Probabilistic Circuits That Know What They Don't Know





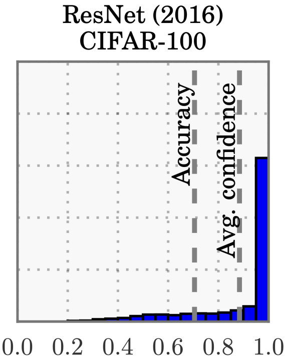
Fabrizio Ventola[*1] Steven Braun[*1] Zhongjie Yu[1] Martin Mundt[1,2] Kristian Kersting[1,2,3,4]

[1]Department of Computer Science, TU Darmstadt
[2]Hessian Center for AI (hessian.AI)
[3]German Research Center for Artificial Intelligence (DFKI)
[4]Centre for Cognitive Science, TU Darmstadt

* indicates equal contribution

Motivation

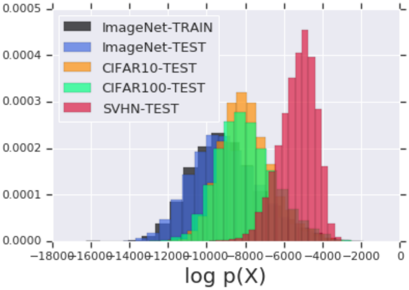
Many ML models are **overconfident** — often assign high probabilities to (wrong) predictions, even for unseen categories.



[Amodei et al., arXiv 2016; Guo et al., ICML 2017; Boult et al., AAAI 2019; Hendrycks et al., ICLR 2019]

Motivation

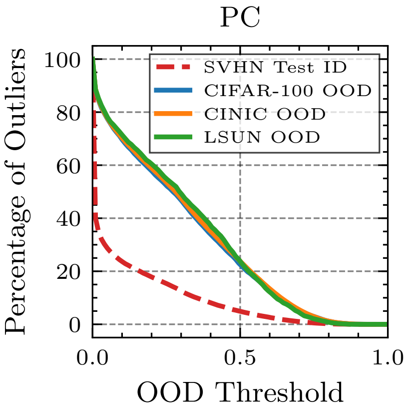
- The issue impacts even major deep models like VAEs and normalizing flows.
  [Nalisnick et al., ICLR 2019]



- **Probabilistic circuits (PCs)** are assumed to overcome this — casted as well-calibrated models.
  [Peharz et al., UAI 2020; Peharz et al., ICML 2020; Choi et al., UCLA 2020]

Our Contributions (1/2)

Show that PCs suffer from overconfidence and struggle to distinguish **in-distribution from out-of-distribution data** in discriminative setting.



Our Contributions (2/2)

- Introduce **Tractable Dropout Inference (TDI)** — provides tractable model uncertainty estimation.
- Derive a **sampling-free analytical solution for Monte Carlo dropout (MCD)**, a Bayesian method for model uncertainty in neural networks (NNs).
- Demonstrate **TDI's robustness** against distribution shifts and out-of-distribution data in three key scenarios.

Probabilistic Circuits

- We focus on sum-product networks (SPNs), a prominent type of PCs.
- SPNs are known for inference capabilities and representational power.
- Like NNs, SPNs are deep graphs but they explicitly encode a normalized probability distribution.
- SPNs can generate new samples and calculate various exact, tractable probabilistic queries, even with partial evidence.

