Deep Probabilistic Surrogate Networks for Universal Simulator Approximation

Andreas Munk, Adam Ścibior, Atılım Güneş Baydin, Andrew Stewart, Goran Fernlund, Anoush Poursartip, Frank Wood

Introduction

A probabilistic programming framework that:

1. Allows for surrogate modeling in higher-order probabilistic programming languages.
2. Speeds up simulations/program execution.

Model the distribution, p, over random variables $\boldsymbol{x}$ and their addresses $\boldsymbol{a}$ using a surrogate s based on neural networks ξ parametarized by θ,

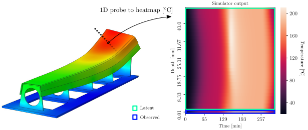
$$p(\boldsymbol{x},\boldsymbol{x}) = \prod_{t=1}^{T} p(a_t|x_{<a_t}, a_{<t})p(x_{a_t}|x_{<a_t}, a_{\leq t}),$$

$$s(\boldsymbol{x},\boldsymbol{a};\theta) = \prod_{t=1}^{T} s(x_{a_t}|\xi_{a_t}(x_{<a_t}, a_{\leq t}, \theta))s(a_t|\xi_{t-1}(x_{<a_t}, a_{<t}, \theta)).$$

- In particular we note that the address transitions, $p(a_t|x_{a_{1:a_{t-1}}}, a_{1:t-1})$, are deterministic.
- Surrogate modeling for higher-order programs requires modeling these address transitions.
- $s(\boldsymbol{x},\boldsymbol{a};\theta)$ is trained by minimizing the KL-divergence,

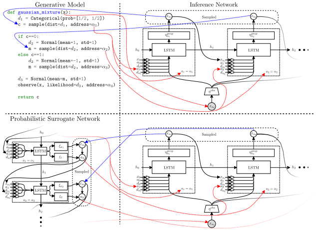
$$L(\theta) = \mathrm{KL}(p(\boldsymbol{x},\boldsymbol{a})||s(\boldsymbol{x},\boldsymbol{a};\theta)) = -\mathbb{E}_{p(\boldsymbol{x},\boldsymbol{a})}[\log s(\boldsymbol{x},\boldsymbol{a};\theta)] + \text{const.}$$

- We showcase the benefits of using our framework on the simulation of composite materials. The aim is to infer the internal unobservable state of the material during the curing process.



Probabilistic Surrogate Networks (PSNs)

- Provides for faster inference as PSNs are compatible with existing inference engines.
- Demonstrated in conjunction with inference compilation (IC) [1].
- Denote latent and observed variables $\boldsymbol{x}_{\text{lat}}$ and $\boldsymbol{x}_{\text{obs}}$ respectively.

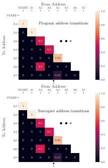


- x_{a_t} - sampled value for the variable at address a_t
- a_t - address of tth variable
- d_t - distribution type of tth variable in the program
- ξ_{a_t} - neural network whose output parameterizes the distribution of x_{a_t}
- ξ_t - neural network whose output parameterizes the distribution of a_{t+1}

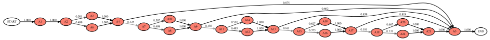
Modeling stochastic control flow

- We illustrate the PSNs ability to accurately model the address transitions associated with the stochastic control flow program.
- We choose $s(x_{a_t}|\xi_{a_t}(x_{<a_t}, a_{\leq t}, \theta))$ to be a categorical distribution.
- Only small deviation are found between the address transition probabilities in the program and in the surrogate.
- The deviations found happen with small probability.

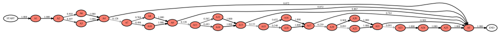
```
def control_flow_program(x):
    d1 = Beta(50, 7)
    θ = sample(dist=d1)
    μ = 0
    while True:
        d2 = Categorical(prob=[1/2, 1/2])
        b = sample(dist=d2)
        if b:
            d3 = Normal(mean=0, std=1/2)
            z = sample(dist=d3)
        else:
            d3 = Normal(mean=2, std=1/2)
            z = sample(dist=d3)
        μ += z
        d4 = Categorical(prob=[1-θ,θ])
        c = sample(dist=d4)
        if c:
            break
    d5 = Normal(mean=μ, std=1)
    observe(x, likelihood=d5)
    return θ
```



Program address transitions



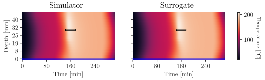
Surrogate address transitions

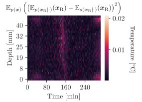


Process simulation of composite materials

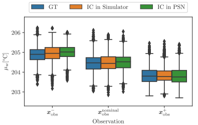
Using PSNs we are able to accurately model the joint distribution defined by the simulator and achieve running times that are magnitudes smaller than that of the original simulator. Using PSNs for inference tasks produces accurate posterior estimations many times faster than using the simulator.

- ~ 90 times faster simulation.





- Let $\mu_w(\boldsymbol{x}_{\text{lat}})$ be the empirical mean across the time window $w = [155, 165]$min at a fixed 30mm depth.
- Infer $\mathbb{E}_{p(\boldsymbol{x}_{\text{lat}}|\boldsymbol{x}_{\text{obs}})}[\mu_w(\boldsymbol{x}_{\text{lat}})]$.
- ~ 15 times faster inference.



$\boldsymbol{x}_R$ - denotes the output of the *RAVEN* [2] (R) simulator used. It is the temperature as function of depth and time.

References

[1] Le, T. A., Baydin, A. G., and Wood, F. (2017). Inference compilation and universal probabilistic programming. In Proceedings of the 20th In Conference on Artificial Intelligence and Statistics, volume 54 of Proceedings of Machine Learning Research, pages 1338–1348, Fort Lauderdale, FL, USA.

[2] Convergent Manufacturing Technologies. 2019. RAVEN Simulation Software. Technical report, Vancouver.

THE UNIVERSITY OF BRITISH COLUMBIA
UNIVERSITY OF OXFORD
CONVERGENT MANUFACTURING TECHNOLOGIES