Discriminative Learning of Sum-Product Networks

Robert Gens
Pedro Domingos
Distributions
Mixtures
Motivation

SPN Review

Discriminative Training

Experiments
Motivation

SPN Review

Discriminative Training

Experiments
Graphical Models

SPNs perform fast, exact inference on high treewidth models.
Deep Architectures

SPNs have full probabilistic semantics and tractable inference over many layers
Discriminative Learning

SPNs combine features with fast, exact inference over high treewidth models.
Motivation

SPN Review

Discriminative Training

Experiments
A Univariate Distribution
Is an SPN.

X

Multinomial  Gaussian  Poisson  ...
A Product of SPNs over Disjoint Variables Is an SPN.
A Weighted Sum of SPNs over the Same Variables Is an SPN.

Sums out a mixture variable
All Marginals Are Computable in Linear Time
All Marginals Are Computable in Linear Time
All Marginals Are Computable in Linear Time
All Marginals Are Computable in Linear Time

\[ P(X=0) ? \]
All Marginals Are Computable in Linear Time

\[ P(X=0) ? \]
All Marginals Are Computable in Linear Time

\[ P(X=0) \, ? \]
All Marginals Are Computable in Linear Time

\[ P(X=0) = 0.26 \]

Evidence

Marginalize

Evidence

Marginalize

0.5

0.4

0.6

0.1
All MAP States Are Computable in Linear Time

\[ \max_y P(X=0, Y=y) \]
All MAP States Are Computable in Linear Time

\[ \max_y P(X=0, Y=y) \]
All MAP States Are Computable in Linear Time

\[
\max_y P(X=0, Y=y) \ ?
\]
All MAP States Are Computable in Linear Time

\[ \max_y P(X=0, Y=y) \]
All MAP States Are Computable in Linear Time

\[
\max_y P(X=0, Y=y) = 0.12
\]
All MAP States Are Computable in Linear Time

$$\max_y P(X=0, Y=y) = 0.12$$
Special Cases of SPNs

• Junction trees
• Hierarchical mixture models
• Non-recursive probabilistic context-free grammars
• Models with context-specific independence
• Models with determinism
• Other high-treewidth models
Compactly Represented Probability Distributions

Graphical Models

Sum-Product Networks

Existing Tractable Models
## Learning SPNs

<table>
<thead>
<tr>
<th>Update</th>
<th>Soft Inference (Marginals)</th>
<th>Hard Inference (MAP States)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen. EM</td>
<td>🟢</td>
<td>🟢</td>
</tr>
<tr>
<td>Gen. Gradient</td>
<td>🟢</td>
<td></td>
</tr>
<tr>
<td>Disc. Gradient</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Poon & Domingos, UAI 2011
Motivation

SPN Review

Discriminative Training

Experiments
Discriminative SPNs

\[ P(Y|X) \]

- \( Y \) Query
- \( H \) Hidden
- \( X \) Evidence

Treat as constants
Discriminative SPNs

Y

X

H
Hidden

Y
Query

X
Evidence

\[\begin{align*}
&Y_1\quad \times \quad 0.4 \quad \times \\
&Y_1\quad \times \quad 0.1 \quad \times \\
&Y_2\quad \times \\
&Y_2
\end{align*}\]
Discriminative SPNs

\[ f(X) = f_1(X)f_2(X) \]

- **Y** Y1, Y1, Y2, Y2
- **H** Hidden
- **Y** Query
- **f(X)** Features (non-negative)
- **X** Evidence
Discriminative SPNs

\[ f(X) \]

\[ f_1(X) \]

\[ f_2(X) \]

\[ X \] Evidence

\[ Y \] Query

\[ H \] Hidden

Features (non-negative)

Y1 Y1 Y2 Y2

0.8 0.1 1.5
Discriminative SPNs

Greater variety than generative SPNs

\[ f(X) \] Features (non-negative)

\[ Y \] Query

\[ X \] Evidence

\[ f_1(X) \]

\[ f_2(X) \]
Discriminative Training

\[ \nabla \log P(y|x) = \nabla \log \frac{P(y, x)}{P(x)} = \]

\[ \nabla \log \sum_h P(y, h, x) - \nabla \log \sum_{y', h} P(y', h, x) \]

Correct label

Best guess

Tractable!
SPN Backpropagation
SPN Backpropagation

\[ f_1(X) \]

\[ f_2(X) \]