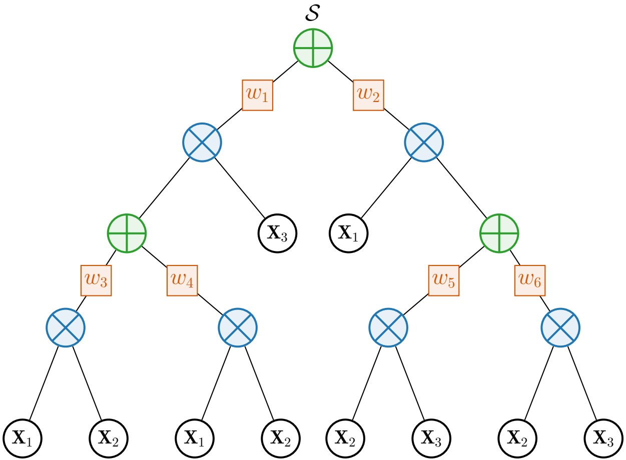
[Poon et al., UAI 2011; Delalleau et al., NeurIPS 2011; Peharz et al., AISTATS 2015]

Probabilistic Circuits

Clear probabilistic semantics!

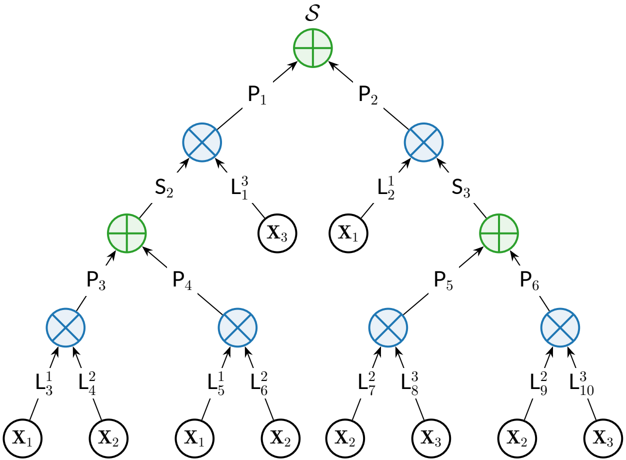
$\oplus$ $\quad \mathsf{S} = \sum_i w_i \mathsf{N}_i$

$\otimes$ $\quad \mathsf{P} = \prod_i \mathsf{N}_i$



Probabilistic Circuits

- Instance LL via forward pass
- $\mathcal{S}(\boldsymbol{x}) = p_{\boldsymbol{X}}(\boldsymbol{X} = \boldsymbol{x})$
- MAP & MPE need top-down pass
- Inference linear in the network size



Remark

- Probabilistic modeling targets uncertainties, but it's crucial to quantify model uncertainty or *epistemic uncertainty*.

- Quantification can tell us how confident the model is in its predictions, including class label distribution $p(Y|\boldsymbol{X} = \boldsymbol{x})$.

- Overconfident models can assign near 1 probabilities to out-of-distribution instances or unseen labels.

- Model uncertainty helps us know when the model "*does not know*", need to take predictions with caution.

[Kendall et al., NeurIPS, 2017; Hüllermeier et al., MLJ 2021]

Model Uncertainty Quantification in Deep Networks

- A common Bayesian way to quantify model uncertainty involves choosing the most likely parameters configuration $\boldsymbol{\theta}$ that represents the data $\mathcal{D}$.

$$\arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathcal{D}) \quad \text{where} \quad p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$$

- This approach is usually intractable as it involves integrating over parameters $\boldsymbol{\theta}$.

- Dropout is a popular method in NNs to prevent overfitting and improve generalization by randomly removing connections between layers.

[MacKay, Neural computation 1992; Srivastava et al., JMLR 2014]

Monte Carlo Dropout

- Gal et al. (ICML 2016) interpret dropout as a Bayesian approximation for model uncertainty, assuming Bernoulli distribution on weights.

