Approximate Inference via Compilation to Arithmetic Circuits

Work by Daniel Lowd and Pedro Domingos
Overview

• Arithmetic circuits (ACs) allow for exact inference in networks with high treewidth by exploiting context-specific independence and determinism

• This work introduces *approximate* inference methods using ACs
Network Polynomial

- Defines a Bayesian network as a multilinear function
- $\lambda_x$ - Evidence indicators: used to select the a value
- $\theta_{x|u}$ - Network parameters: parameters from the CPTs.

$$f = \sum_x \prod_{xu \sim x} \lambda_x \theta_{x|u}$$
Arithmetic Circuit

• Rooted, directed acyclic graph
• Leaves - numeric constants/variables
• Interior nodes - Addition and multiplication operations
Consider the polynomial

\[ f = \lambda_a \lambda_b \lambda_c \theta_a \theta_{b|a} \theta_{c|a} + \lambda_a \lambda_b \lambda_c \theta_a \theta_{b|a} \theta_{c|a} + \ldots + \lambda_a \lambda_b \lambda_c \theta_a \theta_{b|a} \theta_{c|a}. \]

We close this section by noting that [Russell et al.] has observed that one can represent such polynomials easily. Let \( \lambda \) denote the compatibility relation among instantiations. The multi-linear function of a Bayesian network is a multi-variate function that is used to represent the conditional probability distribution specified by a Bayesian network. The multi-linear function for this network is:

| \( A \) | \( B \) | \( \theta_{B|A} \) | \( A \) | \( C \) | \( \theta_{C|A} \) |
|---|---|---|---|---|---|
| true | true | 1 | true | true | .8 |
| true | false | 0 | true | false | .2 |
| false | true | 0 | false | true | .2 |
| false | false | 1 | false | false | .8 |

We also show in Section \( t \) that the partial derivatives of the network polynomial are:

\[ \frac{\partial}{\partial a} \]

But we show in Section \( u \) that one can represent such polynomials easily. The outer sum in the above definition ranges over all instantiations. Table 1. Partial derivatives of the network polynomial

(Figures from Darwiche 2003)
Context-Specific Independence

- Take advantage of context-specific independencies
- Variables may be independent given a certain instantiation
- Use a decision-tree CPD representation
- Arithmetic circuits can also capture these independencies

(Figure from Boutilier et al. 1996)
Log-Linear Model Representation

- $f_i$ - feature: represents an assignment $i$ to a configuration
- $w_i$ - weight: represents the value corresponding to assignment $i$

$$\log P(X = x) = -\log Z + \sum_i w_i f_i(x)$$
KL Divergence

- Average log difference between two distributions
- $Q$ is typically the approximate distribution, while $P$ is the true distribution
- In variational methods, the goal is to find a tractable distribution $Q$ that minimizes this

$$
\text{KL}(Q \parallel P) = \sum_x Q(x) \log \frac{Q(x)}{P(x)} = -H_Q(x) - \sum_i w_i E_Q[f_i] + \log Z_P
$$
Approximate Compilation

• Main idea: Compile an AC that approximates the true BN and perform using that AC

• Two stages
  • Structure search (Performed once)
  • Parameter optimization (Fine tunes the circuit to specific evidence)
Structure Search

• Two methods covered in this work
  • Prune and compile
  • Learn from samples
Decision Tree CPD Structure Learning

- Split operators
  - Complete - $C(v, \pi)$ add leaf nodes as children to $v$, where each leaf corresponds to an assignment of $\pi$
  - Binary - $B(v, \pi, k)$ adds 2 leaf nodes as children to $v$, where one leaf corresponds to assignment $k$ of $\pi$ and the other for all other assignments
  - Merge - $M(v_1, v_2)$ merges 2 leaf nodes; resulting node inherits all parents from both.
Decision Tree CPD Structure Learning

\[ v_1 = P(z \mid y=0) \]
\[ v_2 = P(z \mid y=1) \]
\[ v_3 = P(z \mid y=2) \]

