Tractable Computation of Expected Kernels

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

Uncertaity in Artificial Intelligence

Expected Kernels

Motivation

Given two distributions $\mathbf{p}$ and $\mathbf{q}$, and a kernel $\mathbf{k}$, the task is to compute the *expected kernel*

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

Expected Kernels

Motivation

Given two distributions $\mathbf{p}$ and $\mathbf{q}$, and a kernel $\mathbf{k}$, the task 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!*

Expected Kernels

Motivation

Given two distributions $\mathbf{p}$ and $\mathbf{q}$, and a kernel $\mathbf{k}$, the task is to compute the *expected kernel*

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

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

squared Maximum Mean Discrepancy (MMD)

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

Expected Kernels

Motivation

Given two distributions $\mathbf{p}$ and $\mathbf{q}$, and a kernel $\mathbf{k}$, the task 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!*

Kernelized Discrete Stein Discrepancy (KDSD)

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

Expected Kernels

Motivation

Given two distributions $\mathbf{p}$ and $\mathbf{q}$, and a kernel $\mathbf{k}$, the task 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!*

Kernelized Support Vector Regressor (SVR) with missing features

$\mathbb{E}_{\mathbf{x}\sim\mathbf{p}}[\sum_i w_i \mathbf{k}(\mathbf{x}^{(i)},\mathbf{x}) + \boldsymbol{b}]$

Challenge

Reliability vs. Flexibility

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

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

PRO. Efficient computation

CON. *no guarantees* on error bounds

Challenge

Reliability vs. Flexibility

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

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

PRO. Efficient computation

CON. *no guarantees* on error bounds

$\mathbf{p}, \mathbf{q}, \mathbf{k}$ fully factorized

PRO. Tractable exact computation

CON. Model being too restrictive

Challenge

Reliability vs. Flexibility

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

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

PRO. Efficient computation

CON. *no guarantees* on error bounds

trade-off?

$\mathbf{p}, \mathbf{q}, \mathbf{k}$ fully factorized

PRO. Tractable exact computation

CON. Model being too restrictive

Circuits

Probabilistic Circuits

deep generative models + deep guarantees

Kernel Circuits

express kernels as circuits

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

Probabilistic Circuits (PCs)

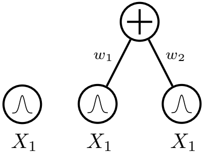
Tractable computational graphs

I. *A simple tractable distribution is a PC*

$\Longrightarrow$ *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*

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

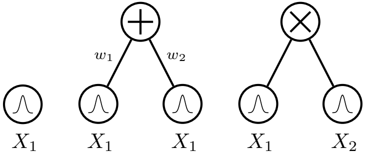
Probabilistic Circuits (PCs)

Tractable computational graphs

I. *A simple tractable distribution is a PC*

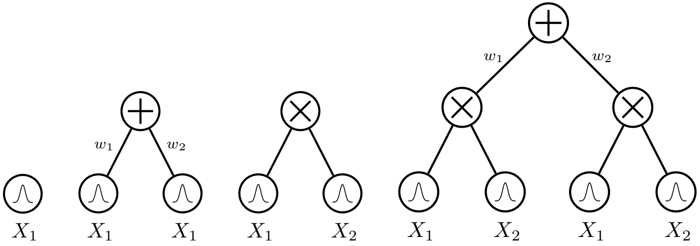
II. *A convex combination of PCs is a PC*

