Tractable Computation of Expected Kernels by Circuits

Wenzhe Li*
Tsinghua University

Zhe Zeng*
University of California, Los Angeles

Antonio Vergari
University of California, Los Angeles

Guy Van den Broeck
University of California, Los Angeles

April 29th, 2021 - Yahoo Research Seminar
Tractable Computation of Expected Kernels by Circuits

Wenzhe Li*
Tsinghua University

Zhe Zeng*
University of California, Los Angeles

Antonio Vergari
University of California, Los Angeles

Guy Van den Broeck
University of California, Los Angeles

April 29th, 2021 - Yahoo Research Seminar
Tractable Computation of Expected Kernels by Circuits

Wenzhe Li*
Tsinghua University

Zhe Zeng*
University of California, Los Angeles

Antonio Vergari
University of California, Los Angeles

Guy Van den Broeck
University of California, Los Angeles

April 29th, 2021 - Yahoo Research Seminar
Tractable Computation of Expected Kernels by Circuits

Wenzhe Li*
Tsinghua University

Zhe Zeng*
University of California, Los Angeles

Antonio Vergari
University of California, Los Angeles

Guy Van den Broeck
University of California, Los Angeles

April 29th, 2021 - Yahoo Research Seminar
Motivation

A Fundamental Task

Given two distributions $p$ and $q$, and a kernel $k$, the task is to compute the expected kernel

$$\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')]$$
Motivation

A Fundamental Task

Given two distributions $p$ and $q$, and a kernel $k$, the task is to compute the expected kernel

$$\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')]$$

⇒ In kernel-based frameworks, expected kernels are omnipresent!
Motivation

A Fundamental Task

Given two distributions $p$ and $q$, and a kernel $k$, the task is to compute the expected kernel

$$\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')]$$

⇒ In kernel-based frameworks, expected kernels are omnipresent!

Squared Maximum Mean Discrepancy (MMD)

$$\mathbb{E}_{x \sim p, x' \sim p}[k(x, x')] + \mathbb{E}_{x \sim q, x' \sim q}[k(x, x')] - 2\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')]$$
Motivation

A Fundamental Task

Given two distributions $p$ and $q$, and a kernel $k$, the task is to compute the expected kernel

$$\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')]$$

$\Rightarrow$ In kernel-based frameworks, expected kernels are omnipresent!

Discrete Kernelized Stein Discrepancy (KDSD)

$$\mathbb{E}_{x, x' \sim q}[k_p(x, x')]$$
\[ E_{x \sim p, x' \sim q}[k(x, x')] = \int_{x, x'} p(x) q(x') k(x, x') \, dx \, dx' \]
**Challenge**

Reliability vs. Flexibility

\[
\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')] = \int_{x, x'} p(x)q(x')k(x, x') \, dx \, dx'
\]

\(p, q, k\) fully factorized

\(p(x) = \prod_i p(x_i), \, q(x) = \prod_i q(x_i)\)

\(k(x, x') = \prod_i k(x_i, x'_i)\)

⇒ expected kernel is **tractable**

\(\prod_i (\int_{x_i, x'_i} p(x_i)q(x'_i)k(x_i, x'_i))\)
**Challenge**

*Reliability vs. Flexibility*

\[
\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')] = \int_{x, x'} p(x)q(x')k(x, x') \, dx \, dx'
\]

\[p, q, k \text{ fully factorized}
\]

\[
p(x) = \prod_i p(x_i), q(x) = \prod_i q(x_i)
\]

\[
k(x, x') = \prod_i k(x_i, x_i)
\]

\[\Rightarrow \text{ expected kernel is tractable}
\]

\[\prod_i (\int_{x_i, x_i'} p(x_i)q(x_i')k(x_i, x_i'))
\]

A computation is **tractable** if it can be done exactly in polynomial time.
**Challenge**

Reliability vs. Flexibility

\[
\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')] = \int_{x, x'} p(x)q(x')k(x, x') \, dx \, dx'
\]

\(p, q, k\) fully factorized

**PRO.** Tractable exact computation

**CON.** Model being too restrictive
**Challenge**

Reliability vs. Flexibility

\[
\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')] = \int_{x, x'} p(x) q(x') k(x, x') \, dx \, dx'
\]