Y_1

Y_1

Y_2

Y_2

0.8

0.3

0.4

0.9

0.6

0.3

0.7

0.4

0.2

0.8

0.9

0.1

0.5

0.5

0.6
SPN Backpropagation

Diagram showing a neural network with nodes labeled with 0.3 and 0.7, and inputs labeled with 0.18, 0.24, and 0.16.
SPN Backpropagation

Diagram of a neural network with nodes and connections.
SPN Backpropagation

For each child $j$:
\[
\frac{\partial S}{\partial S_j} \leftarrow \frac{\partial S}{\partial S_j} + w_{n,j} \frac{\partial S}{\partial S_n}
\]
\[
\frac{\partial S}{\partial w_{n,j}} \leftarrow S_j \frac{\partial S}{\partial S_n}
\]
SPN Backpropagation

For each child $j$:

$$\frac{\partial S}{\partial S_j} \leftarrow \frac{\partial S}{\partial S_j} + \frac{\partial S}{\partial S_n} \prod_{k \in \text{Ch}(n) \setminus \{j\}} S_k$$
SPN Backpropagation

For each child $j$:

$$\frac{\partial S}{\partial S_j} \leftarrow \frac{\partial S}{\partial S_j} + \frac{\partial S}{\partial S_n} \prod_{k \in Ch(n) \setminus \{j\}} S_k$$
Problem with Backpropagation

Gradient diffusion
Hard Inference Overcomes Gradient Diffusion

Soft Inference (Marginals)

Hard Inference (MAP States)
Reasons to Use Hard Inference

- To overcome gradient diffusion
- When goal is to predict most probable structure
- For speed or tractability
Hard Gradient

$$\nabla \log \tilde{P}(y|x) = \nabla \log \frac{\tilde{P}(y, x)}{\tilde{P}(x)} =$$

$$\nabla \log \max_h P(y, h, x) \quad \text{Correct label} \quad \nabla \log \max_{y', h} P(y', h, x) \quad \text{Best guess}$$
Hard Gradient

$$\nabla \log \tilde{P}(y|x) = \nabla \log \frac{\tilde{P}(y, x)}{\tilde{P}(x)} =$$

$$\nabla \log \max_h P(y, h, x) - \nabla \log \max_{y', h} P(y', h, x)$$
Hard Gradient

\[ \nabla \log \tilde{P}(y|x) = \nabla \log \frac{\tilde{P}(y, x)}{\tilde{P}(x)} = \]

\[ \nabla \log \left( \max_{h} P(y, h, x) \right) - \nabla \log \left( \max_{y', h} P(y', h, x) \right) \]
Hard Gradient

$$\nabla \log \tilde{P}(y|x) = \nabla \log \frac{\tilde{P}(y, x)}{\tilde{P}(x)} =$$

$$\nabla \log \max_h P(y, h, x) - \nabla \log \max_{y', h} P(y', h, x)$$
Hard Gradient

\[ \nabla \log \tilde{P}(y|x) = \nabla \log \left( \frac{\tilde{P}(y, x)}{\tilde{P}(x)} \right) = \]

\[ \nabla \log \left( \max_{y'} P(y', h, x) \right) - \nabla \log \left( \max_{y} P(y, h, x) \right) \]
Hard Gradient

$$\nabla \log \tilde{P}(y|x) = \nabla \log \frac{\tilde{P}(y, x)}{\tilde{P}(x)} =$$

$$\nabla \log f(X)$$
Hard Gradient

\[
\nabla \log \tilde{P}(y|x) = \nabla \log \frac{\tilde{P}(y, x)}{\tilde{P}(x)} = \\

\nabla \log \begin{pmatrix}
\max & \max & \max & \max \\
\times & \times & \times & \times \\
\end{pmatrix}

\frac{\partial}{\partial w_i} \log \tilde{P}(y|x) = \frac{\Delta c_i}{w_i}

\]
# Learning SPNs: Summary

<table>
<thead>
<tr>
<th>Update</th>
<th>Soft Inference (Marginals)</th>
<th>Hard Inference (MAP States)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen. EM</td>
<td>$\Delta w_i \propto w_i \frac{\partial S}{\partial S_k}$</td>
<td>$\Delta w_i = c_i$</td>
</tr>
<tr>
<td>Gen. Gradient</td>
<td>$\Delta w_i = \eta \frac{\partial S}{\partial S_k} S_i$</td>
<td>$\Delta w_i = \eta \frac{c_i}{w_i}$</td>
</tr>
<tr>
<td>Disc. Gradient</td>
<td>$\Delta w_i = \eta \left( \frac{S_i \partial S}{S \partial S_k} - \frac{S_i \partial S}{S \partial S_k} \right)$</td>
<td>$\Delta w_i = \frac{\eta}{w_i} \left( \sum_{\text{true}} - \sum_{\text{test}} \right)$</td>
</tr>
</tbody>
</table>

