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 $\mathbf{p}$ and $\mathbf{q}$, and a kernel function $\mathbf{k}$,

Goal is to compute the *expected kernel* tractably

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

Problem Setup

A Fundamental Task

Given two distributions $\mathbf{p}$ and $\mathbf{q}$, and a kernel function $\mathbf{k}$,

Goal is to compute the *expected kernel* tractably

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

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

Problem Setup

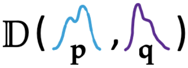
A Fundamental Task

Given two distributions $\mathbf{p}$ and $\mathbf{q}$, and a kernel function $\mathbf{k}$,

Goal is to compute the *expected kernel* tractably

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

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



Problem Setup

A Fundamental Task

Given two distributions $\mathbf{p}$ and $\mathbf{q}$, and a kernel function $\mathbf{k}$,

Goal is to compute the *expected kernel* tractably

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

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

squared Maximum Mean Discrepancy (MMD)

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

Problem Setup

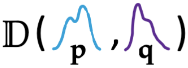
A Fundamental Task

Given two distributions $\mathbf{p}$ and $\mathbf{q}$, and a kernel function $\mathbf{k}$,

Goal is to compute the *expected kernel* tractably

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

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



(Discrete) Kernelized Stein Discrepancy (KDSD)

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

Problem Setup

A Fundamental Task

Given two distributions $\mathbf{p}$ and $\mathbf{q}$, and a kernel function $\mathbf{k}$,

Goal is to compute the *expected kernel*

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

$\Longrightarrow$ *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

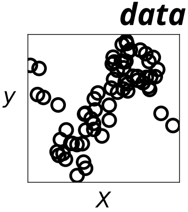
Motivation

Example: Support vector regression with missing features

Motivation

Example: Support vector regression with missing features

Given training data,



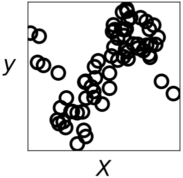
Motivation

Example: Support vector regression with missing features

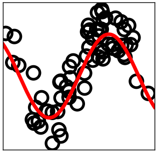
Given training data, and a learned *support vector regression (SVR) model*

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

data



SVR



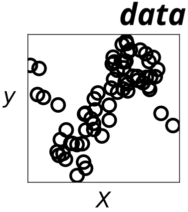
Motivation

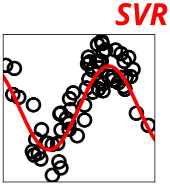
Example: Support vector regression with missing features

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

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

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



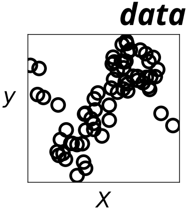


Motivation

Example: Support vector regression with missing features

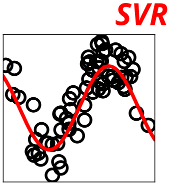
Given training data, and a learned *support vector regression (SVR) model*

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



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

$\Longrightarrow$ ***Expected prediction!***

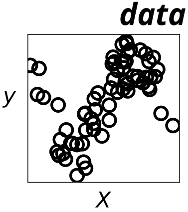


Motivation

Example: Support vector regression with missing features

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

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



With **Missing Features** . . .

Motivation

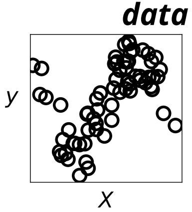
Example: Support vector regression with missing features

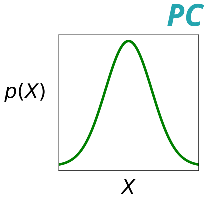
Given training data, and a learned *support vector regression (SVR) model*

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

With **Missing Features** . . .

- first learn a *generative model* for features in *Probabilistic Circuit* PC $\mathbf{p}(\mathbf{X})$ from training data;





Motivation

Example: Support vector regression with missing features

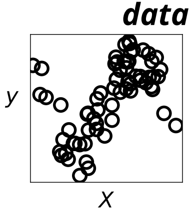
Given training data, and a learned *support vector regression (SVR) model*

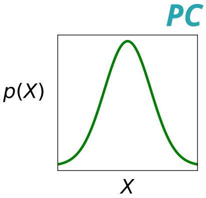
$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \mathbf{k}(\mathbf{x}_i, \mathbf{x}) + b,$$

With **Missing Features** . . .

- first learn a *generative model* for features in *Probabilistic Circuit* PC $\mathbf{p}(\mathbf{X})$ from training data;
- when only features $\mathbf{X}_o = \mathbf{x}_o$ are observed and features $\mathbf{X}_m$ are missing, the *expected prediction* is

$$\mathbb{E}_{\mathbf{x}_m \sim \mathbf{p}(\mathbf{X}_m | \mathbf{x}_o)} [f(\mathbf{x}_o, \mathbf{x}_m)]$$





Motivation

Example: Support vector regression with missing features

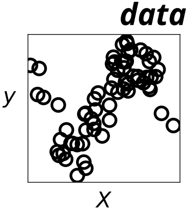
Given training data, and a learned *support vector regression (SVR) model*

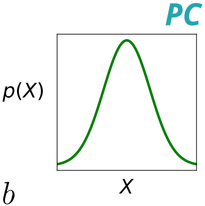
$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \mathbf{k}(\mathbf{x}_i, \mathbf{x}) + b,$$

With **Missing Features** . . .