\( p, q, k \) fully factorized

**PRO.** Tractable exact computation

**CON.** Model being too restrictive

Hard to compute in general. \( \Rightarrow \) approximate with MC or variational inference

**PRO.** Efficient computation

**CON.** no guarantees on error bounds
Challenge
Reliability vs. Flexibility

\[ \mathbb{E}_{x \sim p, x' \sim q}[k(x, x')] = \int_{x, x'} p(x)q(x')k(x, x') \, dx \, dx' \]

\( p, q, k \) fully factorized

**PRO.** Tractable exact computation

**CON.** Model being too restrictive

**trade-off?**

Hard to compute in general.

\[ \Rightarrow \text{ approximate with MC or variational inference} \]

**PRO.** Efficient computation

**CON.** no guarantees on error bounds
Expressive distribution models

+ 

Exact computation of expected kernels?
Expressive distribution models

+ 

Exact computation of expected kernels

= 

Circuits!
Probabilistic Circuits

deep generative models + deep guarantees
Circuits

Probabilistic Circuits

deep generative models + deep guarantees

Kernel Circuits

express kernels as circuits
Circuits

Probabilistic Circuits
deep generative models + deep guarantees

Kernel Circuits
express kernels as circuits

⇒ \[ \mathbb{E}_{x \sim p, x' \sim q}[k(x, x')] \]
1. A simple tractable distribution is a PC
\[ \implies \text{e.g., a multivariate Gaussian} \]
Probabilistic Circuits (PCs)

Tractable computational graphs

I. A simple tractable distribution is a PC
II. A convex combination of PCs is a PC

⇒ e.g., a mixture model
Probabilistic Circuits (PCs)

Tractable computational graphs

I. A simple tractable distribution is a PC
II. A convex combination of PCs is a PC
III. A product of PCs is a PC
Probabilistic Circuits (PCs)

Tractable computational graphs
Probabilistic Circuits (PCs)

Tractable computational graphs
Probabilistic queries = feedforward evaluation

\[ p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2) \]
Probabilistic queries = feedforward evaluation

\[ p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2) \]
Probabilistic queries = feedforward evaluation

\[ p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2) = 0.75 \]
PCs = deep learning

PCs are computational graphs
PCs = deep learning

PCs are computational graphs encoding deep mixture models

⇒ stacking (categorical) latent variables
PCs = **deep learning**

PCs are computational graphs encoding **deep mixture models**

⇒ stacking (categorical) latent variables

PCs compactly represent **polynomials with exponentially many terms**

⇒ universal approximators
PCs = \textit{deep learning}

PCs are computational graphs encoding \textit{deep mixture models} \Rightarrow \text{stacking (categorical) latent variables}

PCs compactly represent \textit{polynomials with exponentially many terms} \Rightarrow \text{universal approximators}

PCs are expressive \textit{deep generative models}!
⇒ we can learn PCs with millions of parameters in minutes on the GPU \cite{Peharz et al. 2020}
On par with intractable models!

How expressive are PCs?

<table>
<thead>
<tr>
<th>dataset</th>
<th>best circuit</th>
<th>BN</th>
<th>MADE</th>
<th>VAE</th>
<th>dataset</th>
<th>best circuit</th>
<th>BN</th>
<th>MADE</th>
<th>VAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nltcs</td>
<td>-5.99</td>
<td>-6.02</td>
<td>-6.04</td>
<td>-5.99</td>
<td>dna</td>
<td>-79.88</td>
<td>-80.65</td>
<td>-82.77</td>
<td>-94.56</td>
</tr>
<tr>
<td>msnbc</td>
<td>-6.04</td>
<td>-6.04</td>
<td>-6.06</td>
<td>-6.09</td>
<td>kosarek</td>
<td>-10.52</td>
<td>-10.83</td>
<td>-</td>
<td>-10.64</td>
</tr>
<tr>
<td>plants</td>
<td>-11.84</td>
<td>-12.65</td>
<td>-12.32</td>
<td>-12.34</td>
<td>book</td>
<td>-33.82</td>
<td>-36.41</td>
<td>-33.95</td>
<td>-33.19</td>
</tr>
<tr>
<td>audio</td>
<td>-39.39</td>
<td>-40.50</td>
<td>-38.95</td>
<td>-38.67</td>
<td>movie</td>
<td>-50.34</td>
<td>-54.37</td>
<td>-48.7</td>
<td>-47.43</td>
</tr>
<tr>
<td>jester</td>
<td>-51.29</td>
<td>-51.07</td>
<td>-52.23</td>
<td>-51.54</td>
<td>webkb</td>
<td>-149.20</td>
<td>-157.43</td>
<td>-149.59</td>
<td>-146.9</td>
</tr>
<tr>
<td>netflix</td>
<td>-55.71</td>
<td>-57.02</td>
<td>-55.16</td>
<td>-54.73</td>
<td>cr52</td>
<td>-81.87</td>
<td>-87.56</td>
<td>-82.80</td>
<td>-81.33</td>
</tr>
<tr>
<td>accidents</td>
<td>-26.89</td>
<td>-26.32</td>
<td>-26.42</td>
<td>-29.11</td>
<td>c20ng</td>
<td>-151.02</td>
<td>-158.95</td>
<td>-153.18</td>
<td>-146.9</td>
</tr>
<tr>
<td>retail</td>
<td>-10.72</td>
<td>-10.87</td>
<td>-10.81</td>
<td>-10.83</td>
<td>bbc</td>
<td>-229.21</td>
<td>-257.86</td>
<td>-242.40</td>
<td>-240.94</td>
</tr>
</tbody>
</table>