\[ \Delta w_i = \frac{\eta}{w_i} (c_i - c_i) \]
## Learning SPNs: Summary

<table>
<thead>
<tr>
<th>Update</th>
<th>Soft Inference (Marginals)</th>
<th>Hard Inference (MAP States)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen. EM</td>
<td>$\Delta w_i \propto w_i \frac{\partial S}{\partial S_k}$</td>
<td>$\Delta w_i = c_i$</td>
</tr>
<tr>
<td>Gen. Gradient</td>
<td>$\Delta w_i = \eta \frac{\partial S}{\partial S_k} S_i$</td>
<td>$\Delta w_i = \eta \frac{c_i}{w_i}$</td>
</tr>
<tr>
<td>Disc. Gradient</td>
<td>$\Delta w_i = \eta \left( \frac{S_i}{S} \frac{\partial S}{\partial S_k} - \frac{S_i}{S} \frac{\partial S}{\partial S_k} \right)$</td>
<td>$\Delta w_i = \eta \frac{\text{true test}}{w_i} \left( c_i - c_i \right)$</td>
</tr>
</tbody>
</table>
## Learning SPNs: Summary

<table>
<thead>
<tr>
<th>Update</th>
<th>Soft Inference (Marginals)</th>
<th>Hard Inference (MAP States)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen. EM</td>
<td>$\Delta w_i \propto w_i \frac{\partial S}{\partial S_k}$</td>
<td>$\Delta w_i = C_i$</td>
</tr>
<tr>
<td>Gen. Gradient</td>
<td>$\Delta w_i = \eta \frac{\partial S}{\partial S_k} S_i$</td>
<td>$\Delta w_i = \eta \frac{\partial S}{w_i}$</td>
</tr>
<tr>
<td>Disc. Gradient</td>
<td>$\Delta w_i = \eta \left( \frac{S_i}{S} \frac{\partial S}{\partial S_k} - \frac{S_i}{S} \frac{\partial S}{\partial S_k} \right)$</td>
<td>$\Delta w_i = \eta \frac{\text{true label}}{w_i} \left( C_i - C_i \right)$</td>
</tr>
</tbody>
</table>
Motivation

SPN Review

Discriminative Training

Experiments
Image Classification

CIFAR-10
32x32px
50k train
10k test

STL-10
96x96px
5k train
8k test
100k unlabeled

10 folds
Feature Extraction

Coates et al., AISTATS 2011

K-means

Triangle encoding

Max-pooling
SPN Architecture

\[ e^{\tilde{x}_{ij} \cdot f_{cpt}} \]

\[ \text{WxWxK} \]

\[ \text{GxGxK} \]
CIFAR-10 Results

Accuracy vs Dictionary Size (K)

- SPN
- Pooling
- SVM
- Autoenc.
- RBM

Accuracy:
- 84%
- 80%
- 76%
- 72%
- 68%
- 64%

Dictionary Size (K):
- 200
- 400
- 800
- 1600
- 4000

4x4xK
CIFAR-10 Results

SPN models spatial structure among features

20x fewer than SVM
CIFAR-10 Results

7x7x400

Accuracy

0k 38k 75k 113k 150k

#Features

Best published

SPN

84%

83%

82%

81%

80%

79%

SVM

Learned Pooling

3-Layer Learned RF
STL-10 results

- 1-layer Vector Quantization: 54.9%
- 1-layer Sparse Coding: 59.0%
- 3-layer Learned Receptive Field: 60.1%
- Discriminative SPN: 62.3%

(without unlabeled data)
Future Work

• Max-margin SPNs
• Learning SPN structure
• Applying discriminative SPNs to structured prediction
• Approximate inference using SPNs
Summary

• Discriminative SPNs combine the advantages of
  • Tractable inference
  • Deep architectures
  • Discriminative learning

• Hard gradient combats diffusion in deep models

• Discriminative SPNs outperform SVMs and deep models on image classification benchmarks