- first learn a *generative model* for features in *Probabilistic Circuit* PC $\mathbf{p}(\mathbf{X})$ from training data;
- when only features $\mathbf{X}_o = \mathbf{x}_o$ are observed and features $\mathbf{X}_m$ are missing, the *expected prediction* is

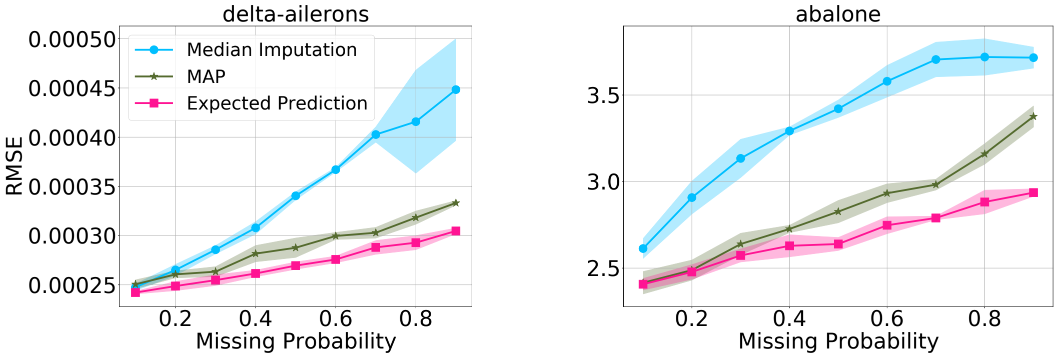
$$\mathbb{E}_{\mathbf{x}_m \sim \mathbf{p}(\mathbf{X}_m|\mathbf{x}_o)}[f(\mathbf{x}_o, \mathbf{x}_m)] = \sum_{i=1}^{m} w_i \mathbb{E}_{\mathbf{x}_m \sim \mathbf{p}(\mathbf{X}_m|\mathbf{x}_o)}[\mathbf{k}(\mathbf{x}_i, (\mathbf{x}_o, \mathbf{x}_m))] + b$$





Motivation

Example: Support vector regression with missing features



$\Longrightarrow$ *Expected prediction improves over the baselines*

Challenge

Reliability vs. Flexibility

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

Challenge

Reliability vs. Flexibility

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

Tractable if $\mathbf{p}, \mathbf{q}$ fully factorized

Challenge

Reliability vs. Flexibility

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

Tractable if $\mathbf{p}, \mathbf{q}$ fully factorized

PRO. Tractable exact computation

CON. Model being too restrictive

Challenge

Reliability vs. Flexibility

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

Tractable if $\mathbf{p}, \mathbf{q}$ fully factorized

PRO. Tractable exact computation

CON. Model being too restrictive

Hard to compute in general.
$\Longrightarrow$ *approximate with MC or variational inference*

PRO. Efficient computation

CON. Slow convergence

Challenge

Reliability vs. Flexibility

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

Tractable if $\mathbf{p}, \mathbf{q}$ fully factorized

PRO. Tractable exact computation

CON. Model being too restrictive

trade-off?

Hard to compute in general.

$\Longrightarrow$ *approximate with MC or variational inference*

PRO. Efficient computation

CON. Slow convergence

Expressive distribution models

+

Exact computation of expected kernels?

Expressive distribution models

+

Exact computation of expectated 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

I. *A simple tractable distribution is a PC*

$\Longrightarrow$ *e.g., a multivariate Gaussian*

X_1

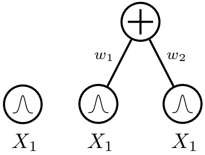
Probabilistic Circuits (PCs)

Tractable computational graphs

I. *A simple tractable distribution is a PC*

II. *A convex combination of PCs is a PC*

$\Rightarrow$ *e.g., a mixture model*



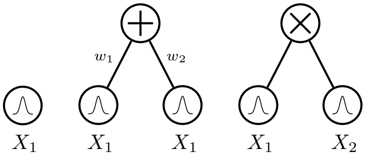
Probabilistic Circuits (PCs)

Tractable computational graphs

I. *A simple tractable distribution is a PC*

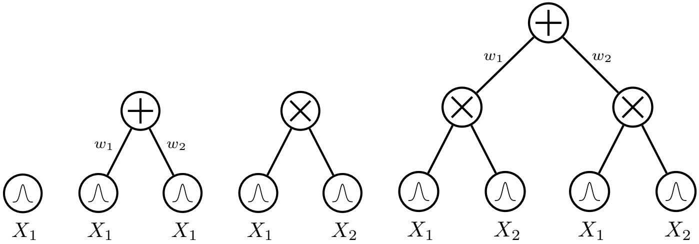
II. *A convex combination of PCs is a PC*

III. *A product of PCs is a PC*



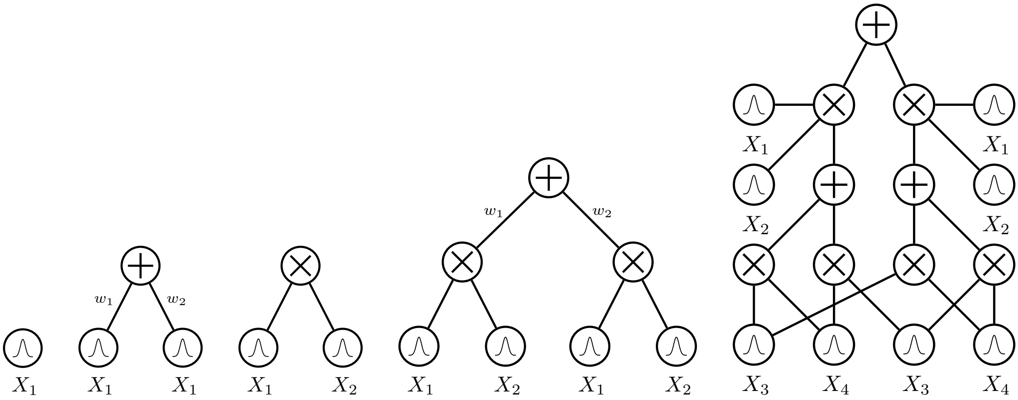
Probabilistic Circuits (PCs)