Unifying existing tractable models

Chow-Liu trees
[Chow and Liu 1968]

Junction trees
[Bach and Jordan 2001]

HMMs
[Rabiner and Juang 1986]

Classical tractable models can be compactly represented as PCs

Chow-Liu trees
[Chow and Liu 1968]

Junction trees
[Bach and Jordan 2001]

HMMs
[Rabiner and Juang 1986]

Cnets
[Rahman et al. 2014]

SPNs
[Poon et al. 2011]

PSDDs
[Kisa et al. 2014]

PDGs
[Jaeger 2004]
PCs = \textcolor{teal}{deep learning} + \textcolor{purple}{deep guarantees} \\

PCs are expressive \textit{deep generative models}! \\

\& \textcolor{orange}{Certifying tractability} for a class of queries \\

= \textcolor{green}{verifying structural properties} of the graph
Which structural constraints ensure tractability?
A PC is *decomposable* if all inputs of product units depend on disjoint sets of variables.
A PC is **decomposable** if all inputs of product units depend on disjoint sets of variables.

A PC is **smooth** if all inputs of sum units depend of the same variable sets.

---

**Darwiche and Marquis**, “A knowledge compilation map”, 2002
decomposable + smooth PCs = ...

decomposable + smooth PCs = ...

MAR sufficient and necessary conditions for computing any marginal

\[ p(y) = \int_{\text{val}(Z)} p(z, y) \, dZ, \quad \forall Y \subseteq X, \quad Z = X \setminus Y \]

\[ \Rightarrow \quad \text{by a single feedforward evaluation} \]

---

**decomposable** + **smooth** PCs = ...

**MAR** \( \text{sufficient and necessary conditions for computing any marginal } \int p(z, y) \, dZ \)

**CON** \( \text{sufficient and necessary conditions for any conditional distribution} \)

\[
p(y \mid z) = \frac{\int_{\text{val}(H)} p(z, y, h) \, dH}{\int_{\text{val}(H)} \int_{\text{val}(Y)} p(z, y, h) \, dH \, dY}, \quad \forall Z, Y \subseteq X
\]

\[ \Rightarrow \text{ by two feedforward evaluations} \]

---

*Choi et al., “Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Modeling”, 2020*
decomposable + smooth PCs = ...

MAR  sufficient and necessary conditions for computing any marginal $\int p(z, y) \, dZ$

CON  sufficient and necessary conditions for any conditional $\frac{\int \int p(z, y, h) \, dH}{\int \int p(z, y) \, dH \, dY}$

?  What about the expected kernel $\mathbb{E}_{x \sim p, x' \sim q}[k(x, x')]$?

