Tractable Computation of Expected Kernels by Circuit Representations

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

Nov. 29th, 2021 - MSR – New England
Tractable Computation of Expected Kernels by Circuit Representations

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

Nov. 29th, 2021 - MSR – New England
Tractable Computation of Expected Kernels by Circuit Representations

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

Nov. 29th, 2021 - MSR – New England
Tractable Computation of Expected Kernels by Circuit Representations

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

Nov. 29th, 2021 - MSR – New England
Problem Setup

A Fundamental Task

**Given** two distributions $p$ and $q$, and a kernel function $k$,

**Goal** is to compute the expected kernel tractably

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

A Fundamental Task

*Given* two distributions $p$ and $q$, and a kernel function $k$,

*Goal* is to compute the *expected kernel* tractably

$$E_{x \sim p, x' \sim q}[k(x, x')] .$$

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

A Fundamental Task

**Given** two distributions $p$ and $q$, and a kernel function $k$,

**Goal** is to compute the *expected kernel* tractably

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

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

$D(p, q)$
Problem Setup

A Fundamental Task

Given two distributions $p$ and $q$, and a kernel function $k$,

Goal is to compute the expected kernel tractably

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

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

$$\mathcal{D}(p, q) \quad \text{squared Maximum Mean Discrepancy (MMD)}$$

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

A Fundamental Task

**Given** two distributions $p$ and $q$, and a kernel function $k$,

**Goal** is to compute the *expected kernel* tractably

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

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

(Discrete) Kernelized Stein Discrepancy (KDSD)

\[ D(p, q) \]
Problem Setup

A Fundamental Task

Given two distributions $p$ and $q$, and a kernel function $k$,

Goal 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!

This talk how to compute the expected kernels exactly and tractably, by leveraging recent advances in probabilistic circuit representations.
Outline

- Problem Setup
- **Motivation: SVR with Missingness**
- Circuit Representation
- Approach: Tractable Expected Kernels
- Application: Collapsed Black-box Importance Sampling
Motivation

Example: Support vector regression with missing features
Motivation

Example: Support vector regression with missing features

Given training data,
Motivation

Example: Support vector regression with missing features

Given training data, and a learned support vector regression (SVR) model

\[ f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b, \]
Motivation

Example: Support vector regression with missing features

Given training data, and a learned support vector regression (SVR) model

\[ f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b, \]

Task at deployment time, what happen if we only observe partial features and some are missing?
**Motivation**

Example: Support vector regression with missing features

**Given** training data, and a learned *support vector regression (SVR) model*

\[ f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b, \]

**Task** at deployment time, what happen if we only observe partial features and some are missing?

⇒ *Expected prediction!*

---

8/38
**Motivation**

*Example: Support vector regression with missing features*

**Given** training data, and a learned support vector regression (SVR) model

\[ f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b, \]

With **Missing Features** . . .
Motivation

Example: Support vector regression with missing features

Given training data, and a learned support vector regression (SVR) model

\[ f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b, \]

With Missing Features . . .

- first learn a generative model for features in Probabilistic Circuit PC \( p(X) \) from training data;
Motivation

Example: Support vector regression with missing features

Given training data, and a learned support vector regression (SVR) model

\[ f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b, \]

With Missing Features . . .

1. first learn a generative model for features in Probabilistic Circuit PC \( p(X) \) from training data;
2. when only features \( X_o = x_o \) are observed and features \( X_m \) are missing, the expected prediction is

\[ \mathbb{E}_{x_m \sim p(x_m|x_o)}[f(x_o, x_m)] \]
Motivation

Example: Support vector regression with missing features

Given training data, and a learned support vector regression (SVR) model

$$f(x) = \sum_{i=1}^{m} w_i k(x_i, x) + b,$$

With Missing Features . . .

- first learn a generative model for features in Probabilistic Circuit PC $p(X)$ from training data;
- when only features $X_o = x_o$ are observed and features $X_m$ are missing, the expected prediction is

$$\mathbb{E}_{x_m \sim p(x_m|x_o)}[f(x_o, x_m)] = \sum_{i=1}^{m} w_i \mathbb{E}_{x_m \sim p(x_m|x_o)}[k(x_i, (x_o, x_m))] + b$$
Motivation

**Example: Support vector regression with missing features**

\[ \Rightarrow \text{Expected prediction improves over the baselines} \]
\[ \mathbb{E}_{x \sim p, x' \sim q}[k(x, x')] = \int_{x, x'} p(x) q(x') k(x, x') \, dx \, dx' \]
Challenge

Reliability vs. Flexibility

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

Tractable if \( p, q \) fully factorized
Challenge

Reliability vs. Flexibility

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

Tractable if \( p, q \) fully factorized

**PRO.** Tractable exact computation

**CON.** Model being too restrictive
Challenge
Reliability vs. Flexibility

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

Tractable if \( p, q \) fully factorized

**PRO.** Tractable exact computation
**CON.** Model being too restrictive

Hard to compute in general.

