Discriminative Learning of Sum-Product Networks

Robert Gens
Pedro Domingos
Features
Mixtures
Parts
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.
A Univariate Distribution
Is an SPN.

X

Multinomial
Gaussian
Poisson

...
A Univariate Distribution Is an SPN.
A Product of SPNs over Disjoint Variables Is an SPN.
A Product of SPNs over Disjoint Variables Is an SPN.
A Weighted Sum of SPNs over the Same Variables is an SPN.
A Weighted Sum of SPNs over the Same Variables Is an SPN.

Sums out a mixture variable

\[ w_1 \times X \] \[ w_2 \times Y \]
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
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

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

$P(X=0)$?
All Marginals Are Computable in Linear Time

\[ P(X=0) = 0.26 \]
All MAP States Are Computable in Linear Time
All MAP States Are Computable in Linear Time
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

\[
\max_y P(X=0, Y=y) = 0.12
\]
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
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 Representable Probability Distributions
Compactly Representable Probability Distributions

Graphical Models
Compactly Representable Probability Distributions

Graphical Models

Sum-Product Networks
Compactly Representable Probability Distributions

Graphical Models

Existing Tractable Models

Sum-Product Networks
# 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

\[ P(Y|X) \]
Discriminative SPNs

\[ P(Y|X) \]

Y  Query
Discriminative SPNs

\[ P(Y|X) \]

Y  Query

H  Hidden
Discriminative SPNs

\[ P(Y|X) \]

Y  Query

H  Hidden

X  Evidence
Discriminative SPNs

\[ P(Y|X) \]

Y  Query
H  Hidden
X  Evidence

Treat as constants
Discriminative SPNs

\[ Y \quad \text{Query} \]

\[ H \quad \text{Hidden} \]

\[ X \quad \text{Evidence} \]
Discriminative SPNs

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

\[ H \quad \text{Hidden} \]

\[ Y \quad \text{Query} \]

\[ f(X) \quad \text{Features} \quad \text{(non-negative)} \]

\[ X \quad \text{Evidence} \]
Discriminative SPNs

$H$ Hidden

$Y$ Query

$f(X)$ Features (non-negative)

$X$ Evidence

$Y_1, Y_1, Y_2, Y_2$

$0.4, 0.1, 0.5$

$f_1(X), f_2(X)$
Discriminative SPNs

\[ f(X) \]

Features (non-negative)

Evidence

Query

Hidden

\[ H \]

\[ f_1(X) \]

\[ f_2(X) \]
Discriminative SPNs

Greater variety than generative SPNs

$f_1(X)$

$f_2(X)$

$Y_1$

$Y_1$

$Y_2$

$Y_2$

0.4

0.1

0.5

$X$

Query

Features (non-negative)

Evidence
Discriminative Training

\[ \nabla \log P(y|x) = \]
Discriminative Training

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

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

The diagram illustrates the SPN backpropagation process with nodes labeled $f_1(X)$ and $f_2(X)$, and intermediate nodes labeled $Y_1$ and $Y_2$. The numbers represent the values propagated through the network, with the final output being a combination of these values.
SPN Backpropagation
SPN Backpropagation

The diagram represents a SPN (Sum-Product Network) model with two functions $f_1(X)$ and $f_2(X)$, each composed of nodes that perform addition (+) and multiplication (×). The values and nodes are as follows:

- $f_1(X)$:
  - $f_1(0.9, 0.6) = 0.45$, with intermediate values 0.9 and 0.6 leading to 0.45.
  - The output of $f_1$ is further combined with $f_2$ through addition.

- $f_2(X)$:
  - $f_2(0.4, 0.6) = 0.65$, with intermediate values 0.4 and 0.6 leading to 0.65.
  - The output of $f_2$ is also combined with $f_1$ through addition.