Can we represent kernels as circuits to characterize tractability of its queries?
**Kernel Circuits (KCs)**

**Exa.** Radial basis function (RBF) kernel $k(x, x') = \exp\left(-\sum_{i=1}^{4} |X_i - X'_i|^2\right)$

\[
\begin{align*}
\exp(-|X_1 - X'_1|^2) & \land \\
\exp(-|X_2 - X'_2|^2) & \land \\
\exp(-|X_3 - X'_3|^2) & \land \\
\exp(-|X_4 - X'_4|^2) & \land \\
\end{align*}
\]
**Kernel Circuits (KCs)**

**Exa.** Radial basis function (RBF) kernel $k(x, x') = \exp\left(-\sum_{i=1}^{4} |X_i - X_i'|^2\right)$

\[
\exp(-|X_1 - X_1'|^2) \land \\
\exp(-|X_2 - X_2'|^2) \land \\
\exp(-|X_3 - X_3'|^2) \land \\
\exp(-|X_4 - X_4'|^2)
\]

*decomposable* if all inputs of product units depend on disjoint sets of variables
**Kernel Circuits (KCs)**

**Exa.** Radial basis function (RBF) kernel \( k(x, x') = \exp \left( - \sum_{i=1}^{4} |X_i - X'_i|^2 \right) \)

\[
\exp(-|X_1 - X'_1|^2) \land \\
\exp(-|X_2 - X'_2|^2) \land \\
\exp(-|X_3 - X'_3|^2) \land \\
\exp(-|X_4 - X'_4|^2)
\]

**decomposable** if all inputs of product units depend on disjoint sets of variables

**smooth** if all inputs of sum units depend of the same variable sets
Kernel Circuits (KCs)

Common kernels can be compactly represented as **decomposable** + **smooth** KCs:

- RBF, (exponentiated) Hamming, polynomial ...
Expected Kernel

tractable computation via circuit operations

i) PCs $p$ and $q$, and KC $k$ are decomposable + smooth
Expected Kernel

tractable computation via circuit operations

i) PCs $p$ and $q$, and KC $k$ are \textbf{decomposable} + smooth

ii) PCs $p$ and $q$, and KC $k$ are \textbf{compatible}

$\Rightarrow$ decompose in the same way
**Expected Kernel**

*tractable computation via circuit operations*

i) PCs $p$ and $q$, and KC $k$ are **decomposable + smooth**

ii) PCs $p$ and $q$, and KC $k$ are **compatible**
Expected Kernel

tractable computation via circuit operations

i) PCs $p$ and $q$, and KC $k$ are decomposable + smooth

ii) PCs $p$ and $q$, and KC $k$ are compatible
**Expected Kernel**

*tractable computation via circuit operations*

i) PCs $p$ and $q$, and KC $k$ are **decomposable** + **smooth**

ii) PCs $p$ and $q$, and KC $k$ are **compatible**
Expected Kernel

tractable computation via circuit operations

i) PCs $p$ and $q$, and KC $k$ are decomposable + smooth

ii) PCs $p$ and $q$, and KC $k$ are compatible
Expected Kernel

tractable computation via circuit operations

i) PCs $p$ and $q$, and KC $k$ are decomposable + smooth

ii) PCs $p$ and $q$, and KC $k$ are compatible

Then computing expected kernels can be done tractably by a forward pass $\Rightarrow$ product of the sizes of each circuit!
smooth + decomposable + compatible = tractable $E[k]$.

**[Sum Nodes]** $p(X) = \sum_i w_i p_i(X)$, $q(X') = \sum_j w'_j q_j(X')$, and kernel $k(X, X') = \sum_l w''_l k_l(X, X')$:

\[
p(X) = \sum_i w_i p_i(X), \quad q(X') = \sum_j w'_j q_j(X'), \quad k(X, X') = \sum_l w''_l k_l(X, X').
\]
**smooth** + **decomposable** + **compatible** = **tractable** $E[k]$

[Sum Nodes] $p(X) = \sum_i w_i p_i(X)$, $q(X') = \sum_j w'_j q_j(X')$, and kernel $k(X, X') = \sum_l w''_l k_l(X, X')$:

$$\sum_{x, x'} p(x) q(x') k(x, x') = \sum_{i, j, l} w_i w'_j w''_l p_i(x) q_j(x) k_l(x, x')$$
smooth + decomposable + compatible = tractable $E[k]$

[Sum Nodes] $p(X) = \sum_i w_i p_i(X)$, $q(X') = \sum_j w'_j q_j(X')$, and kernel $k(X, X') = \sum_l w''_l k_l(X, X')$:

$\mathbb{E}_{p, q}[k(X, X')] = \sum_{i, j, l} w_i w'_j w''_l \mathbb{E}_{p_i, q_j}[k_l(X, X')]$