Tractable computational graphs



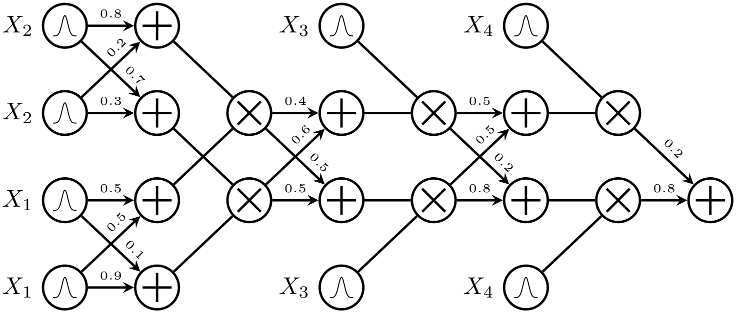
Probabilistic Circuits (PCs)

Tractable computational graphs



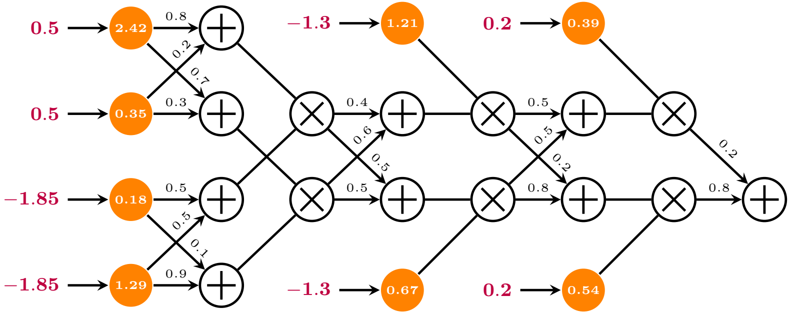
Probabilistic queries = feedforward evaluation

$$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$$



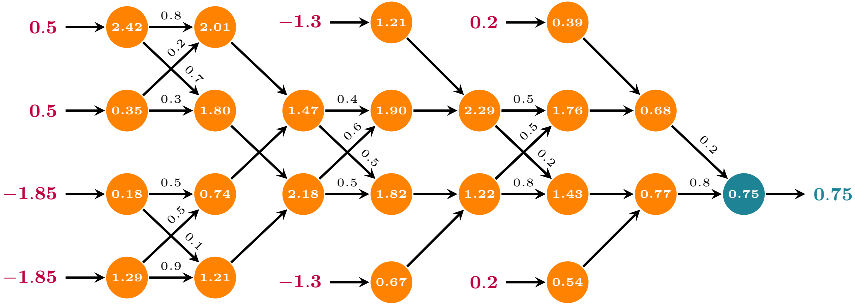
Probabilistic queries = feedforward evaluation

$$p(X_1 = \mathbf{-1.85}, X_2 = \mathbf{0.5}, X_3 = \mathbf{-1.3}, X_4 = \mathbf{0.2})$$



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***

$\Longrightarrow$ *stacking (categorical) latent variables*

PCs = *deep learning*

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

$\Longrightarrow$ *stacking (categorical) latent variables*

PCs are expressive *deep generative models*!

$\Longrightarrow$ *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?

Dataset	PCs	IDF	Hierarchical VAE	PixelVAE
MNIST	**1.20**	1.90	1.27	1.39
FashionMNIST	3.34	3.47	**3.28**	3.66
EMNIST (Letter split)	**1.80**	1.95	1.84	2.26
EMNIST (ByClass split)	**1.85**	1.98	1.87	2.23

Model	*CIFAR10*	*ImageNet32*	*ImageNet64*
RealNVP	3.49	4.28	3.98
Glow	3.35	4.09	3.81
IDF	3.32	4.15	3.90
IDF++	**3.24**	4.10	3.81
PCs+IDF	3.28	**3.99**	**3.71**

Liu et al., "Lossless Compression with Probabilistic Circuits", 2021

PCs = *deep learning* + *deep guarantees*

PCs are expressive ***deep generative models*!**

&

Certifying tractability for a class of queries

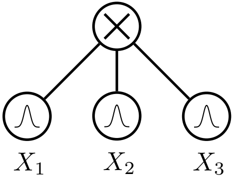
via

Verifying structural properties of the graph

Which structural constraints ensure tractability?

decomposable PCs

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 = ...

Choi et al., "Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Modeling", 2020

decomposable **PCs = ...**

MAR ***sufficient*** *and* ***necessary*** *conditions for computing any marginal*

$$p(\mathbf{y}) = \int p(\mathbf{z}, \mathbf{y})\, d\mathbf{z}$$

$\Longrightarrow$ *by a single feedforward evaluation*

Choi et al., "Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Modeling", 2020

decomposable **PCs = ...**

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

CON ***sufficient*** *and* ***necessary*** *conditions for any conditional distribution*

$$p(\mathbf{y} \mid \mathbf{z}) = \frac{p(\mathbf{y}, \mathbf{z})}{\int p(\mathbf{y}, \mathbf{z})\, d\mathbf{z}}$$

$\Longrightarrow$ *by* ***two*** *feedforward evaluations*

Choi et al., "Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Modeling", 2020

decomposable PCs = ...

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

CON ***sufficient*** *and* ***necessary*** *conditions for any conditional* $\frac{p(\mathbf{y},\mathbf{z})}{\int p(\mathbf{y},\mathbf{z})\, d\mathbf{z}}$

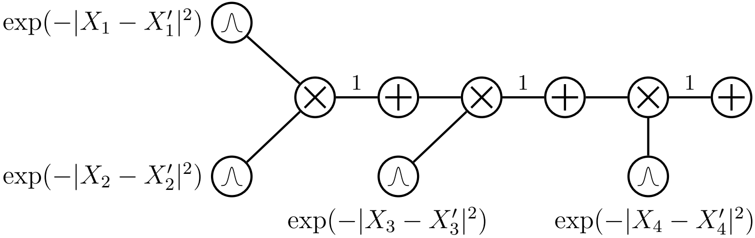
Choi et al., "Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Modeling", 2020

Can we represent kernels as circuits
to characterize tractability of its queries?