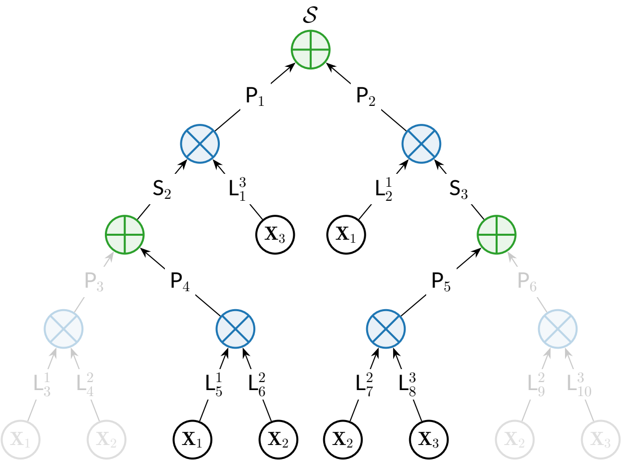
- The intractable integration over parameters is approximated by a tractable sum over n parameter sets: $\boldsymbol{\theta}_i \sim p\,(\boldsymbol{\theta} \,|\, \mathcal{D})$.

- With n predictions, first (mean) and second (variance) raw moments serve as prediction and model uncertainty.

[Gal et al., *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, ICML 2016]

MCD in PCs

- Place Bernoulli distribution at sum node weights
- Calculating two raw moments **requires n evals**!



Can we do better?

MCD needs n stochastic forward-passes of the model ... $\rightarrow$ **inefficient!**

Var$[\mathcal{S}]$ is induced by random instantiations of dropout at sum nodes.

Can we find a *closed-form* solution to obtain Var$[\mathcal{S}]$ in a **single** forward-pass?

Yes! $\Rightarrow$ **Tractable Dropout Inference (TDI)**

Tractable Dropout Inference (TDI)

Key idea:

1) View node output as function f of inputs and dropout RVs
2) Compute Var$[f] \rightarrow$ **decomposes** into input variances (and more)
3) **Propagate** Var$[f]$ from leaf nodes to root node

TDI only needs **a single** forward evaluation of a model ... $\rightarrow$ **efficient!**

TDI: Basic Idea

Consider sum nodes as linear combinations of their Bernoulli dropout RVs and inputs:

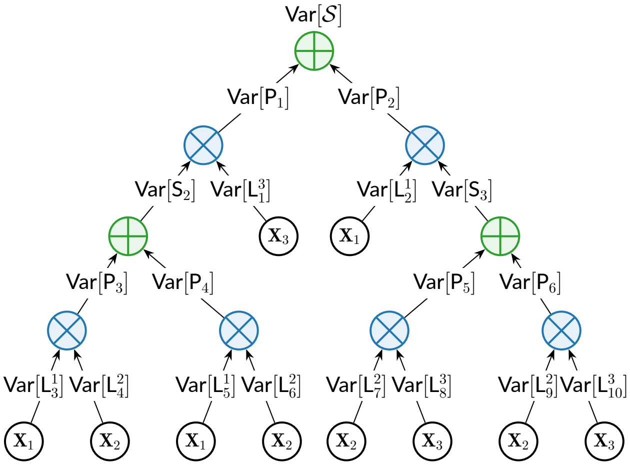
$$\mathsf{S} = \sum_i \overbrace{\delta_i}^{\text{dropout RVs}} \underbrace{w_i}_{\text{weights}} \overbrace{\mathsf{N}_i}^{\text{inputs}} ,$$

where $\delta_i \sim \mathsf{Bern}(q)$ and $p = 1 - q$ is the dropout probability.

How do we *tractably* compute the variance of a PC with dropout?

$\Rightarrow$ Find a closed-form expression of $\mathsf{Var}[\mathsf{S}]$ and perform **variance propagation**!

TDI: Variance Propagation



Closed-form Solutions

Recall: $\oplus$ $\mathsf{S} = \sum_i \delta_i w_i \mathsf{N}_i$ $\otimes$ $\mathsf{P} = \prod_i \mathsf{N}_i$

input variance | input expectation | input covariance

Variance

$\oplus$ $\mathrm{Var}[\mathsf{S}] = q \sum_i w_i^2 (\mathrm{Var}[\mathsf{N}_i] + p\, \mathbb{E}[\mathsf{N}_i]^2) + q^2 \sum_{i \neq j} w_i w_j \mathrm{Cov}[\mathsf{N}_i, \mathsf{N}_j]$