$\implies$ expectation is “pushed down” to children
**smooth** + **decomposable** + **compatible** = **tractable** $E[k]$

**[Product Nodes]** $p_x(X) = \prod_i p_i(X_i)$, $q_x(X') = \prod_i q_j(X'_i)$, and kernel $k_x(X, X') = \prod_i k_i(X_i, X'_i)$:

![Diagram of product nodes]
\[ \text{smooth} + \text{decomposable} + \text{compatible} = \text{tractable } E[k] \]

**[Product Nodes]** \( p_x(x) = \prod_i p_i(x_i) \), \( q_x(x') = \prod_i q_i(x'_i) \), and kernel \( k_x(x, x') = \prod_i k_i(x_i, x'_i) \):

\[
\begin{align*}
\sum_{x, x'} p_x(x) q_x(x') k_x(x, x') &= \sum_{x, x'} \prod_i p(x_i) q(x_i) k_i(x_i, x'_i) \\
&= \prod_i \left( \sum_{x_i, x'_i} p(x_i) q(x_i) k_i(x_i, x'_i) \right)
\end{align*}
\]
\textbf{smooth} + \textbf{decomposable} + \textbf{compatible} = \textbf{tractable} \ E[k]

\textbf{[Product Nodes]} \ p_x(X) = \prod_i p_i(x_i), \ q_x(X') = \prod_i q_i(x_i'), \ \text{and kernel} \ k_x(x, x') = \prod_i k_i(x_i, x_i'):

\[ \prod_i p_i(x_i) \times \prod_i q_i(x_i') \times \prod_i k_i(x_i, x_i') \]

\[
\mathbb{E}_{p_x, q_x}[k_x(x, x')] = \prod_i \mathbb{E}_{p, q}[k(x_i, x_i')]
\]

\[
\implies \text{expectation decomposes into easier ones}
\]
**smooth** + **decomposable** + **compatible** = **tractable** $E[k]$

**Algorithm 1** $E_{p_n,q_m}[k_l]$ — Computing the expected kernel

**Input:** Two compatible PCs $p_n$ and $q_m$, and a KC $k_l$ that is kernel-compatible with the PC pair $p_n$ and $q_m$.

1: if $m$, $n$, $l$ are input nodes then
2: return $E_{p_n,q_m}[k_l]$
3: else if $m$, $n$, $l$ are sum nodes then
4: return $\sum_{i\in in(n), j\in in(m), c\in in(l)} w_i w_j' w_c'' E_{p_i,q_j}[k_c]$
5: else if $m$, $n$, $l$ are product nodes then
6: return $E_{p_{nL},q_{mL}}[k_L] \cdot E_{p_{nR},q_{mR}}[k_R]$

Computation can be done in one forward pass!
smooth + decomposable + compatible = tractable $E[k]$

Algorithm 2 $E_{p_n,q_m}[k_l]$ — Computing the expected kernel

**Input:** Two compatible PCs $p_n$ and $q_m$, and a KC $k_l$ that is kernel-compatible with the PC pair $p_n$ and $q_m$.

1: if $m, n, l$ are input nodes then
2: return $E_{p_n,q_m}[k_l]$
3: else if $m, n, l$ are sum nodes then
4: return $\sum_{i \in \text{in}(n), j \in \text{in}(m), c \in \text{in}(l)} w_i w'_j w''_c E_{p_i,q_j}[k_c]$
5: else if $m, n, l$ are product nodes then
6: return $E_{p_{nL},q_{mL}}[k_L] \cdot E_{p_{nR},q_{mR}}[k_R]$

Computation can be done in one forward pass!

⇒ squared maximum mean discrepancy $MMD[p, q]$ [Gretton et al. 2012]
⇒ + determinism, kernelized discrete Stein discrepancy (KDSD) [Yang et al. 2018]
Applications

Support vector regression with missing features
Given training data, we can learn a support vector regression (SVR) model \( f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b; \) also we can learn a generative model for features in PC \( p(X). \)
Support vector regression with missing features

Given training data,

we can learn a support vector regression (SVR) model \( f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b; \)

also we can learn a generative model for features in PC \( p(X). \)
Support vector regression with missing features

- Given training data,
- we can learn a support vector regression (SVR) model $f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b$;
- also we can learn a generative model for features in PC $p(X)$.
Given training data,
we can learn a *support vector regression (SVR) model* \( f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b; \)
also we can learn a *generative model* for features in \( \text{PC} \ p(X). \)
Support vector regression with missing features

- Given training data,
- we can learn a support vector regression (SVR) model \( f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b; \)
- also we can learn a generative model for features in PC \( p(X) \).