🤔

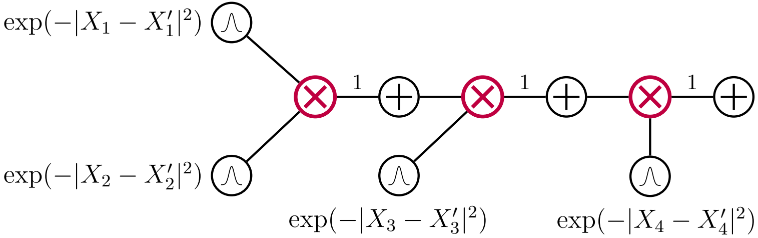
Kernel Circuits (KCs)

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



Kernel Circuits (KCs)

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



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

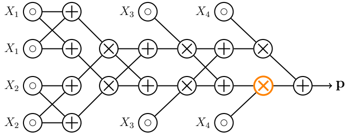
tractable computation via circuit operations

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

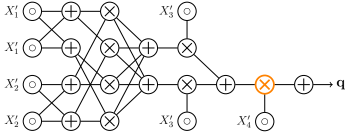
Expected Kernel

tractable computation via circuit operations

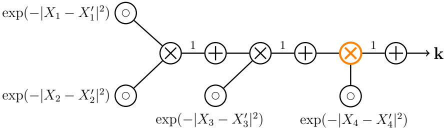
Main result. If PCs $\mathbf{p}$ and $\mathbf{q}$, and KC $\mathbf{k}$ *decompose in the same way*,



$\{X_1, X_2, X_3\}\{X_4\}$



$\{X_1', X_2', X_3'\}\{X_4'\}$



$\{(X_1, X_1'), (X_2, X_2'), (X_3, X_3')\}\{(X_4, X_4')\}$

Expected Kernel

tractable computation via circuit operations

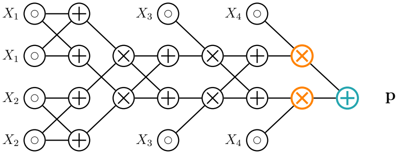
Main result. If PCs $\mathbf{p}$ and $\mathbf{q}$, and KC $\mathbf{k}$ *decompose in the same way*,

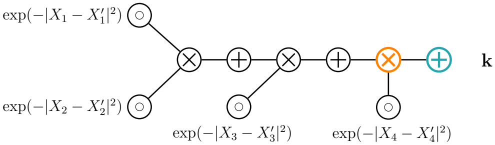
then computing expected kernels can be done ***tractably*** by one forward pass

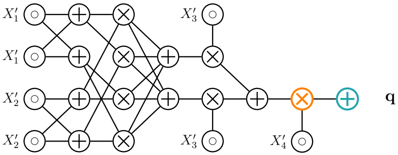
$\Longrightarrow$ *product of the sizes of each circuit!*

decomposable + compatible = tractable E[k]

[Sum Nodes] $\mathbf{p}(\mathbf{X}) = \sum_i w_i \mathbf{p}_i(\mathbf{X})$, $\mathbf{q}(\mathbf{X}') = \sum_j w'_j \mathbf{q}_j(\mathbf{X}')$, and kernel $\mathbf{k}(\mathbf{X}, \mathbf{X}') = \sum_l w''_l \mathbf{k}_l(\mathbf{X}, \mathbf{X}')$:

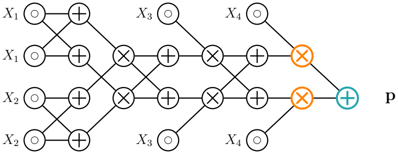


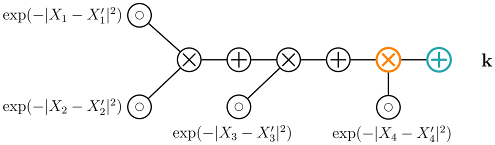


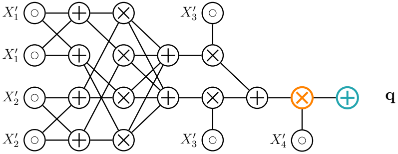


decomposable + compatible = tractable E[k]

[Sum Nodes] $\mathbf{p}(\mathbf{X}) = \sum_i w_i \mathbf{p}_i(\mathbf{X})$, $\mathbf{q}(\mathbf{X}') = \sum_j w'_j \mathbf{q}_j(\mathbf{X}')$, and kernel $\mathbf{k}(\mathbf{X}, \mathbf{X}') = \sum_l w''_l \mathbf{k}_l(\mathbf{X}, \mathbf{X}')$:



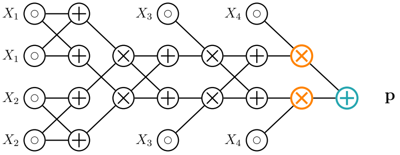


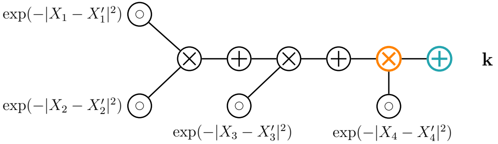


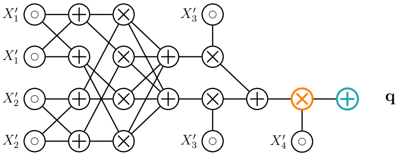
$$\sum_{\mathbf{x},\mathbf{x}'} \mathbf{p}(\mathbf{x})\mathbf{q}(\mathbf{x}')\mathbf{k}(\mathbf{x}, \mathbf{x}')$$
$$= \sum_{i,j,l} w_i w'_j w''_l \mathbf{p}_i(\mathbf{x})\mathbf{q}_j(\mathbf{x}')\mathbf{k}_l(\mathbf{x}, \mathbf{x}')$$

decomposable + compatible = tractable E[k]