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

**PRO.** Efficient computation
**CON.** Slow convergence
**Challenge**

Reliability vs. Flexibility

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

Tractable if \( p, q \) fully factorized

**trade-off?** Hard to compute in general.

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

**PRO.** Tractable exact computation

**CON.** Model being too restrictive

**PRO.** Efficient computation

**CON.** Slow convergence
Expressive distribution models +

Exact computation of expected kernels?
Expressive distribution models

+  

Exact computation of expected kernels

=  

Circuits!
Outline

- Problem Setup
- Motivation: SVR with Missingness
- Circuit Representation
- Approach: Tractable Expected Kernels
- Application: Collapsed Black-box Importance Sampling
Circuit Representation

Probabilistic Circuits

deep generative models + guarantees
Circuit Representation

Probabilistic Circuits
deep generative models + guarantees

Kernel Circuits
express kernels as circuits
Probabilistic Circuits (PCs)

Tractable computational graphs

1. A simple tractable distribution is a PC
   \[ \implies \text{e.g., a multivariate Gaussian} \]
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 = \textit{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 are expressive deep generative models!

⇒ we can learn PCs with millions of parameters in minutes on the GPU [Peharz et al. 2020]
**On par with intractable models!**

*How expressive are PCs?*

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PCs</th>
<th>IDF</th>
<th>Hierarchical VAE</th>
<th>PixelVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MNIST</strong></td>
<td>1.20</td>
<td>1.90</td>
<td>1.27</td>
<td>1.39</td>
</tr>
<tr>
<td><strong>FashionMNIST</strong></td>
<td>3.34</td>
<td>3.47</td>
<td><strong>3.28</strong></td>
<td>3.66</td>
</tr>
<tr>
<td><strong>EMNIST (Letter split)</strong></td>
<td><strong>1.80</strong></td>
<td>1.95</td>
<td>1.84</td>
<td>2.26</td>
</tr>
<tr>
<td><strong>EMNIST (ByClass split)</strong></td>
<td><strong>1.85</strong></td>
<td>1.98</td>
<td>1.87</td>
<td>2.23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th><strong>CIFAR10</strong></th>
<th>ImageNet32</th>
<th>ImageNet64</th>
</tr>
</thead>
<tbody>
<tr>
<td>RealNVP</td>
<td>3.49</td>
<td>4.28</td>
<td>3.98</td>
</tr>
<tr>
<td>Glow</td>
<td>3.35</td>
<td>4.09</td>
<td>3.81</td>
</tr>
<tr>
<td>IDF</td>
<td>3.32</td>
<td>4.15</td>
<td>3.90</td>
</tr>
<tr>
<td>IDF++</td>
<td><strong>3.24</strong></td>
<td>4.10</td>
<td>3.81</td>
</tr>
<tr>
<td><strong>PCs+IDF</strong></td>
<td>3.28</td>
<td><strong>3.99</strong></td>
<td><strong>3.71</strong></td>
</tr>
</tbody>
</table>

*Liu et al., “Lossless Compression with Probabilistic Circuits”, 2021*
PCs = \textcolor{teal}{deep learning} + \textcolor{purple}{deep guarantees}

PCs are expressive \textit{deep generative models}!

\&

\textit{Certifying tractability} for a class of queries

\textit{via}

\textit{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.

**decomposable circuit**

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

---

**decomposable** PCs = ...

**MAR** \textit{sufficient} and \textit{necessary} conditions for computing any marginal

\[ p(y) = \int p(z, y) \, dz \]

\[ \rightarrow \text{by a single feedforward evaluation} \]

---

**decomposable PCs**

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

\[ \text{CON} \text{ sufficient and necessary conditions for any conditional distribution} \]

\[ p(y \mid z) = \frac{p(y, z)}{\int p(y, z) \, dz} \]

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

**decomposable** PCs = ...

**MAR**  
**sufficient** and **necessary** conditions for computing any marginal  
\[ \int p(z, y) \, dz \]

**CON**  
**sufficient** and **necessary** conditions for any conditional  
\[ \frac{p(y, z)}{\int p(y, z) \, dz} \]

---

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

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

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

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

1. $\exp(-|X_1 - X'_1|^2)$  
2. $\exp(-|X_2 - X'_2|^2)$  
3. $\exp(-|X_3 - X'_3|^2)$  
4. $\exp(-|X_4 - X'_4|^2)$

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

Common kernels can be compactly represented as decomposable KCs:

RBF, (exponentiated) Hamming, polynomial ...
Outline

- Problem Setup
- Motivation: SVR with Missingness
- Circuit Representation
- Approach: Tractable Expected Kernels
- Application: Collapsed Black-box Importance Sampling
Expected Kernel

tractable computation via circuit operations

Main result.
**Expected Kernel**

tractable computation via circuit operations

**Main result.** If PCs $p$ and $q$, and KC $k$ decompose in the same way,
**Expected Kernel**

tractable computation via circuit operations

**Main result.** If PCs $p$ and $q$, and KC $k$ decompose in the same way,

$$
\begin{align*}
\{X_1, X_2, X_3\} & \{X_4\} \\
\{X'_1, X'_2, X'_3\} & \{X'_4\}
\end{align*}
$$
Expected Kernel

tractable computation via circuit operations

Main result. If PCs $p$ and $q$, and KC $k$ decompose in the same way,
then computing expected kernels can be done tractably by one forward pass
$\Rightarrow$ product of the sizes of each circuit!
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 $k(x, x') = \sum_l w''_l k_l(x, x')$:
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 $k(X, X') = \sum_l w''_l k_l(X, X')$:

$$\sum_{x, x'} p(x) q(x') k(x, x') = \sum_{i, j, l} w_i w'_j w''_l p_i(x) q_j(x') k_l(x, x')$$
**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 $k(x, x') = \sum_l w''_l k_l(x, x')$:

\[
\begin{align*}
E_{p, q}[k(x, x')] &= \sum_{i,j,l} w_i w'_j w''_l E_{p_i, q_j}[k_l(x, x')] \\
\implies \text{expectation is "pushed down" to children}
\end{align*}
\]
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 \( k_x(x, x') = \prod_i k_i(x_i, x'_i) \):
**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 $k_x(x, X') = \prod_i k_i(x_i, x'_i)$:

\[
\sum_{x, x'} p_x(x) q_x(x') k_x(x, x') = \sum_{x, x'} \prod_i p(x_i) q(x_i) k_i(x_i, x'_i)
\]

= \prod_i (\sum_{x_i, x'_i} p(x_i) q(x_i) k_i(x_i, x'_i))
**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 \( k_x(X, X') = \prod_i k_i(X_i, X'_i) \):

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

\( \Rightarrow \) expectation decomposes into easier ones
decomposable + compatible = tractable $E[k]$
**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!

$\Rightarrow$ squared maximum mean discrepancy $MMD[p, q]$ [Gretton et al. 2012]

$\Rightarrow$ + determinism, kernelized discrete Stein discrepancy (KDSD) [Yang et al. 2018]
Outline

- Problem Setup
- Motivation: SVR with Missingness
- Circuit Representation
- Approach: Tractable Expected Kernels
- Application: Collapsed Black-box Importance Sampling
Recap **Black-box Importance Sampling** [Liu et al. 2016]

**Given** a target distribution $\mathbf{p}$, and samples $\{\mathbf{x}^{(i)}\}_{i=1}^n$. 
Recap *Black-box Importance Sampling* [Liu et al. 2016]

*Given* a target distribution $p$, and samples $\{x^{(i)}\}_{i=1}^n$,

*Task* is how to obtain weights $\mathbf{w}$ such that $\{w^{(i)}, x^{(i)}\}$ approximates $p$?
Recap **Black-box Importance Sampling** [Liu et al. 2016]

**Given** a target distribution $p$, and samples $\{x^{(i)}\}_{i=1}^n$,

**Task** is how to obtain weights $w$ such that $\{w^{(i)}, x^{(i)}\}$ approximates $p$?

The black-box importance sampling obtains weights by minimizing empirical KDSD:
Recap **Black-box Importance Sampling** [Liu et al. 2016]

Given a target distribution $p$, and samples $\{x^{(i)}\}_{i=1}^n$, 

Task is how to obtain weights $w$ such that $\{w^{(i)}, x^{(i)}\}$ approximates $p$?

The black-box importance sampling obtains weights by minimizing empirical KDSD:

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

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

Given a target distribution $p$, and samples $\{x^{(i)}\}_{i=1}^n$,

**Task** is how to obtain weights $w$ such that $\{w^{(i)}, x^{(i)}\}$ approximates $p$?

The black-box importance sampling obtains weights by minimizing empirical KDSD:

- empirical KDSD $\mathcal{S}(\{w^{(i)}, x^{(i)}\}_{i=1}^n \mid p) = w^\top K_p w$, with $[K_p]_{ij} = k_p(x^{(i)}, x^{(j)})$

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

**Given** a target distribution $p$, and samples $\{x^{(i)}\}_{i=1}^{n}$,

**Task** is how to obtain weights $w$ such that $\{w^{(i)}, x^{(i)}\}$ approximates $p$?

The black-box importance sampling obtains weights by minimizing empirical KDSD:

- empirical KDSD $S(\{w^{(i)}, x^{(i)}\}_{i=1}^{n} \parallel p) = w^\top K_p w$, with $[K_p]_{ij} = k_p(x^{(i)}, x^{(j)})$

- solving optimization problem $w^* = \arg\min_w \{w^\top K_p w \mid \sum_{i=1}^{n} w_i = 1, w_i \geq 0\}$

*Complexity quadratic in the number of samples $O(N^2)$!*
Recap \textbf{Black-box Importance Sampling} [Liu et al. 2016]

Given a target distribution $p$, and samples $\{x^{(i)}\}_{i=1}^n$.

Task is how to obtain weights $w$ such that $\{w^{(i)}, x^{(i)}\}$ approximates $p$?

The black-box importance sampling obtains weights by minimizing empirical KDSD:

\begin{align*}
\text{empirical KDSD } S(\{w^{(i)}, x^{(i)}\}_{i=1}^n \parallel p) &= w^T K_p w, \quad \text{with } [K_p]_{ij} = k_p(x^{(i)}, x^{(j)}) \\
\end{align*}

\begin{align*}
\text{solving optimization problem } w^* &= \arg\min_w \{w^T K_p w \mid \sum_{i=1}^n w_i = 1, w_i \geq 0\}
\end{align*}

\text{Complexity quadratic in the number of samples } \mathcal{O}(N^2)!

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

Given a target distribution $\mathbf{p}$, and samples $\{\mathbf{x}^{(i)}\}_{i=1}^{n}$, the task is how to obtain weights $\mathbf{w}$ such that $\{w^{(i)}, \mathbf{x}^{(i)}\}$ approximates $\mathbf{p}$?

The black-box importance sampling obtains weights by minimizing empirical KDSD:

$$\text{empirical KDSD} \ S(\{\begin{array}{c} w^{(i)} \, \, \mathbf{x}^{(i)} \end{array}\}_{i=1}^{n} || \mathbf{p}) = \mathbf{w}^{\top} \mathbf{K}_{p} \mathbf{w}, \text{ with } [\mathbf{K}_{p}]_{ij} = k_{p}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

solving optimization problem $\mathbf{w}^{\ast} = \arg\min_{\mathbf{w}} \left\{ \mathbf{w}^{\top} \mathbf{K}_{p} \mathbf{w} \mid \sum_{i=1}^{n} w_{i} = 1, w_{i} \geq 0 \right\}$

**Complexity quadratic in the number of samples** $\mathcal{O}(N^2)$!

Can we use less samples but maintain the same or even better performance?  

⇒ **Collapsed samples!**
**Collapsed Black-box Importance Sampling**

- 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 $p$

$$S_s^2(\{x_s^{(i)}, w_i\} \| p) = w^\top K_{p,s} w$$

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

$$w^* = \arg \min_w \left\{ w^\top K_{p,s} w \left| \sum_{i=1}^n w_i = 1, w_i \geq 0 \right. \right\}$$
**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

$$w^* = \arg\min_w \left\{ w^\top K_{p,s} w \mid \sum_{i=1}^n w_i = 1, w_i \geq 0 \right\}$$
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 \( p(X_c \mid x_s^{(i)}) \) 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

\[
\mathbf{w}^* = \arg\min_w \left\{ w^\top K_{p,s} w \left| \sum_{i=1}^n w_i = 1, w_i \geq 0 \right. \right\}
\]
**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 \( p(X_c \mid x_s^{(i)}) \) 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

\[
w^* = \arg\min_w \left\{ w^\top K_{p,s} w \left| \sum_{i=1}^n w_i = 1, w_i \geq 0 \right. \right\}
\]
**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 \( p(X_c \mid x_s^{(i)}) \) 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

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

$$
\mathbf{w}^* = \arg\min \left\{ w^\top K_{p,s} w \mid \sum_{i=1}^n w_i = 1, \ w_i \geq 0 \right\}
$$
methods with collapsed samples all outperform their non-collapsed counterparts

CBBIS performs equally well or better than other baselines

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

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