III. *A product of PCs is a PC*



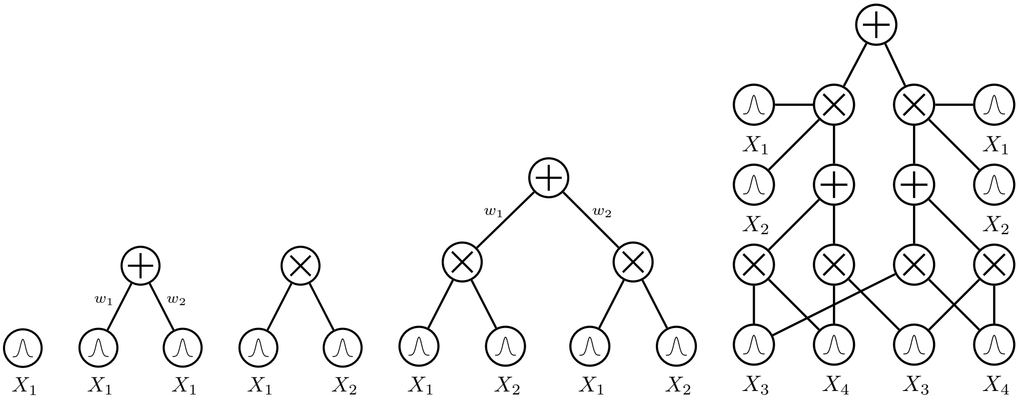
Probabilistic Circuits (PCs)

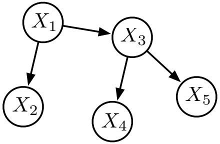
Tractable computational graphs



Probabilistic Circuits (PCs)

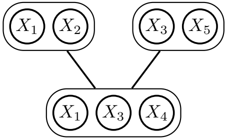
Tractable computational graphs





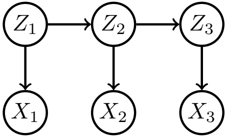
Chow-Liu trees

[Chow and Liu 1968]



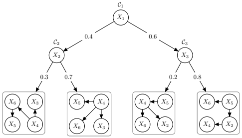
Junction trees

[Bach and Jordan 2001]



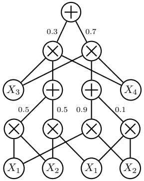
HMMs

[Rabiner and Juang 1986]



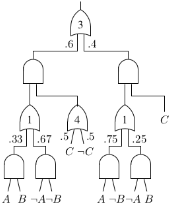
CNets

[Rahman et al. 2014]



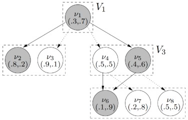
SPNs

[Poon et al. 2011]



PSDDs

[Kisa et al. 2014]



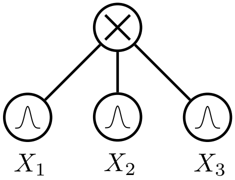
PDGs

[Jaeger 2004]

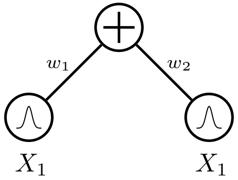
Which structural constraints ensure tractability?

decomposable + smooth PCs

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 on the same variable sets



decomposable circuit



smooth circuit

Darwiche and Marquis, "A knowledge compilation map", 2002

decomposable + smooth PCs = ...

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

CON $\frac{\int p(\mathbf{z},\mathbf{y},\mathbf{h})\, d\boldsymbol{H}}{\int \int p(\mathbf{z},\mathbf{y},\mathbf{h})\, d\mathbf{H}\, d\mathbf{Y}}$

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

decomposable + *smooth* PCs = ...

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

CON $\frac{\int p(\mathbf{z},\mathbf{y},\mathbf{h}) \, d\boldsymbol{H}}{\int \int p(\mathbf{z},\mathbf{y},\mathbf{h}) \, d\mathbf{H} \, d\mathbf{Y}}$

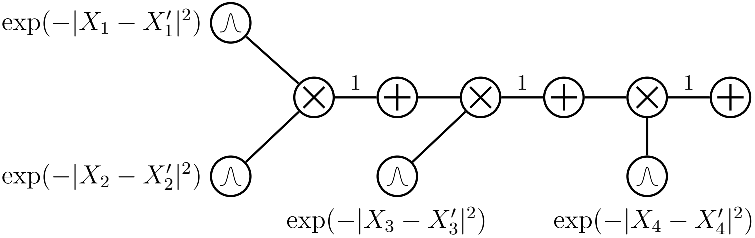
? *What about the* ***expected kernel*** $\mathbb{E}_{\mathbf{x} \sim \mathbf{p}, \mathbf{x}' \sim \mathbf{q}}[\mathbf{k}(\mathbf{x}, \mathbf{x}')]$*?*

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?