[Sum Nodes] $\mathbf{p}(\mathbf{X}) = \sum_i w_i \mathbf{p}_i(\mathbf{X})$, $\mathbf{q}(\mathbf{X}') = \sum_j w'_j \mathbf{q}_j(\mathbf{X}')$, and kernel $\mathbf{k}(\mathbf{X}, \mathbf{X}') = \sum_l w''_l \mathbf{k}_l(\mathbf{X}, \mathbf{X}')$:





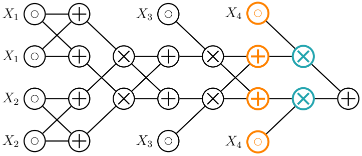


$$\mathbb{E}_{\mathbf{p},\mathbf{q}}[\mathbf{k}(\mathbf{x}, \mathbf{x}')] = \sum_{i,j,l} w_i w'_j w''_l \mathbb{E}_{\mathbf{p}_i,\mathbf{q}_j}[\mathbf{k}_l(\mathbf{x}, \mathbf{x}')]$$

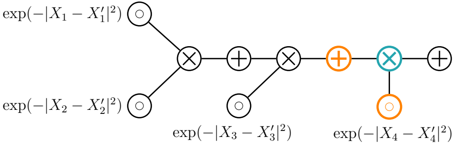
$\Longrightarrow$ *expectation is "pushed down" to children*

decomposable + compatible = tractable E[k]

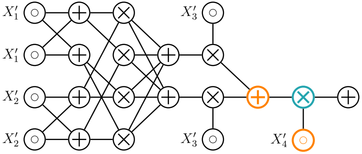
[Product Nodes] $\mathbf{p}_\times(\mathbf{X}) = \prod_i \mathbf{p}_i(\mathbf{X}_i)$, $\mathbf{q}_\times(\mathbf{X}') = \prod_i \mathbf{q}_j(\mathbf{X}'_i)$, and kernel $\mathbf{k}_\times(\mathbf{X}, \mathbf{X}') = \prod_i \mathbf{k}_i(\mathbf{X}_i, \mathbf{X}'_i)$:



$\mathbf{p}$



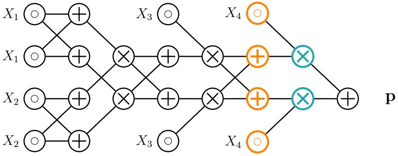
$\mathbf{k}$

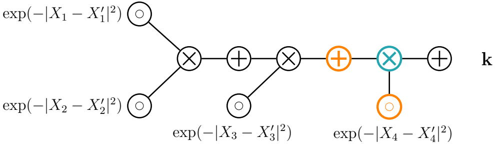


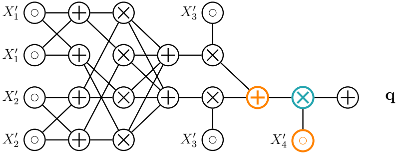
$\mathbf{q}$

decomposable + compatible = tractable E[k]

[Product Nodes] $\mathbf{p}_\times(\mathbf{X}) = \prod_i \mathbf{p}_i(\mathbf{X}_i)$, $\mathbf{q}_\times(\mathbf{X}') = \prod_i \mathbf{q}_j(\mathbf{X}'_i)$, and kernel $\mathbf{k}_\times(\mathbf{X}, \mathbf{X}') = \prod_i \mathbf{k}_i(\mathbf{X}_i, \mathbf{X}'_i)$:



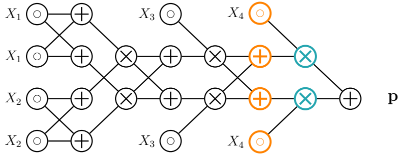


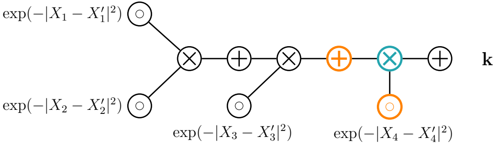


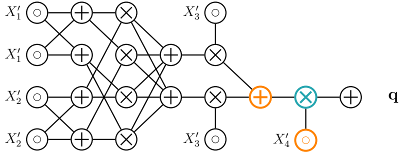
$$
\begin{aligned}
&\sum_{\mathbf{x},\mathbf{x}'} \mathbf{p}_\times(\mathbf{x})\mathbf{q}_\times(\mathbf{x}')\mathbf{k}_\times(\mathbf{x}, \mathbf{x}') \\
&= \sum_{\mathbf{x},\mathbf{x}'} \prod_i \mathbf{p}(\mathbf{x}_i)\mathbf{q}(\mathbf{x}_i)\mathbf{k}_i(\mathbf{x}_i, \mathbf{x}'_i) \\
&= \prod_i \Big(\sum_{\mathbf{x}_i,\mathbf{x}'_i} \mathbf{p}(\mathbf{x}_i)\mathbf{q}(\mathbf{x}_i)\mathbf{k}_i(\mathbf{x}_i, \mathbf{x}'_i)\Big)
\end{aligned}
$$

decomposable + compatible = tractable E[k]

[Product Nodes] $\mathbf{p}_\times(\mathbf{X}) = \prod_i \mathbf{p}_i(\mathbf{X}_i)$, $\mathbf{q}_\times(\mathbf{X}') = \prod_i \mathbf{q}_j(\mathbf{X}'_i)$, and kernel $\mathbf{k}_\times(\mathbf{X}, \mathbf{X}') = \prod_i \mathbf{k}_i(\mathbf{X}_i, \mathbf{X}'_i)$:







$$\mathbb{E}_{\mathbf{p}_\times, \mathbf{q}_\times}[\mathbf{k}_\times(\mathbf{x}, \mathbf{x}')] = \prod_i \mathbb{E}_{\mathbf{p},\mathbf{q}}[\mathbf{k}(\mathbf{x}_i, \mathbf{x}'_i)]$$

$\Longrightarrow$ *expectation decomposes into easier ones*

decomposable + **compatible** = **tractable E[k]**

Algorithm 1 $\mathbb{E}_{\mathbf{p}_n,\mathbf{q}_m}[\mathbf{k}_l]$ — Computing the expected kernel

Input: Two compatible PCs $\mathbf{p}_n$ and $\mathbf{q}_m$, and a KC $\mathbf{k}_l$ that is kernel-compatible with the PC pair $\mathbf{p}_n$ and $\mathbf{q}_m$.