$\otimes$ $\mathrm{Var}[\mathsf{P}] = \prod_i (\mathrm{Var}[\mathsf{N}_i] + \mathbb{E}[\mathsf{N}_i]^2) - \prod_i \mathbb{E}[\mathsf{N}_i]^2$

Expectation

$\oplus$ $\mathbb{E}[\mathsf{S}] = q \sum_i w_i \mathbb{E}[\mathsf{N}_i]$

$\otimes$ $\mathbb{E}[\mathsf{P}] = \prod_i \mathbb{E}[\mathsf{N}_i]$

Covariance

$\oplus$ $\mathrm{Cov}[\mathsf{S}^A, \mathsf{S}^B] = q^2 \sum_i w_i^A \sum_j w_j^B \mathrm{Cov}\left[\mathsf{N}_i^A, \mathsf{N}_j^B\right]$

$\otimes$ $\mathrm{Cov}\left[\mathsf{P}^A, \mathsf{P}^B\right] = \mathbb{E}\left[\prod_i \mathsf{N}_i^A \prod_j \mathsf{N}_j^B\right] - \prod_i \mathbb{E}[\mathsf{N}_i^A] \prod_j \mathbb{E}\left[\mathsf{N}_j^B\right]$

↯ not decomposable in general ↯

Covariance: Three Solutions

↯ not decomposable in general ↯

$$\otimes \qquad \mathsf{Cov}\left[\mathsf{P}^A, \mathsf{P}^B\right] = \boxed{\mathbb{E}\left[\prod_i \mathsf{N}_i^A \prod_j \mathsf{N}_j^B\right]} - \prod_i \mathbb{E}\left[\mathsf{N}_i^A\right] \prod_j \mathbb{E}\left[\mathsf{N}_j^B\right]$$

We're not at a dead-end with covariance! We provide three solutions:

a) **Structural Knowledge**: LearnSPN, RAT-SPN, ... $\rightarrow$ add. independencies

b) **Covariance Bounds**: Cauchy-Schwarz inequality $\rightarrow$ lower/upper bound

$$\mathsf{Cov}[\mathsf{N}_i, \mathsf{N}_j] \in \left[-\sqrt{\mathsf{Var}[\mathsf{N}_i]\,\mathsf{Var}[\mathsf{N}_j]},\ +\sqrt{\mathsf{Var}[\mathsf{N}_i]\,\mathsf{Var}[\mathsf{N}_j]}\right]$$

c) **Copy-paste Solution**: Graph augmentation $\rightarrow$ enforce covariance to be zero

Leaf Nodes

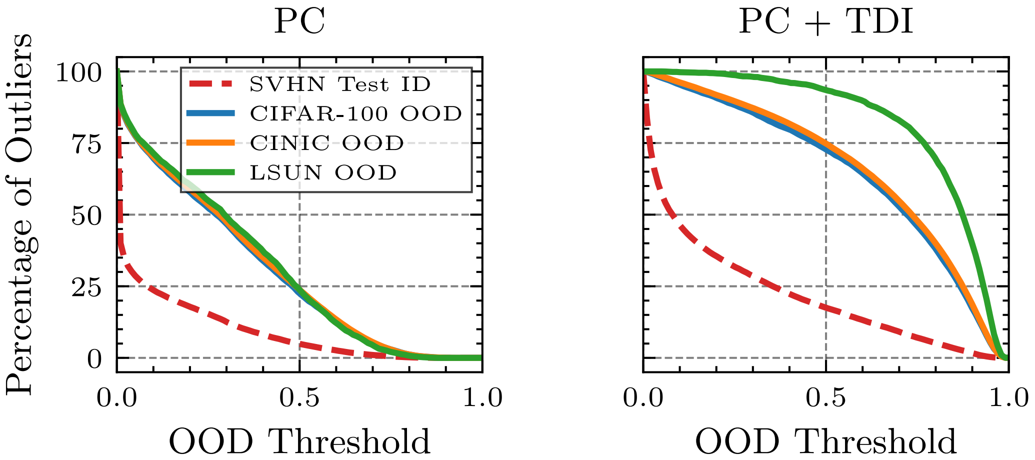
No dropout (in our current framework) $\rightarrow$ leaves become a point estimate!