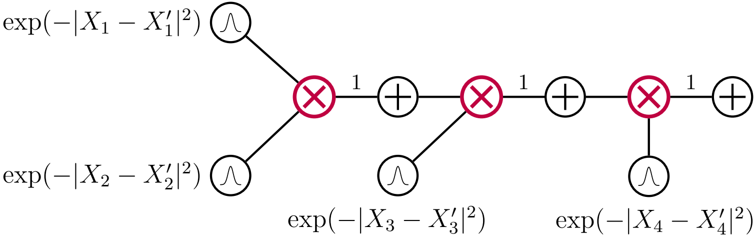
Kernel Circuits (KCs)

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)

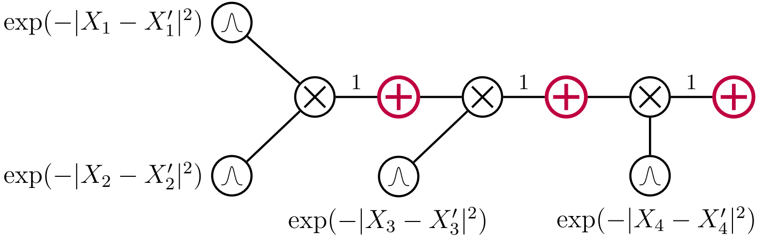
Exa. Radial basis function (RBF) kernel $\mathbf{k}(\mathbf{X}, \mathbf{X}') = \exp\left(-\sum_{i=1}^{4} \mid X_i - X_i' \mid^2\right)$



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

Kernel Circuits (KCs)

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

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 $\mathbf{p}$ and $\mathbf{q}$, and KC $\mathbf{k}$ are **decomposable** + **smooth**

Expected Kernel

tractable computation via circuit operations

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

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

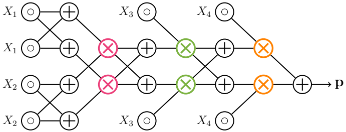
$\Longrightarrow$ *decompose in the same way*

Expected Kernel

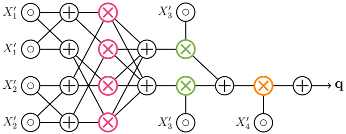
tractable computation via circuit operations

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

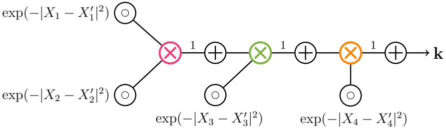
ii) PCs $\mathbf{p}$ and $\mathbf{q}$, and KC $\mathbf{k}$ are **compatible**



$\{X_1\}\{X_2\}$



$\{X_1'\}\{X_2'\}$



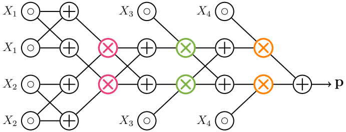
$\{(X_1, X_1')\}\{(X_2, X_2')\}$

Expected Kernel

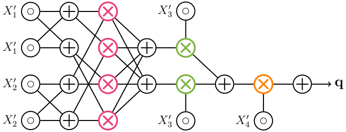
tractable computation via circuit operations

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

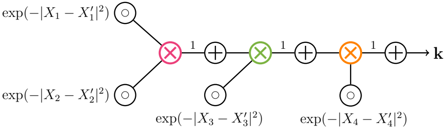
ii) PCs $\mathbf{p}$ and $\mathbf{q}$, and KC $\mathbf{k}$ are **compatible**



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



$\{X'_1, X'_2\}\{X'_3\}$



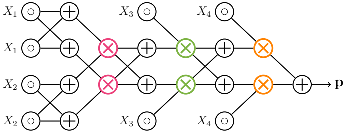
$\{(X_1, X'_1), (X_2, X'_2)\}\{(X_3, X'_3)\}$

Expected Kernel

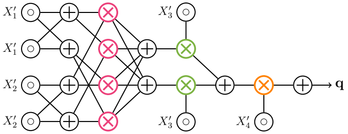
tractable computation via circuit operations

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

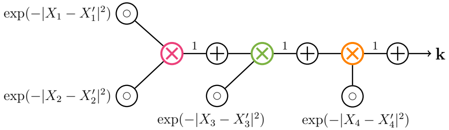
ii) PCs $\mathbf{p}$ and $\mathbf{q}$, and KC $\mathbf{k}$ are **compatible**