1: **if** m, n, l are ***input*** nodes **then**
2:     **return** $\mathbb{E}_{\mathbf{p}_n,\mathbf{q}_m}[\mathbf{k}_l]$
3: **else if** m, n, l are ***sum*** nodes **then**
4:     **return** $\sum_{i\in\mathsf{in}(n),j\in\mathsf{in}(m),c\in\mathsf{in}(l)} w_i w'_j w''_c\, \mathbb{E}_{\mathbf{p}_i,\mathbf{q}_j}[\mathbf{k}_c]$
5: **else if** m, n, l are ***product*** nodes **then**
6:     **return** $\mathbb{E}_{\mathbf{p}_{n_\mathsf{L}},\mathbf{q}_{m_\mathsf{L}}}[\mathbf{k}_\mathsf{L}] \cdot \mathbb{E}_{\mathbf{p}_{n_\mathsf{R}},\mathbf{q}_{m_\mathsf{R}}}[\mathbf{k}_\mathsf{R}]$

Computation can be done in one forward pass!

decomposable + **compatible** = **tractable E[k]**

Algorithm 2 $\mathbb{E}_{\mathbf{p}_n,\mathbf{q}_m}[\mathbf{k}_l]$ — Computing the expected kernel

Input: Two compatible PCs $\mathbf{p}_n$ and $\mathbf{q}_m$, and a KC $\mathbf{k}_l$ that is kernel-compatible with the PC pair $\mathbf{p}_n$ and $\mathbf{q}_m$.

1: **if** m, n, l are ***input*** nodes **then**
2: **return** $\mathbb{E}_{\mathbf{p}_n,\mathbf{q}_m}[\mathbf{k}_l]$
3: **else if** m, n, l are ***sum*** nodes **then**
4: **return** $\sum_{i\in\mathsf{in}(n),j\in\mathsf{in}(m),c\in\mathsf{in}(l)} w_i w'_j w''_c \, \mathbb{E}_{\mathbf{p}_i,\mathbf{q}_j}[\mathbf{k}_c]$
5: **else if** m, n, l are ***product*** nodes **then**
6: **return** $\mathbb{E}_{\mathbf{p}_{n_\mathsf{L}},\mathbf{q}_{m_\mathsf{L}}}[\mathbf{k}_\mathsf{L}] \cdot \mathbb{E}_{\mathbf{p}_{n_\mathsf{R}},\mathbf{q}_{m_\mathsf{R}}}[\mathbf{k}_\mathsf{R}]$

Computation can be done in one forward pass!

$\Longrightarrow$ *squared maximum mean discrepancy* $MMD[\mathbf{p}, \mathbf{q}]$ *[Gretton et al. 2012]*

$\Longrightarrow$ *+ 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 $\mathbf{p}$, and samples $\{\mathbf{x}^{(i)}\}_{i=1}^{n}$,

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

Recap Black-box Importance Sampling [Liu et al. 2016]

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

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

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

Recap Black-box Importance Sampling [Liu et al. 2016]

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

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

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

- empirical KDSD $\mathbb{S}(\{\underbrace{w^{(i)}}_{\text{weights}}, \underbrace{\mathbf{x}^{(i)}}_{\text{samples}}\}_{i=1}^n \parallel \mathbf{p}) = \boldsymbol{w}^\top \boldsymbol{K_p} \boldsymbol{w}$, with $[\boldsymbol{K}_p]_{ij} = k_p(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$
- solving optimization problem $\boldsymbol{w}^* = \operatorname{argmin}_{\boldsymbol{w}} \{\boldsymbol{w}^\top \boldsymbol{K_p} \boldsymbol{w} \mid \sum_{i=1}^n w_i = 1,\ w_i \geq 0\}$

Recap Black-box Importance Sampling [Liu et al. 2016]

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

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

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

- empirical KDSD $\mathbb{S}(\{\underbrace{w^{(i)}}_{\text{weights}}, \underbrace{\mathbf{x}^{(i)}}_{\text{samples}}\}_{i=1}^{n} \parallel \mathbf{p}) = \boldsymbol{w}^\top \boldsymbol{K_p} \boldsymbol{w}$, with $[\boldsymbol{K}_p]_{ij} = k_p(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$
- solving optimization problem $\boldsymbol{w}^* = \mathrm{argmin}_{\boldsymbol{w}} \left\{\boldsymbol{w}^\top \boldsymbol{K_p} \boldsymbol{w} \,\middle|\, \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0\right\}$

Recap Black-box Importance Sampling [Liu et al. 2016]

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

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

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

- empirical KDSD $\mathbb{S}(\{\underbrace{w^{(i)}}_{\text{weights}}, \underbrace{\mathbf{x}^{(i)}}_{\text{samples}}\}_{i=1}^{n} \parallel \mathbf{p}) = \boldsymbol{w}^\top \boldsymbol{K_p} \boldsymbol{w}$, with $[\boldsymbol{K}_p]_{ij} = k_p(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$
- solving optimization problem $\boldsymbol{w}^* = \mathrm{argmin}_{\boldsymbol{w}} \left\{ \boldsymbol{w}^\top \boldsymbol{K_p} \boldsymbol{w} \,\middle|\, \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0 \right\}$