At deployment time, what happen if we observe partial features and some are missing?
Support vector regression with missing features

- Given training data,
- we can learn a support vector regression (SVR) model $f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b$;
- also we can learn a generative model for features in PC $p(X)$.

At deployment time, what happen if we observe partial features and some are missing?

⇒ Expected prediction!
Given training data,

we can learn a support vector regression (SVR) model \( f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b; \)

also we can learn a generative model for features in PC \( p(X). \)

At deployment time, in the case when only features \( X_o = x_o \) are observed
and features \( X_m \) are missing, with \( X = (X_o, X_m) \), the expected
prediction is
Support vector regression with missing features

Given training data,
we can learn a support vector regression (SVR) model \( f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b; \)
also we can learn a generative model for features in \( \text{PC} \ p(X). \)

At deployment time, in the case when only features \( X_o = x_o \) are observed
and features \( X_m \) are missing, with \( X = (X_o, X_m) \), the expected prediction is

\[
\mathbb{E}_{x_m \sim p(X_m | x_o)} [f(x_o, x_m)]
\]
Support vector regression with missing features

Given training data,

we can learn a support vector regression (SVR) model $f(x) = \sum_{i=1}^{m} w_i k(x, x) + b$;

also we can learn a generative model for features in $PC p(X)$.

At deployment time, in the case when only features $X_o = x_o$ are observed and features $X_m$ are missing, with $X = (X_o, X_m)$, the expected prediction is

$$\mathbb{E}_{x_m \sim p(x_m|x_o)}[f(x, x)] = \sum_{i=1}^{m} w_i \mathbb{E}_{x_m \sim p(x_m|x_o)}[k(x, (x_o, x_m))] + b$$
Support vector regression with missing features

Expected prediction improves over the baselines
Applications

- Support vector regression with missing features
- Collapsed black-box importance sampling
Recap **Black-box Importance Sampling** [Liu et al. 2016]

Empirical KDSD \( \mathbb{S}(\{w^{(i)} , x^{(i)}\}_{i=1}^{n} \parallel p) \)

\[
\mathbb{S}^2(\{w^{(i)} , x^{(i)}\}_{i=1}^{n} \parallel p) = w^\top K_p w, \text{ with } [K_p]_{ij} = k_p(x^{(i)}, x^{(j)})
\]

Given a distribution \( p \), and samples \( \{x^{(i)}\}_{i=1}^{n} \), the black-box importance sampling obtains weights by solving optimization problem

\[
w^* = \arg\min_w \left\{ w^\top K_p w \mid \sum_{i=1}^{n} w_i = 1, w_i \geq 0 \right\}
\]
Recap **Black-box Importance Sampling** [Liu et al. 2016]

- **Empirical KDSD**
  \[
  \mathbb{S}(\{ w^{(i)}, x^{(i)} \}_{i=1}^n \parallel p)
  \]

  \[
  \mathbb{S}^2(\{ w^{(i)}, x^{(i)} \}_{i=1}^n \parallel p) = w^\top K_p w, \quad \text{with } [K_p]_{ij} = k_p(x^{(i)}, x^{(j)})
  \]

- **Given a distribution** \(p\), and samples \(\{x^{(i)}\}_{i=1}^n\), the black-box importance sampling obtains weights by solving optimization problem

  \[
  w^* = \underset{w}{\text{argmin}} \left\{ w^\top K_p w \mid \sum_{i=1}^n w_i = 1, \ w_i \geq 0 \right\}
  \]
Recap **Black-box Importance Sampling** [Liu et al. 2016]

Empirical KDSD \( S\left(\left\{ w^{(i)}, x^{(i)} \right\}_{i=1}^{n} \parallel p \right) \)

\[
S^{2}\left(\left\{ w^{(i)}, x^{(i)} \right\}_{i=1}^{n} \parallel p \right) = w^{\top} K_{p} w, \quad \text{with} \quad [K_{p}]_{ij} = k_{p}(x^{(i)}, x^{(j)})
\]