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



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



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

Expected Kernel

tractable computation via circuit operations

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

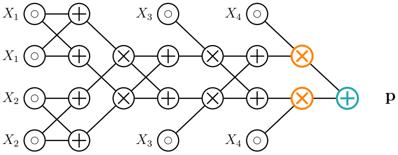
ii) PCs $\mathbf{p}$ and $\mathbf{q}$, and KC $\mathbf{k}$ are **compatible**

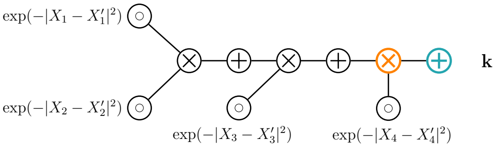
Then computing expected kernels can be done ***tractably*** by a forward pass

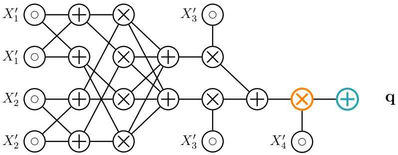
$$\Longrightarrow \quad \mathcal{O}(|\mathbf{p}||\mathbf{q}||\mathbf{k}|)$$

smooth + 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 $\mathbf{k}(\mathbf{X}, \mathbf{X}') = \sum_l w''_l \mathbf{k}_l(\mathbf{X}, \mathbf{X}')$:

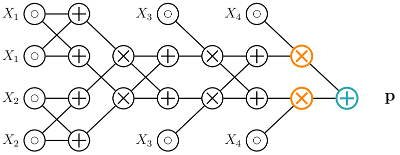


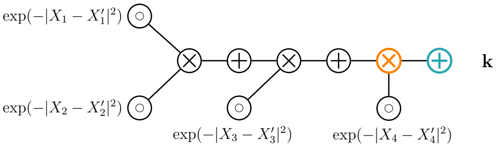


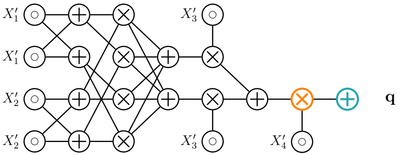


smooth + ***decomposable*** + ***compatible*** = ***tractable E[k]***

[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 $\mathbf{k}(\mathbf{X}, \mathbf{X}') = \sum_l w''_l \mathbf{k}_l(\mathbf{X}, \mathbf{X}')$:





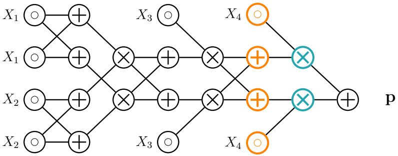


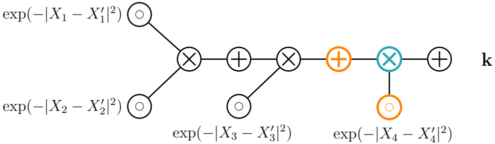
$$\mathbb{E}_{\mathbf{p},\mathbf{q}}[\mathbf{k}(\mathbf{x}, \mathbf{x}')] = \sum_{i,j,l} w_i w'_j w''_l \mathbb{E}_{\mathbf{p}_i,\mathbf{q}_j}[\mathbf{k}_l(\mathbf{x}, \mathbf{x}')]$$

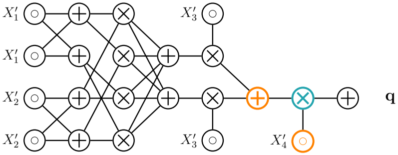
$\Longrightarrow$ *expectation is "pushed down" to inputs*

smooth + decomposable + compatible = tractable E[k]

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

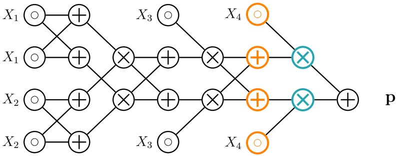


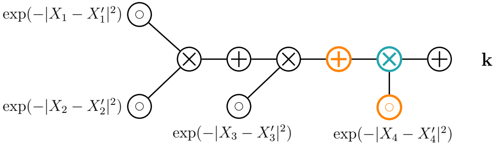


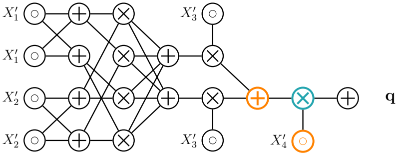


smooth + decomposable + compatible = tractable E[k]

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







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

$\Longrightarrow$ *expectation decomposes into easier ones*

smooth + **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!