\( C(v_3, x) \)  \( B(v_3, x, 0) \)  \( M(v_2, v_3) \)
Prune and Compile

- Goal: Find a simplified network Q from pruning splits in the decision tree CPD
- Q’s structure is a subset of P’s - can decompose KL divergence
- Need to compute parent distributions P(\(\pi_i\))
  - Intractable
  - Need to approximate
    - P-Samp: Estimate joint distributions with samples
    - P-MF: Mean field

\[
\text{KL}(P \parallel Q) = \sum_i \sum_{\pi_i} P(\pi_i) \sum_{x_i} P(x_i|\pi_i) \log \frac{P(x_i|\pi_i)}{Q(x_i|\pi_i)}
\]
Prune and Compile

- Algorithm
  - Initialize with fully pruned network
  - Add in splits greedily that best decrease the KL divergence
  - Every 10 splits, compile the network to an AC check the number of edges
  - Stop if this value exceeds a pre-specified bound
Learning from Samples

- Generate samples with forward sampling
- Use samples as data for learning AC structure directly
Learning from Samples

- The LearnAC algorithm (Lowd and Domingos, UAI 2008)
  - Initialize circuit as a product of marginals (a BN with no edges)
  - Apply AC splits that add the fewest edges while increasing a score function
    - The score function computes the log-likelihood of the training data, with penalties on the number of edges and parameters
    - Modified in this work to stop when a size limit is reached
Parameter Optimization

• Three methods
  • Forward sampling
  • Variational optimization
  • Gibbs sampling
AC²-F

- Generate a set of samples from the original BN
- Use maximum likelihood estimation to set AC parameters
- Can be viewed as approximately minimizing KL(P||Q)
- For conditional queries, we are more interested in the divergence of the conditional distributions, KL(P( . | x_{ev})||Q( . | x_{ev})
- Expected to perform more poorly on rare evidence (see bound below)

\[
KL(P( . | x_{ev}) \parallel Q( . | x_{ev})) \leq \frac{1}{P(x_{ev})} KL(P \parallel Q)
\]
$AC^2-V$

- Try to address the problem when conditioning on poor evidence
- Minimize $KL(Q||P)$ where $P$ is the original BN conditioned on evidence
AC$^2$-V

- Representing $P$ and $Q$ as log-linear models...

  $\log P(x) = -\log Z_P + \sum_i w_i f_i(x)$  \hspace{1cm}  $\log Q(x) = -\log Z_Q + \sum_j v_j g_j(x)$

- KL divergence and gradient

  $\text{KL}(Q \| P) = \sum_j v_j E_Q(g_j) - \sum_i w_i E_Q(f_i) + \log \frac{Z_P}{Z_Q}$

  $\frac{\partial}{\partial v_j} \text{KL}(Q \| P) = \sum_k v_k (E_Q(g_k g_j) - Q(g_k) Q(g_j)) - \sum_i v_i (E_Q(f_i g_j) - Q(f_i) Q(g_j))$
AC$^2$-G

- Minimize $\text{KL}(P\|Q)$, since expectations over $Q$ may assign small or zero probabilities to important modes of $P$
- Approximate the expectations of $P$ with Gibbs sampling

$$\text{KL}(P\|Q) = \sum_i w_i E_P(f_i) - \sum_j v_j E_P(g_j) + \log Z_Q / Z_P$$

$$\frac{\partial}{\partial v_j} \text{KL}(P\|Q) = E_Q(g_j) - E_P(g_j)$$
Experiments

• Structure Selection
  • Compute KL divergences (used random samples)
  • Prune and compile (P-Samp and P-MF)
  • Chow-Liu Trees
  • LearnAC
  • Parameters set using AC²-F
  • Training times ranged from 17 minutes to 8 hours

\[ D(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} = E_P[\log(P(x)/Q(x))] \approx \frac{1}{m} \sum_i \log(P(x^{(i)})/Q(x^{(i)})) \]
Experiments

<table>
<thead>
<tr>
<th></th>
<th>P-MF</th>
<th>P-Samp</th>
<th>C-L</th>
<th>LAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDD Cup</td>
<td>2.44</td>
<td>0.10</td>
<td>0.23</td>
<td>0.07</td>
</tr>
<tr>
<td>Plants</td>
<td>8.41</td>
<td>2.29</td>
<td>4.48</td>
<td>1.27</td>
</tr>
<tr>
<td>Audio</td>
<td>4.99</td>
<td>3.31</td>
<td>4.47</td>
<td>2.12</td>
</tr>
<tr>
<td>Jester</td>
<td>5.14</td>
<td>3.55</td>
<td>5.08</td>
<td>2.82</td>
</tr>
<tr>
<td>Netflix</td>
<td>3.83</td>
<td>3.06</td>
<td>4.14</td>
<td>2.24</td>
</tr>
<tr>
<td>MSWeb</td>
<td>1.78</td>
<td>0.52</td>
<td>0.70</td>
<td>0.38</td>
</tr>
<tr>
<td>Book</td>
<td>4.90</td>
<td>2.43</td>
<td>2.84</td>
<td>1.89</td>
</tr>
<tr>
<td>EachMovie</td>
<td>29.66</td>
<td>17.61</td>
<td>17.11</td>
<td>11.12</td>
</tr>
</tbody>
</table>
Experiments

• Generated 100 random samples
• Select random subset of variables to use as evidence
  • Generate additional samples (up to 100,000) and pick out ones consistent with the evidence in the 100 samples.
• Measure the log conditional probability of non-evidence variables of the samples
• Serves as an approximation of the KL divergence between the true and inferred conditional distributions
# Experiments

- Mean Query Times

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AC$^2$-F</th>
<th>AC$^2$-V</th>
<th>AC$^2$-G</th>
<th>BP</th>
<th>MF</th>
<th>Gibbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDD Cup</td>
<td>0.022</td>
<td>3803</td>
<td>11.2</td>
<td>0.050</td>
<td>0.025</td>
<td>2.5</td>
</tr>
<tr>
<td>Plants</td>
<td>0.022</td>
<td>2741</td>
<td>11.2</td>
<td>0.081</td>
<td>0.073</td>
<td>2.8</td>
</tr>
<tr>
<td>Audio</td>
<td>0.023</td>
<td>4184</td>
<td>14.4</td>
<td>0.063</td>
<td>0.048</td>
<td>3.4</td>
</tr>
<tr>
<td>Jester</td>
<td>0.019</td>
<td>3448</td>
<td>13.8</td>
<td>0.054</td>
<td>0.057</td>
<td>3.3</td>
</tr>
<tr>
<td>Netflix</td>
<td>0.021</td>
<td>3050</td>
<td>12.3</td>
<td>0.057</td>
<td>0.053</td>
<td>3.3</td>
</tr>
<tr>
<td>MSWeb</td>
<td>0.022</td>
<td>2831</td>
<td>12.2</td>
<td>0.277</td>
<td>0.046</td>
<td>4.3</td>
</tr>
<tr>
<td>Book</td>
<td>0.020</td>
<td>5190</td>
<td>16.1</td>
<td>0.864</td>
<td>0.059</td>
<td>6.6</td>
</tr>
<tr>
<td>EachMovie</td>
<td>0.022</td>
<td>10204</td>
<td>28.6</td>
<td>1.441</td>
<td>0.342</td>
<td>11.0</td>
</tr>
</tbody>
</table>
Experiments

Figure 1: Average conditional log likelihood of the query variables (y axis), divided by the number of query variables (x axis). Higher is better. Gibbs often performs too badly to appear in the frame.
Experiments

- $\text{AC}^2$-F: Good for fast inference on with small to moderate amounts of evidence
- $\text{AC}^2$-V: Better when there is more evidence
  - Reducing $\text{KL}(Q||P)$ may increase $\text{KL}(P||Q)$
  - Slower than the other algorithms
- $\text{AC}^2$-G: Most accurate
  - Dominates BP, MF, and Gibbs on the datasets
  - Takes longer than Gibbs due to parameter optimization step and computations of expectations
Conclusion

• ACs are alternatives to junction trees
  • Exploits determinism and context-specific independence
• Combining sampling and learning is a good strategy for accurate approximate inference
  • sampling - get a coarse approximation
  • learning - smooth the approximation