$$\mathbb{E}[\mathsf{L}] = \mathsf{L}, \quad \mathsf{Var}[\mathsf{L}] = 0, \quad \mathsf{Cov}[\mathsf{L}_i, \mathsf{L}_j] = 0$$

BUT: This allows to include **prior knowledge** about *aleatoric* and *epistemic* uncertainty by setting $\mathsf{Var}[\mathsf{L}] > 0$ and $\mathsf{Cov}[\mathsf{L}_i, \mathsf{L}_j] \neq 0$ (future work).

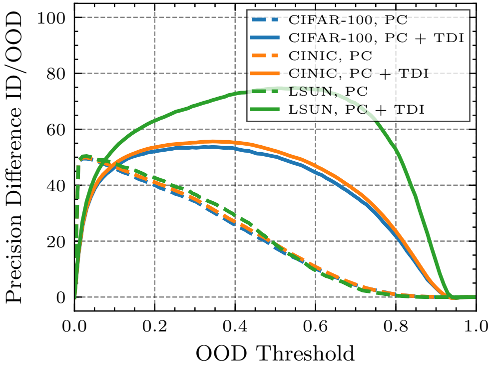
(Note: MCD does *not* allow that)

Experiments: Out-of-Distribution Detection



TDI remarkably improves OOD detection in PCs!

Experiments: Out-of-Distribution Detection

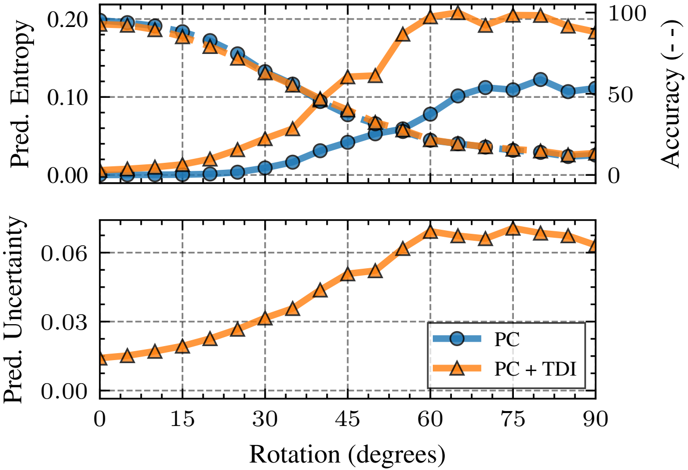


TDI allows PCs to adequately balance ID vs. OOD detection!

AUC (↑)	CIFAR	CINIC	LSUN
PC	29.3	29.9	30.3
PC + TDI	**64.6**	**66.1**	**81.8**
PC + MCD	68.5	70.0	84.9

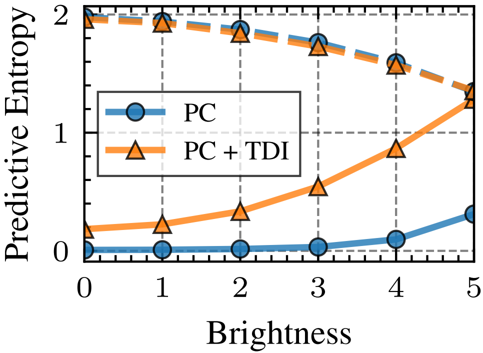
TDI is competitive with MCD and **only needs a single instead of $n = 100$ forward passes!**