The combined output is then further processed through more additions and multiplications, resulting in the final output values 0.18 and 0.16, which are shown at the top of the diagram.
SPN Backpropagation
SPN Backpropagation
SPN Backpropagation

1

0.3 + 0.7

0.18

0.24

0.16
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 $j$:
\[
\frac{\partial S}{\partial S_j} \leftarrow \frac{\partial S}{\partial S_j} + w_{n,j} \frac{\partial S}{\partial S_n}
\]
\[
\frac{\partial S}{\partial w_{n,j}} \leftarrow s_j \frac{\partial S}{\partial S_n}
\]
SPN Backpropagation
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 $j$:

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

For each child $j$:

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

$f_1(X)$
SPN Backpropagation

For each child $j$:

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

Gradient diffusion
Hard Inference Overcomes Gradient Diffusion

Soft Inference
(Marginals)

Hard Inference
(MAP States)
Reasons to Use Hard Inference
Reasons to Use Hard Inference

• To overcome gradient diffusion
Reasons to Use Hard Inference

- To overcome gradient diffusion
- When goal is to predict most probable structure
Reasons to Use Hard Inference

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

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

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

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

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

Correct label
Hard Gradient

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

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

Correct label

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

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

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

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

$$\nabla \log \max \ P(y, h, x)$$

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

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

$$\nabla \log f_1(X) = \max_y P(y, h, x)$$

$$\max_{y', h} P(y', h, x)$$
Hard Gradient

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

$$\nabla \log f_1(X)$$

Diagram:

- $Y_1$, $Y_1$, $Y_2$, $Y_2$ nodes with max operations.
- $f_1(X)$ node with a cross.
- Relates to the log derivative of the probability distribution.
Hard Gradient

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

\[ \nabla \log \begin{pmatrix} f_1(X) \times \max \atop \max \atop \max \atop \max \atop \max \end{pmatrix} = \]

\[ \begin{pmatrix} Y_1 \times \max \atop \max \atop \max \atop \max \atop \max \end{pmatrix} \]
Hard Gradient

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

\[ \nabla \log \left( \begin{array}{c}
\max \ f_1(X) \\
\times \\
\max \ f_2(X) \\
\end{array} \right) \]

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

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

\[ \nabla \log \begin{pmatrix} f_1(X) \\ \text{max} \\ f_2(X) \end{pmatrix} = \begin{pmatrix} Y_1 \\ \text{max} \\ Y_2 \\ \text{max} \end{pmatrix} - \begin{pmatrix} Y_1 \\ \text{max} \\ Y_2 \end{pmatrix} \]

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

# w/ correct label - # w/ model guess
## 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( \overbrace{\frac{S_i}{S} \frac{\partial S}{\partial S_k}}^{\text{true label}} - \overbrace{\frac{S_i}{S} \frac{\partial S}{\partial S_k}}^{\text{exp. label}} \right) )</td>
<td>( \Delta w_i = \eta \left( \frac{c_i}{w_i} \right) \left( \frac{\text{true}}{\text{test}} \right) \left( c_i - c_i \right) )</td>
</tr>
</tbody>
</table>

**Update**

- **Gen. EM**
- **Gen. Gradient**
- **Disc. Gradient**

**Soft Inference (Marginals)**

- \( \Delta w_i \propto w_i \frac{\partial S}{\partial S_k} \)
- \( \Delta w_i = \eta \frac{\partial S}{\partial S_k} S_i \)

**Hard Inference (MAP States)**

- \( \Delta w_i = c_i \)
- \( \Delta w_i = \eta \frac{c_i}{w_i} \)
- \( \Delta w_i = \eta \left( \frac{c_i}{w_i} \right) \left( \frac{\text{true}}{\text{test}} \right) \left( c_i - c_i \right) \)

**Diagram**

- Node labeled \( S_k \) with two outgoing edges: one to \( w_i \) and another to a node labeled \( S_i \) with a self-loop.
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>$\Delta w_i = \eta \left( \frac{c_{\text{true}}}{w_i} - \frac{c_{\text{test}}}{w_i} \right)$</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>$\Delta w_i = \eta \frac{C_i}{w_i}$</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>$\Delta w_i = \eta \frac{\text{true}}{\text{test}} (C_i - C_i)$</td>
</tr>
</tbody>
</table>