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

Recap Black-box Importance Sampling [Liu et al. 2016]

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

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

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

- empirical KDSD $\mathbb{S}(\{\underbrace{w^{(i)}}_{\text{weights}}, \underbrace{\mathbf{x}^{(i)}}_{\text{samples}}\}_{i=1}^{n} \parallel \mathbf{p}) = \boldsymbol{w}^{\top}\boldsymbol{K}_{\boldsymbol{p}}\boldsymbol{w}$, with $[\boldsymbol{K}_p]_{ij} = k_p(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$
- solving optimization problem $\boldsymbol{w}^* = \operatorname{argmin}_{\boldsymbol{w}} \{\boldsymbol{w}^{\top}\boldsymbol{K}_{\boldsymbol{p}}\boldsymbol{w} \mid \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0\}$

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

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 $\{\mathbf{x}^{(i)}\}_{i=1}^{n}$,

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

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

- empirical KDSD $\mathbb{S}(\{\underbrace{w^{(i)}}_{\text{weights}}, \underbrace{\mathbf{x}^{(i)}}_{\text{samples}}\}_{i=1}^{n} \parallel \mathbf{p}) = \boldsymbol{w}^\top \boldsymbol{K_p} \boldsymbol{w},$ with $[\boldsymbol{K}_p]_{ij} = k_p(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$
- solving optimization problem $\boldsymbol{w}^* = \operatorname{argmin}_{\boldsymbol{w}} \{\boldsymbol{w}^\top \boldsymbol{K_p} \boldsymbol{w} \mid \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0\}$

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

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

$\Longrightarrow$ ***Collapsed samples!***

Collapsed Black-box Importance Sampling

- Represent the conditional distributions $\mathbf{p}(\mathbf{X_c} \mid \mathbf{x_s}^{(i)})$ as PCs $\mathbf{p}_i$ by *knowledge compilation* *[Shen et al. 2016]*
- Compile the kernel function $\mathbf{k}(\mathbf{X_c}, \mathbf{X_c}')$ as KC $\mathbf{k}$
- Empirical KDSD between collapsed samples and the target distribution $\mathbf{p}$

$$\mathbb{S}^2_{\mathbf{s}}(\{\mathbf{x_s}^{(i)}, w_i\} \parallel p) = \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w}$$

with $[\boldsymbol{K}_{p,\mathbf{s}}]_{ij} = \mathbb{E}_{\mathbf{x_c} \sim \mathbf{p}_i, \mathbf{x}'_\mathbf{c} \sim \mathbf{p}_j} [\mathbf{k}_p(\mathbf{x}, \mathbf{x}')]$

- Finally, obtain the importance weights $\boldsymbol{w}$ by solving

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w} \;\middle|\; \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0 \right\}$$

Collapsed Black-box Importance Sampling

Given partial samples $\{\mathbf{x_s}^{(i)}\}_{i=1}^{n}$, with $(\mathbf{X_s}, \mathbf{X_c})$ a partition of $\mathbf{X}$,

- Represent the conditional distributions $\mathrm{p}(\mathbf{X_c} \mid \mathbf{x_s}^{(i)})$ as PCs p_i by *knowledge compilation* [Shen et al. 2016]
- Compile the kernel function $\mathrm{k}(\mathbf{X_c}, \mathbf{X_c}')$ as KC k
- Empirical KDSD between collapsed samples and the target distribution p

$$\mathbb{S}_{\mathbf{s}}^2(\{\mathbf{x_s}^{(i)}, w_i\} \parallel p) = \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w}$$

with $[\boldsymbol{K}_{p,\mathbf{s}}]_{ij} = \mathbb{E}_{\mathbf{x_c} \sim \mathrm{p}_i, \mathbf{x}'_\mathbf{c} \sim \mathrm{p}_j} [\mathrm{k}_p(\mathbf{x}, \mathbf{x}')]$

- Finally, obtain the importance weights $\boldsymbol{w}$ by solving

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w} \,\middle|\, \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0 \right\}$$

Collapsed Black-box Importance Sampling

Given partial samples $\{\mathbf{x_s}^{(i)}\}_{i=1}^{n}$, with $(\mathbf{X_s}, \mathbf{X_c})$ a partition of $\mathbf{X}$,

- Represent the conditional distributions $\mathbf{p}(\mathbf{X_c} \mid \mathbf{x_s}^{(i)})$ as PCs $\mathbf{p}_i$ by *knowledge compilation* *[Shen et al. 2016]*
- Compile the kernel function $\mathbf{k}(\mathbf{X_c}, \mathbf{X_c}')$ as KC $\mathbf{k}$
- Empirical KDSD between collapsed samples and the target distribution $\mathbf{p}$

$$\mathbb{S}_{\mathbf{s}}^2(\{\mathbf{x_s}^{(i)}, w_i\} \parallel p) = \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w}$$

with $[\boldsymbol{K}_{p,\mathbf{s}}]_{ij} = \mathbb{E}_{\mathbf{x_c} \sim \mathbf{p}_i, \mathbf{x}'_\mathbf{c} \sim \mathbf{p}_j} [\mathbf{k}_p(\mathbf{x}, \mathbf{x}')]$

- Finally, obtain the importance weights $\boldsymbol{w}$ by solving

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w} \;\middle|\; \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0 \right\}$$

Collapsed Black-box Importance Sampling

Given partial samples $\{\mathbf{x_s}^{(i)}\}_{i=1}^{n}$, with $(\mathbf{X_s}, \mathbf{X_c})$ a partition of $\mathbf{X}$,