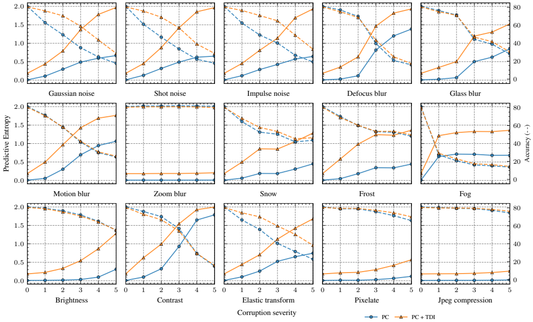
Experiments: Data Perturbations (Rotated MNIST)



TDI better captures the distribution shift while retaining predictive accuracy!

Experiments: Data Corruptions (MNIST)





TDI detects distribution shifts and is more robust in predictive accuracy against the corruption!

Future Work

- Influence of **dropout parameter** p?
- Can we generalize to **arbitrary PC structures**?
- **Density estimation**?
- Can we use **uncertainty during training**?

Conclusion

- Overconfidence in PCs makes ID and OOD data separation challenging
- **Tractable Dropout Inference**: MCD-inspired solution, offers straightforward single-pass uncertainty estimation for PCs which ...
  - enhances PC robustness and helps in detecting distribution shifts
  - removes the computational burden of MCD
  - paves the way to include uncertainty into PC training
  - allows for prior knowledge of epistemic or alleatoric uncertainty

Paper

Code

Backup Slides

Structural Knowledge

The easiest solution is when we know two product nodes P^A and P^B share no common ancestors. This implies $\mathsf{P}^A \perp\!\!\!\perp \mathsf{P}^B$ and thus $\mathsf{Cov}[\mathsf{P}^A, \mathsf{P}^B] = 0$.

This is always the case for tree-structured PCs!

For binary tree random and tensorized (RAT) structures, the covariance simplifies to:

$$\begin{aligned}\mathsf{Cov}\left[\mathsf{P}_{l,r}, \mathsf{P}_{l',r'}\right] = \quad & \mathsf{Cov}\left[\mathsf{S}^L_l, \mathsf{S}^L_{l'}\right] \mathbb{E}\left[\mathsf{S}^R_r\right] \mathbb{E}\left[\mathsf{S}^R_{r'}\right] \\ & + \mathsf{Cov}\left[\mathsf{S}^R_r, \mathsf{S}^R_{r'}\right] \mathbb{E}\left[\mathsf{S}^L_l\right] \mathbb{E}\left[\mathsf{S}^L_{l'}\right] \\ & + \mathsf{Cov}\left[\mathsf{S}^L_l, \mathsf{S}^L_{l'}\right] \mathsf{Cov}\left[\mathsf{S}^R_r, \mathsf{S}^R_{r'}\right] \ .\end{aligned}$$

In this case, the covariance of two product nodes only depends on the covariance of the input sum nodes of the same graph partition (L or R).

Covariance Bounds

If we don't have structural knowledge, we can still find lower and upper bounds for the covariance using the Cauchy-Schwarz inequality:

$$\begin{aligned}
&\mathsf{Cov}[\mathsf{N}_i, \mathsf{N}_j]^2 \leq \mathsf{Var}[\mathsf{N}_i]\,\mathsf{Var}[\mathsf{N}_j] \\
\Leftrightarrow\quad &\mathsf{Cov}[\mathsf{N}_i, \mathsf{N}_j] \;\in \Bigg[-\sqrt{\mathsf{Var}[\mathsf{N}_i]\,\mathsf{Var}[\mathsf{N}_j]}, \\
&\qquad\qquad\qquad +\sqrt{\mathsf{Var}[\mathsf{N}_i]\,\mathsf{Var}[\mathsf{N}_j]}\Bigg] \quad .
\end{aligned}$$

Copy-paste Solution

- We can augment the DAG to enforce zero covariance between two nodes, N_A and N_B, by treating their common input as two separate nodes.
- For each node N_C with paths $\mathsf{Path}_A := \mathsf{N}_A \to \mathsf{N}_C$ and $\mathsf{Path}_B := \mathsf{N}_B \to \mathsf{N}_C$, we "copy" N_C to an equivalent node $\mathsf{N}_{C'}$ and replace the original N_C in Path_B with $\mathsf{N}_{C'}$.
- This enforces a tree structure on the PC, making the covariance between two inputs of a node N zero.
- In practice, we don't need to modify the DAG. Instead, *we can simply ignore the covariance*.

