothing
Hierarchies translation (example)
Hierarchies translation (example)
Hierarchies translation (example)
Hierarchies translation (example)
Co-clustering & SPNs: caveats

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

X Y Z W

---

X Y Z W

---

X Y Z W

---

X Y Z W
HiCC-SPN I

First co-clustering algorithm used in the experimentation: **HiCC Hierarchical Incremental Co-Clustering** [Ienco, Pensa, and Meo 2009]

- almost parameter-free (only the number of iterations to get initial clusters)
- optimizing an association measure, **Goodman-Kruskal's** $\tau$ divergence:

$$\tau_{C_R|c_C} = \frac{E_{C_R} - E_{C_R|c_C}}{E_{C_R}}$$

- builds an initial co-clustering then recursively splits each partition (yielding two hierarchies)
- using a Stochastic Local Search approach to find an optimal partitioning:
  - moving an element (row or column) from one cluster to another (cluster creation or deletion are allowed)
  - conditioning on the previous clustering on columns (and vice versa)
  - using restarts for hierarchy levels to differentiate the search
HiCC-SPN II

Adding a constraint to column splits to preserve independence by checking a \textit{conditional mutual information} criterion.

Given a row clustering $\mathbf{R}$, we allow feature $x$ (column) to be moved from cluster $C_i$ to $C_j$ whether:

$$\frac{1}{|C_i|} \sum_{x_i \in C_i} I(x; x_i | \mathbf{R}) > \frac{1}{|C_j - 1|} \sum_{x \neq x_j \in C_j} I(x; x_j | \mathbf{R}).$$

$$I(x; y | \mathbf{R}) = H(x | \mathbf{R}) - H(x | y, \mathbf{R})$$

$$= \sum_{R \in \mathbf{R}} p(R) \sum_{x_i \sim x} \sum_{y_j \sim y} p(x_i, y_j | R) \log \frac{p(x_i, y_j | R)}{p(x_i | R)p(y_j | R)}.$$

where $p(R)$, the prior on the cluster $R$ on the row partitioning $\mathbf{R}$ is estimated as the cluster size proportion and $p(x_i, y_j | R)$, $p(x_i | R)$ and $p(y_j | R)$ are estimated as the frequencies of the values seen in $R$. 
Experiments

Comparing LearnSPN ans HiCC-SPN in the *generative* framework of graphical models structure learning [Gens and Domingos 2013]:

- comparing the *average log-likelihood* on predicting instances from a test set

- 11 different datasets with binary features, standard in PGMs comparisons [Lowd and Davis 2010] [Haaren and Davis 2012]
  - ranging from classification, recommending, frequent pattern mining
  - 16 to 500 features, 1600 to 22000 instances, 0.01 to 0.5 density
  - Training 75% Validation 10% Test 15% (no cv)

- Model selection via *grid search* in this parameter space:
  - for LearnSPN: $\lambda \in \{0.2, 0.6, 0.8\}$, $\rho \in \{5, 10, 20\}$, $m \in \{1, 100, 400\}$, $\alpha \in \{0.01, 0.1, 1.0\}$
  - for HiCC-SPN: $k \in \{1 : 10\}$, $m \in \{1, 100, 400\}$, $\alpha \in \{0.001, 0.01, 0.1\}$
## Experiment #1: results

|      | $|X|$ | $|T_{tr}|$ | $|T_v|$ | $|T_{te}|$ | LearnSPN       | HiCC-SPN       |
|------|------|---------|--------|--------|---------------|---------------|
| NLTCS| 16   | 16181   | 2157   | 3236   | $-6.111 \pm 3.11$ | $-6.150 \pm 3.11$ |
| Plants| 69   | 17412   | 2321   | 3482   | $-12.942 \pm 8.52$ | $-13.762 \pm 9.70$ |
| Audio| 100  | 15000   | 2000   | 3000   | $-40.465 \pm 16.05$ | $-44.924 \pm 18.47$ |
| Jester| 100  | 9000    | 1000   | 4116   | $-53.605 \pm 13.29$ | $-56.460 \pm 10.82$ |
| Netflix| 100  | 15000   | 2000   | 3000   | $-57.353 \pm 6.29$ | $-62.730 \pm 5.90$ |
| Accidents| 111  | 12758   | 1700   | 2551   | $-36.306 \pm 6.00$ | $-42.790 \pm 6.40$ |
| Retail| 135  | 22041   | 2938   | 4408   | $-11.053 \pm 7.16$ | $-11.064 \pm 7.27$ |
| Pumsb-star| 163  | 12262   | 1635   | 2452   | $-24.512 \pm 8.07$ | $-37.854 \pm 12.69$ |
| DNA| 180  | 1600    | 400    | 1186   | $-83.986 \pm 11.30$ | $-99.273 \pm 6.7$ |
| Book| 500  | 8700    | 1159   | 1739   | $-35.888 \pm 40.94$ | $-36.939 \pm 47.33$ |
| EachMovie| 500  | 4525    | 1002   | 591    | $-52.685 \pm 53.78$ | $-55.411 \pm 56.35$ |
Conclusions and Further Works

Translating a co-cluster hierarchy into an SPN could be promising, the exact and tractable inference could be derived given a row and column partitioning...

...but not every co-clustering algorithm is directly usable, plus some representational issues have to be taken into account.

We are now investigating nested partition models [Rodriguez and Ghosh 2012] that allow for guillotine splits of the data matrix in order to better capture the underlying latent interactions and have deeper insights into SPN structure learning.
References


Discussion

References