Given a distribution \( p \), and samples \( \left\{ x^{(i)} \right\}_{i=1}^{n} \), the black-box importance sampling obtains weights by solving optimization problem

\[
w^{*} = \arg\min_{w} \left\{ w^{\top} K_{p} w \right\} \quad \text{s.t.} \quad \sum_{i=1}^{n} w_{i} = 1, \quad w_{i} \geq 0
\]

*Can we use less samples but maintain the same or even better performance?*
Recap  Black-box Importance Sampling  [Liu et al. 2016]

- Empirical KDS
  \[ S(\{ w^{(i)}, x^{(i)} \}_{i=1}^n \| p) \]

  \[ S^2(\{ w^{(i)}, x^{(i)} \}_{i=1}^n \| p) = w^\top K_p w, \text{ with } [K_p]_{ij} = k_p(x^{(i)}, x^{(j)}) \]

- Given a distribution \( p \), and samples \( \{ x^{(i)} \}_{i=1}^n \), the black-box importance sampling obtains weights by solving optimization problem

  \[ w^* = \arg\min_w \left\{ w^\top K_p w \mid \sum_{i=1}^n w_i = 1, w_i \geq 0 \right\} \]

Can we use less samples but maintain the same or even better performance?  \( \Rightarrow \)  Collapsed samples!
Collapsed **Black-box Importance Sampling**

- Given partial samples \( \{x_s^{(i)}\}_{i=1}^n \), with \((X_s, X_c)\) a partition of \(X\),
- Represent the conditional distributions \(p(X_c | x_s^{(i)})\) as PCs \(p_i\) by knowledge compilation [Shen et al. 2016]
- Compile the kernel function \(k(X_c, X_c')\) as KC \(k\)
- Empirical KDSD between collapsed samples and the target distribution \(p\)

\[
S_s^2(\{x_s^{(i)}, w_i\} \parallel p) = w^\top K_{p,s} w
\]

with \([K_{p,s}]_{ij} = \mathbb{E}_{x_c \sim p_i, x_c' \sim p_j} [k_p(x, x')]\]

- Finally, obtain the importance weights \(w\) by solving

\[
w^* = \arg\min_w \left\{ w^\top K_{p,s} w \right\} \quad \left| \begin{array}{l} \sum_{i=1}^n w_i = 1, \ w_i \geq 0 \end{array} \right\}
\]
Collapsed Black-box Importance Sampling

- Given partial samples \( \{x_s^{(i)}\}_{i=1}^n \), with \((X_s, X_c)\) a partition of \(X\),
- Represent the conditional distributions \(p(X_c | x_s^{(i)})\) as PCs \(p_i\) by knowledge compilation [Shen et al. 2016]
- Compile the kernel function \(k(X_c, X_c')\) as KC \(k\)
- Empirical KDSD between collapsed samples and the target distribution \(p\)

\[
S^2_s(\{x_s^{(i)}, w_i\} \parallel p) = w^\top K_{p,s} w
\]

with \([K_{p,s}]_{ij} = \mathbb{E}_{x_c \sim p_i, x_c' \sim p_j} [k_p(x, x')]\]

- Finally, obtain the importance weights \(w\) by solving

\[
\mathbf{w}^* = \arg\min_w \left\{ w^\top K_{p,s} w \bigg| \sum_{i=1}^n w_i = 1, w_i \geq 0 \right\}
\]
Collapsed **Black-box Importance Sampling**

- Given partial samples \( \{x_s^{(i)}\}_{i=1}^n \), with \((X_s, X_c)\) a partition of \(X\),
- Represent the conditional distributions \(p(X_c | x_s^{(i)})\) as PCs \(p_i\) by knowledge compilation [Shen et al. 2016]
- Compile the kernel function \(k(X_c, X_c')\) as KC \(k\)
- Empirical KDSD between collapsed samples and the target distribution \(p\)

\[
S^2_s(\{x_s^{(i)}, w_i\} \parallel p) = \mathbf{w}^\top K_{p,s} \mathbf{w}
\]

with \([K_{p,s}]_{ij} = \mathbb{E}_{x_c \sim p_i, x_c' \sim p_j} [k_p(x, x')]\)