Classification Uncertainty

In classification, we express class conditionals $p\left(\mathbf{x} \mid y_i\right) = \mathsf{S}_i$ with class priors $p\left(y_i\right) = c_i$, and find the posterior via Bayes' rule:

$$p\left(y_i \mid \mathbf{x}\right) = \frac{\mathsf{S}_i c_i}{\sum_j \mathsf{S}_j c_j} .$$

Here, the expectation and variance of the posterior are those of a random variable ratio, $\mathbb{E}\left[\frac{A}{B}\right]$ and $\mathrm{Var}\left[\frac{A}{B}\right]$, with $A = \mathsf{S}_i c_i$ and $B = \sum_j \mathsf{S}_j c_j$.
While this ratio isn't well-defined, we can approximate it using a second-order Taylor approximation:

$$\mathbb{E}\left[\frac{A}{B}\right] \approx \frac{\mathbb{E}[A]}{\mathbb{E}[B]} - \frac{\mathrm{Cov}[A, B]}{\left(\mathbb{E}[B]\right)^2} + \frac{\mathrm{Var}[B]\,\mathbb{E}[A]}{\left(\mathbb{E}[B]\right)^3}$$

$$\mathrm{Var}\left[\frac{A}{B}\right] \approx \frac{\mathbb{E}[A]^2}{\mathbb{E}[B]^2}\left[\frac{\mathrm{Var}[A]}{\mathbb{E}[A]^2} - 2\frac{\mathrm{Cov}[A, B]}{\mathbb{E}[A]\,\mathbb{E}[B]} + \frac{\mathrm{Var}[B]}{\mathbb{E}[B]^2}\right] .$$

Classification Uncertainty

We can simplify each component of the previous equations.
The expectations are:

$$\mathbb{E}[A] = \mathbb{E}[\mathsf{S}_i c_i] = \mathbb{E}[\mathsf{S}_i]\, c_i$$

$$\mathbb{E}[B] = \mathbb{E}\left[\sum_j \mathsf{S}_j c_j\right] = \sum_j \mathbb{E}[\mathsf{S}_j]\, c_j\ .$$

The variances become:

$$\mathsf{Var}[A] = \mathsf{Var}[\mathsf{S}_i]\, c_i^2$$

$$\mathsf{Var}[B] = \sum_j \mathsf{Var}[\mathsf{S}_j]\, c_j^2 + \sum_{j_1 \neq j_2} \mathsf{Cov}[\mathsf{S}_{j_1}, \mathsf{S}_{j_2}]\, c_{j_1} c_{j_2}\ .$$

Classification Uncertainty

- Covariance between a root node and sum of all root nodes can be deconstructed:

$$\mathsf{Cov}[A, B] = c_i \sum_j c_j \mathsf{Cov}[\mathsf{S}_i, \mathsf{S}_j] \quad ,$$

- Assumption of statistical independence between A and B is an approximation.
- This approximation has proven effective in practice.

[Seltman N., CMU 2018]

Tractability

- PCs and SPNs are tractable models with linear time queries.
- TDI formulations have at most quadratic space and time complexity.
- Sparse structures like trees allow for linear computational cost.
- In cases needing all covariance combinations, cost is locally quadratic.
- Using the "copy-paste" DAG augmentation, cost can be reduced to linear.
- Full bottom-up pass is tractable and parallelizable.