- Represent the conditional distributions $\mathbf{p}(\mathbf{X_c} \mid \mathbf{x_s}^{(i)})$ as PCs $\mathbf{p}_i$ by *knowledge compilation* *[Shen et al. 2016]*
- Compile the kernel function $\mathbf{k}(\mathbf{X_c}, \mathbf{X_c}')$ as KC $\mathbf{k}$
- Empirical KDSD between collapsed samples and the target distribution $\mathbf{p}$

$$\mathbb{S}^2_{\mathbf{s}}(\{\mathbf{x_s}^{(i)}, w_i\} \parallel p) = \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w}$$

with $[\boldsymbol{K}_{p,\mathbf{s}}]_{ij} = \mathbb{E}_{\mathbf{x_c} \sim \mathbf{p}_i, \mathbf{x}'_\mathbf{c} \sim \mathbf{p}_j} [\mathbf{k}_p(\mathbf{x}, \mathbf{x}')]$

- Finally, obtain the importance weights $\boldsymbol{w}$ by solving

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w} \;\middle|\; \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0 \right\}$$

Collapsed Black-box Importance Sampling

Given partial samples $\{\mathbf{x_s}^{(i)}\}_{i=1}^{n}$, with $(\mathbf{X_s}, \mathbf{X_c})$ a partition of $\mathbf{X}$,

- Represent the conditional distributions $\mathbf{p}(\mathbf{X_c} \mid \mathbf{x_s}^{(i)})$ as PCs $\mathbf{p}_i$ by *knowledge compilation* [Shen et al. 2016]
- Compile the kernel function $\mathbf{k}(\mathbf{X_c}, \mathbf{X_c}')$ as KC $\mathbf{k}$
- Empirical KDSD between collapsed samples and the target distribution $\mathbf{p}$

$$\mathbb{S}^2_{\mathbf{s}}(\{\mathbf{x_s}^{(i)}, w_i\} \parallel p) = \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w}$$

with $[\boldsymbol{K}_{p,\mathbf{s}}]_{ij} = \mathbb{E}_{\mathbf{x_c} \sim \mathbf{p}_i, \mathbf{x}'_\mathbf{c} \sim \mathbf{p}_j} [\mathbf{k}_p(\mathbf{x}, \mathbf{x}')]$

- Finally, obtain the importance weights $\boldsymbol{w}$ by solving

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w} \,\middle|\, \sum_{i=1}^{n} w_i = 1,\; w_i \geq 0 \right\}$$

Collapsed Black-box Importance Sampling

Given partial samples $\{\mathbf{x_s}^{(i)}\}_{i=1}^{n}$, with $(\mathbf{X_s}, \mathbf{X_c})$ a partition of $\mathbf{X}$,

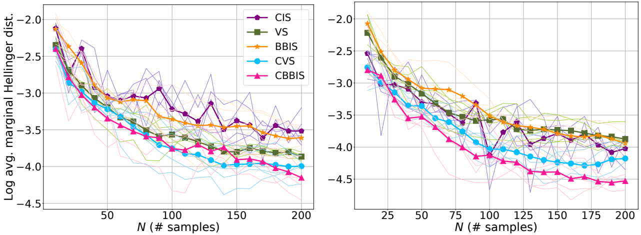
- Represent the conditional distributions $\mathbf{p}(\mathbf{X_c} \mid \mathbf{x_s}^{(i)})$ as PCs $\mathbf{p}_i$ by *knowledge compilation* *[Shen et al. 2016]*
- Compile the kernel function $\mathbf{k}(\mathbf{X_c}, \mathbf{X_c}')$ as KC $\mathbf{k}$
- Empirical KDSD between collapsed samples and the target distribution $\mathbf{p}$

$$\mathbb{S}_{\mathbf{s}}^2(\{\mathbf{x_s}^{(i)}, w_i\} \parallel p) = \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w}$$

  with $[\boldsymbol{K}_{p,\mathbf{s}}]_{ij} = \mathbb{E}_{\mathbf{x_c} \sim \mathbf{p}_i, \mathbf{x}'_\mathbf{c} \sim \mathbf{p}_j} [\mathbf{k}_p(\mathbf{x}, \mathbf{x}')]$
- Finally, obtain the importance weights $\boldsymbol{w}$ by solving

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} \left\{ \boldsymbol{w}^\top \boldsymbol{K}_{p,\mathbf{s}} \boldsymbol{w} \;\middle|\; \sum_{i=1}^{n} w_i = 1,\ w_i \geq 0 \right\}$$

Collapsed Black-box Importance Sampling



$\Rightarrow$ *methods with collapsed samples all outperform their non-collapsed counterparts*

$\Rightarrow$ *CBBIS performs equally well or better than other baselines*

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

Liu and Lee, "Black-box importance sampling", 2016

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 I

- Darwiche, Adnan and Pierre Marquis (2002). “A knowledge compilation map”. In: *Journal of Artificial Intelligence Research* 17, pp. 229–264.
- Liu, Qiang and Jason D Lee (2016). “Black-box importance sampling”. In: *arXiv preprint arXiv:1610.05247*.
- Friedman, Tal and Guy Van den Broeck (Dec. 2018). “Approximate Knowledge Compilation by Online Collapsed Importance Sampling”. In: *Advances in Neural Information Processing Systems 31 (NeurIPS)*. URL: `http://starai.cs.ucla.edu/papers/FriedmanNeurIPS18.pdf`.
- Choi, YooJung, Antonio Vergari, and Guy Van den Broeck (2020). “Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Modeling”. In:
- Peharz, Robert, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani (2020). “Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits”. In: *International Conference of Machine Learning*.
- Liu, Anji, Stephan Mandt, and Guy Van den Broeck (2021). “Lossless Compression with Probabilistic Circuits”. In: *arXiv preprint arXiv:2111.11632*.