- Finally, obtain the importance weights \(w\) by solving

\[
\mathbf{w}^* = \arg\min_{\mathbf{w}} \left\{ \mathbf{w}^\top K_{p,s} \mathbf{w} \left| \begin{array}{c}
\sum_{i=1}^n w_i = 1, \ w_i \geq 0
\end{array} \right. \right\}
\]
**Collapsed Black-box Importance Sampling**

- Given partial samples \( \{x_s^{(i)}\}_{i=1}^n \), with \((X_s, X_c)\) a partition of \(X\),
- Represent the conditional distributions \(p(X_c | x_s^{(i)})\) as PCs \(p_i\) by knowledge compilation [Shen et al. 2016]
- Compile the kernel function \(k(X_c, X_c')\) as KC \(k\)
- Empirical KDSD between collapsed samples and the target distribution \(p\)

\[
S^2_s(\{x_s^{(i)}, w_i\} \parallel p) = w^\top K_{p,s}w
\]

with \([K_{p,s}]_{ij} = \mathbb{E}_{x_c \sim p_i, x_c' \sim p_j}[k_p(x, x')]\]

- Finally, obtain the importance weights \(w\) by solving

\[
w^* = \underset{w}{\text{argmin}} \left\{ w^\top K_{p,s}w \ \middle| \ \sum_{i=1}^n w_i = 1, w_i \geq 0 \right\}
\]
**Collapsed Black-box Importance Sampling**

- Given partial samples \( \{x_s^{(i)}\}_{i=1}^n \), with \((X_s, X_c)\) a partition of \( X \),
- Represent the conditional distributions \( p(X_c | x_s^{(i)}) \) as PCs \( p_i \) by knowledge compilation [Shen et al. 2016]
- Compile the kernel function \( k(X_c, X_c') \) as KC \( k \)
- Empirical KDSD between collapsed samples and the target distribution \( p \)
  \[
  S^2_s(\{x_s^{(i)}, w_i\} \parallel p) = w^\top K_{p,s} w
  \]
  with \( [K_{p,s}]_{ij} = \mathbb{E}_{x_c \sim p_i, x_c' \sim p_j} [k_p(x, x')] \)
- Finally, obtain the importance weights \( w \) by solving
  \[
  w^* = \arg\min_w \left\{ w^\top K_{p,s} w \middle| \sum_{i=1}^n w_i = 1, w_i \geq 0 \right\}
  \]
Collapsed Black-box Importance Sampling

- Given partial samples \( \{ x_s^{(i)} \}_{i=1}^n \), with \((X_s, X_c)\) a partition of \(X\),
- Represent the conditional distributions \( p(X_c | x_s^{(i)}) \) as PCs \( p_i \) by knowledge compilation [Shen et al. 2016]
- Compile the kernel function \( k(X_c, X_c') \) as KC \( k \)
- Empirical KDSD between collapsed samples and the target distribution \( p \)

\[
S^2_s(\{x_s^{(i)}, w_i\} \parallel p) = w^\top K_{p,s} w
\]

with \( [K_{p,s}]_{ij} = \mathbb{E}_{x_c \sim p_i, x'_c \sim p_j} [k_p(x, x')] \)

- Finally, obtain the importance weights \( w \) by solving

\[
w^* = \operatorname{argmin}_w \left\{ w^\top K_{p,s} w \middle| \sum_{i=1}^n w_i = 1, w_i \geq 0 \right\}
\]
Collapsed Black-box Importance Sampling

⇒ methods with collapsed samples all outperform their non-collapsed counterparts
⇒ CBBIS performs equally well or better than other baselines

Friedman and Broeck, “Approximate Knowledge Compilation by Online Collapsed Importance Sampling”, 2018
Applications

- Support vector regression with missing features
- Collapsed black-box importance sampling
Conclusion

Takeaways

#1: you can be both tractable and expressive

#2: circuits are a foundation for tractable inference over kernels

What else?

What other applications would benefit from the tractable computation of the expected kernels?
More on circuits ...

Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Models
starai.cs.ucla.edu/papers/ProbCirc20.pdf

Probabilistic Circuits: Representations, Inference, Learning and Theory
youtube.com/watch?v=2RAG5-L9R70

Probabilistic Circuits
arranger1044.github.io/probabilistic-circuits/

Foundations of Sum-Product Networks for probabilistic modeling
tinyurl.com/w65po5d
Questions?
References

References II