**Update**
- Gen. EM
- Gen. Gradient
- Disc. Gradient

**Soft Inference (Marginals)**
- $\Delta w_i \propto w_i \frac{\partial S}{\partial S_k}$
- $\Delta w_i = \eta \frac{\partial S}{\partial S_k} S_i$
- $\Delta w_i = \eta \left( \frac{S_i \partial S}{S \partial S_k} - \frac{S_i \partial S}{S \partial S_k} \right)$

**Hard Inference (MAP States)**
- $\Delta w_i = C_i$
- $\Delta w_i = \eta \frac{C_i}{w_i}$
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
Feature Extraction

Coates et al., AISTATS 2011

K-means 6x6
Feature Extraction
Coates et al., AISTATS 2011

32x32

K-means

6x6
Feature Extraction
Coates et al., AISTATS 2011

32x32

K-means

6x6

Triangle encoding

K
Feature Extraction

Coates et al., AISTATS 2011
Feature Extraction
Coates et al., AISTATS 2011

32x32 → 6x6
K-means → Triangle encoding

27x27xK → Max-pooling

GxGxK

Coates et al., AISTATS 2011
Feature Extraction
Coates et al., AISTATS 2011

32x32 → 27x27xK → GxGxK
SPN Architecture

GxGxK
SPN Architecture

Classes \( \oplus \)

G\(x\)G\(x\)K
SPN Architecture

Classes +

Parts ×

GxGxK
SPN Architecture

GxGxK

Classes +

Parts ×

Mixture +

GxGxK
SPN Architecture

Location $\times \frac{e^{x_{ij}} \cdot \bar{f}_{cpt}}{f_{cpt}}$ ~ $x_{ij} \cdot f_{cpt}$

Parts $\times$

Mixture $+$

Classes $+$

WxWxK

GxGxK
CIFAR-10 Results
CIFAR-10 Results

Accuracy

84%
80%
76%
72%
68%
64%

Dictionary Size (K)

4x4xK
CIFAR-10 Results

Accuracy

Dictionary Size (K)

SVM

Autoenc.

RBM

4x4xK

64% 68% 72% 76% 80% 84%
CIFAR-10 Results

Accuracy vs Dictionary Size (K)

- Pooling
- SVM
- Autoenc.
- RBM

4x4xK

Accuracy: 64% to 84%

Dictionary Size (K): 200 to 4000
CIFAR-10 Results

Accuracy vs. Dictionary Size (K)

- SPN
- Pooling
- SVM
- Autoenc.
- RBM

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

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

4x4xK
CIFAR-10 Results

Accuracy

Dictionary Size (K)

SVM

Pooling

SPN
CIFAR-10 Results

Accuracy vs. Dictionary Size (K) for SVM, SPN, and Pooling.
CIFAR-10 Results

20x fewer than SVM

Accuracy

Dictionary Size (K)
CIFAR-10 Results

20x fewer than SVM

SPN models spatial structure among features

SPN

Pooling

SVM
CIFAR-10 Results

7x7x400

- SPN: 84%
- Learned Pooling: 83%
- 3-Layer Learned RF: 82%
- SVM: 79%

#Features

0k, 38k, 75k, 113k, 150k
CIFAR-10 Results

7x7x400

Accuracy

#Features

- SPN: 84%
- SVM: 79%
- Best published
- Learned Pooling: 82%
- 3-Layer Learned RF: 83%
STL-10 results

Without unlabeled data

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

• Max-margin SPNs
Future Work

• Max-margin SPNs
• Learning SPN structure
Future Work

• Max-margin SPNs
• Learning SPN structure
• Applying discriminative SPNs to structured prediction
Future Work

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

• Discriminative SPNs combine the advantages of
Summary

• Discriminative SPNs combine the advantages of
  • Tractable inference
Summary

- Discriminative SPNs combine the advantages of
  - Tractable inference
  - Deep architectures
Summary

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

- Discriminative SPNs combine the advantages of
  - Tractable inference
  - Deep architectures
  - Discriminative learning
- Hard gradient combats diffusion in deep models
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