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

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

$\Longrightarrow$ *support vector regression (SVR) with missing features*

Support vector regression with missing features

Given a regressor $f : \mathcal{X} \to \mathcal{Y}$, in the case when only features $\mathbf{X}_o = \mathbf{x}_o$ are *observed* and features $\mathbf{X}_m$ are *missing*, with $\mathbf{X} = (\mathbf{X}_o, \mathbf{X}_m)$, the expected prediction is

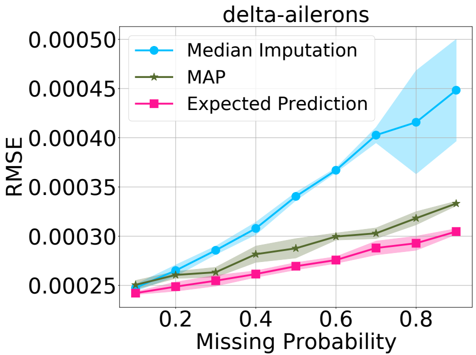
$$\mathbb{E}_{\mathbf{x}_m \sim \mathbf{p}(\mathbf{X}_m | \mathbf{x}_o)} \left[f\left(\mathbf{x}_o, \mathbf{x}_m\right) \right]$$

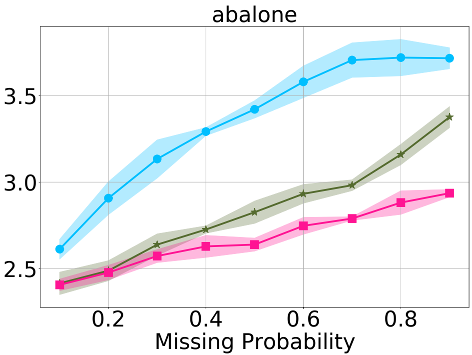
Support vector regression with missing features

For a kernel support vector regressor $f(\mathbf{x}) = \sum_{i=1}^{m} w_i \mathbf{k}(\mathbf{x}_i, \mathbf{x}) + b$, in the case when only features $\mathbf{X}_o = \mathbf{x}_o$ are *observed* and features $\mathbf{X}_m$ are *missing*, with $\mathbf{X} = (\mathbf{X}_o, \mathbf{X}_m)$, the expected prediction is

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

Support vector regression with missing features





⟹ *Expected prediction improves over the baselines*

Applications

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

$\Rightarrow$ *What about intractable models?*

Conclusion

Takeaways

#1: you can be both tractable and expressive

#2: circuits are a foundation for tractable inference over 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

References I

- Chow, C and C Liu (1968). "Approximating discrete probability distributions with dependence trees". In: *IEEE Transactions on Information Theory* 14.3, pp. 462–467.
- Rabiner, Lawrence and Biinghwang Juang (1986). "An introduction to hidden Markov models". In: *ieee assp magazine* 3.1, pp. 4–16.
- Bach, Francis R. and Michael I. Jordan (2001). "Thin Junction Trees". In: *Advances in Neural Information Processing Systems 14*. MIT Press, pp. 569–576.
- Darwiche, Adnan and Pierre Marquis (2002). "A knowledge compilation map". In: *Journal of Artificial Intelligence Research* 17, pp. 229–264.
- Jaeger, Manfred (2004). "Probabilistic decision graphs—combining verification and AI techniques for probabilistic inference". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 12.supp01, pp. 19–42.
- Kisa, Doga, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche (July 2014). "Probabilistic sentential decision diagrams". In: *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. Vienna, Austria. URL: `http://starai.cs.ucla.edu/papers/KisaKR14.pdf`.
- Choi, YooJung, Antonio Vergari, and Guy Van den Broeck (2020). "Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Modeling". In: