Updates and No-ether Networks

Yogesh Verma
Doctoral Candidate
Aalto University

Updates

- Paper read:
  - Noether Networks (`https://arxiv.org/pdf/2112.03321.pdf`)
  - Interaction Networks for Learning about Objects, Relations and Physics
  - Physics-informed machine learning (Review)
  - Variational multiple shooting for Bayesian ODEs with Gaussian processes [reading]

Updates

- Paper read:
  - Noether Networks (`https://arxiv.org/pdf/2112.03321.pdf`)
  - Interaction Networks for Learning about Objects, Relations and Physics
  - Physics-informed machine learning (Review)
  - Variational multiple shooting for Bayesian ODEs with Gaussian processes [reading]
- Mathematical Formulation of Spectral Density over Graphs

No-ether Theorem

For every continuous symmetry property of a dynamical system, there is a corresponding quantity whose value is conserved in time.

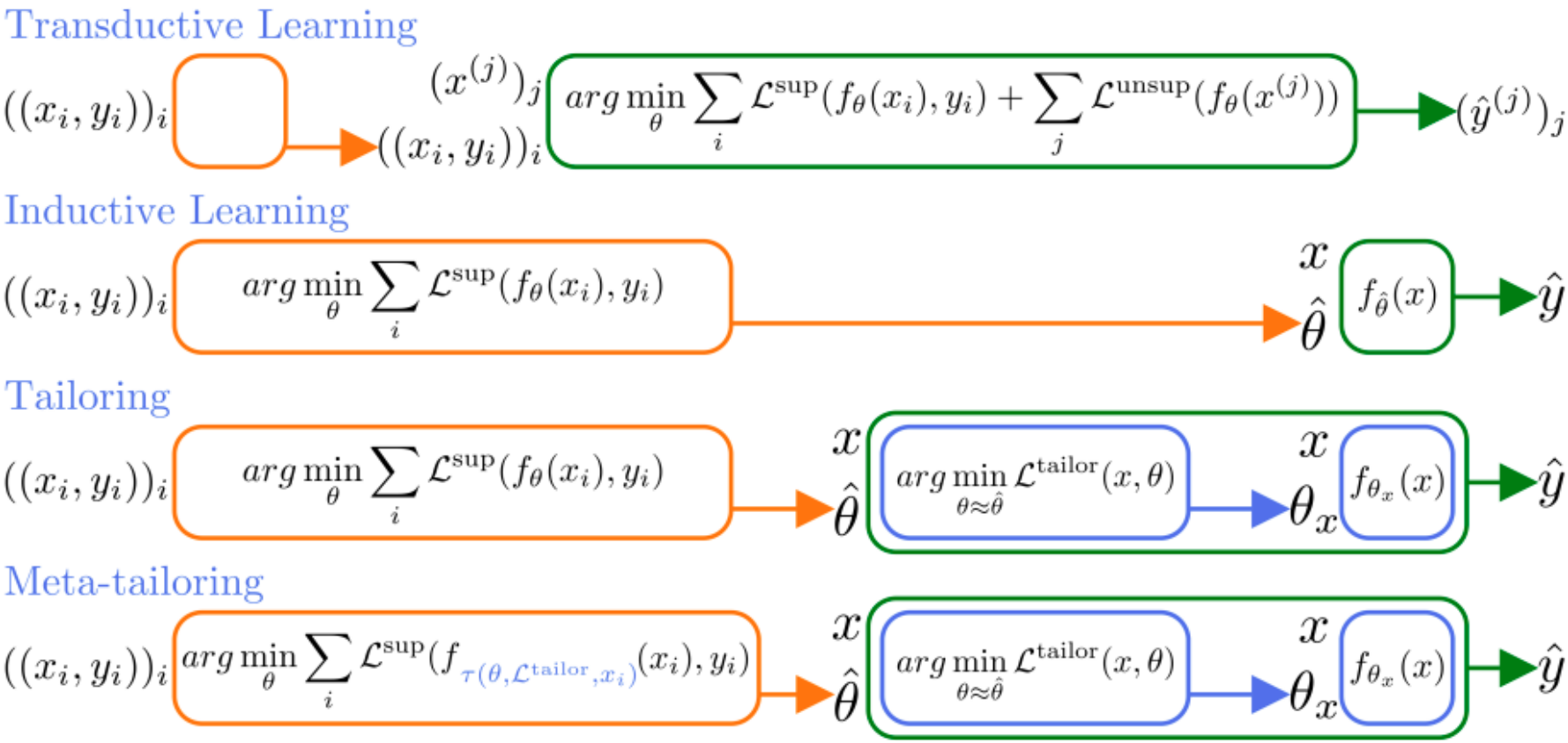
No-ether Theorem

For every continuous symmetry property of a dynamical system, there is a corresponding quantity whose value is conserved in time.

- Approximate conservation laws are a powerful paradigm for meta-learning useful inductive biases in sequential prediction problems.
- Approach involves meta-learning a parametric conservation loss function which is useful to the prediction task by using tailoring framework

Tailoring for meta learning

Encode inductive biases by fine-tuning neural networks with hand-designed unsupervised losses inside the prediction function.



No-ether Networks

- Prediction-time optimization to encourage outputs to follow conservation laws using the tailoring framework. Tailoring encodes inductive biases in the form of unsupervised losses optimized inside the prediction function

No-ether Networks

- Prediction-time optimization to encourage outputs to follow conservation laws using the tailoring framework. Tailoring encodes inductive biases in the form of unsupervised losses optimized inside the prediction function
- Why Tailoring?
  - Fine-tunes the model to each query to ensure that it minimizes the unsupervised loss for that query, whereas auxiliary losses optimise loss only for the training points, tailoring allows us to ensure conservation at test time

No-ether Networks

- Prediction-time optimization to encourage outputs to follow conservation laws using the tailoring framework. Tailoring encodes inductive biases in the form of unsupervised losses optimized inside the prediction function
- Why Tailoring?
  - Fine-tunes the model to each query to ensure that it minimizes the unsupervised loss for that query, whereas auxiliary losses optimise loss only for the training points, tailoring allows us to ensure conservation at test time
  - Easier to meta-learn whereas auxiliary losses are pooled over all examples and training epochs and their effect is only known at validation/test time

No-ether Networks

- Prediction-time optimization to encourage outputs to follow conservation laws using the tailoring framework. Tailoring encodes inductive biases in the form of unsupervised losses optimized inside the prediction function
- Why Tailoring?
  - Fine-tunes the model to each query to ensure that it minimizes the unsupervised loss for that query, whereas auxiliary losses optimise loss only for the training points, tailoring allows us to ensure conservation at test time
  - Easier to meta-learn whereas auxiliary losses are pooled over all examples and training epochs and their effect is only known at validation/test time
- Search for useful conservation laws, whose (approximate) enforcing brings us closer to the true data manifold for the current sequence

Meta Learning Neural Loss function

- Base predictor $f_\theta : x_t \rightarrow x_{t+1}$

Meta Learning Neural Loss function

- Base predictor $f_\theta : x_t \rightarrow x_{t+1}$
- Meta-learned tailoring loss, parameterized as the conservation of a neural embedding $g_\phi : x_t \rightarrow \mathbb{R}^k$ as

$$\mathcal{L}_{Noether}(x_0, x_{1:T}, g_\phi) = \sum_{t=1}^{T} |g_\phi(x_0) - g_\phi(x_t)|^2 \approx \sum_{t=1}^{T} |g_\phi(x_{t-1}) - g_\phi(x_t)|^2$$

Meta Learning Neural Loss function

- Base predictor $f_\theta : x_t \rightarrow x_{t+1}$
- Meta-learned tailoring loss, parameterized as the conservation of a neural embedding $g_\phi : x_t \rightarrow \mathbb{R}^k$ as

$$\mathcal{L}_{Noether}(x_0, x_{1:T}, g_\phi) = \sum_{t=1}^{T} |g_\phi(x_0) - g_\phi(x_t)|^2 \approx \sum_{t=1}^{T} |g_\phi(x_{t-1}) - g_\phi(x_t)|^2$$

- Prediction after tuning the tailored weights by $\mathcal{L}_{Noether}$ as $x_t = f_{\theta(x_0,\phi)}(x_{t-1})$

Described Algorithm

Algorithm 1 Prediction and training procedures for Noether Networks with neural conservation loss

Given: predictive model class f; embedding model class g; prediction horizon T; training dist. $\mathcal{D}_{\text{train}}$; batch size N; learning rates $\lambda_{\text{in}}, \lambda_{\text{out}}, \lambda_{\text{emb}}$; task loss $\mathcal{L}_{\text{task}}$; Noether loss $\mathcal{L}_{\text{Noether}}$

procedure PREDICTSEQUENCE($x_0; \theta, \phi$)
  $\tilde{x}_0, \hat{x}_0 \leftarrow x_0, x_0$
  $\tilde{x}_t \leftarrow f_\theta(\tilde{x}_{t-1})\ \forall t \in \{1, \dots, T\}$ ▷ Initial predictions
  $\theta(x_0; \phi) \leftarrow \theta - \lambda_{\text{in}} \nabla_\theta \mathcal{L}_{\text{Noether}}(x_0, \tilde{x}_{1:T}; g_\phi)$ ▷ Inner step with Noether loss
  $\hat{x}_t \leftarrow f_{\theta(x_0;\phi)}(\hat{x}_{t-1})\ \forall t \in \{1, \dots, T\}$ ▷ Final prediction with tailored weights
  return $\hat{x}_{1:T}$

procedure TRAIN
  $\phi \leftarrow$ randomly initialized weights ▷ Initialize weights for Noether embedding g
  $\theta \leftarrow$ randomly initialized weights ▷ Initialize weights for predictive model f
  while not done **do**
    Sample batch $x^{(0)}_{0:T}, \dots, x^{(N)}_{0:T} \sim \mathcal{D}_{\text{train}}$
    for $0 \leq n \leq N$ **do**
      $\hat{x}^{(n)}_{1:T} \leftarrow$ PREDICTSEQUENCE($x^{(n)}_0; \theta, \phi$)
    $\phi \leftarrow \phi - \lambda_{\text{emb}} \nabla_\phi \sum_{n=0}^{N} \mathcal{L}_{\text{task}}(\hat{x}^{(n)}_{1:T}, x^{(n)}_{1:T})$ ▷ Outer step with task loss for embedding
    $\theta \leftarrow \theta - \lambda_{\text{out}} \nabla_\theta \sum_{n=0}^{N} \mathcal{L}_{\text{task}}(\hat{x}^{(n)}_{1:T}, x^{(n)}_{1:T})$ ▷ Outer step for predictive model
  return ϕ, θ

Experiments and Results

Can Noether Networks recover known conservation laws in scientific data?

- Generate all valid formulas (symbolic,treated as conserved quantity), removing operations between incompatible physical units and formulas equivalent up to commutativity by DSL: AI-Feynman

Experiments and Results

Can Noether Networks recover known conservation laws in scientific data?

- Generate all valid formulas (symbolic,treated as conserved quantity), removing operations between incompatible physical units and formulas equivalent up to commutativity by DSL: AI-Feynman
- For each potentially conserved quantity, we try it as meta-tailoring loss: evaluate the fine-tuned model on long-term predictions, keeping the expression with the best MSE loss

Table 1: Recovering $\mathcal{H}$ for the ideal pendulum.

Method	Description	RMSE
Vanilla MLP	N/A	0.0563
Noether Nets	$p^2 - 2.99\cos(q)$	0.0423
True $\mathcal{H}$ [oracle]	$p^2 - 3.00\cos(q)$	0.0422

Table 2: Recovering $\mathcal{H}$ for the ideal spring.

Method	Description	RMSE
Vanilla MLP	N/A	.0174
Noether Nets	$q^2 + 1.002\,p^2$	.0165
True $\mathcal{H}$ [oracle]	$q^2 + 1.000\,p^2$	.0166

Experiments and Results

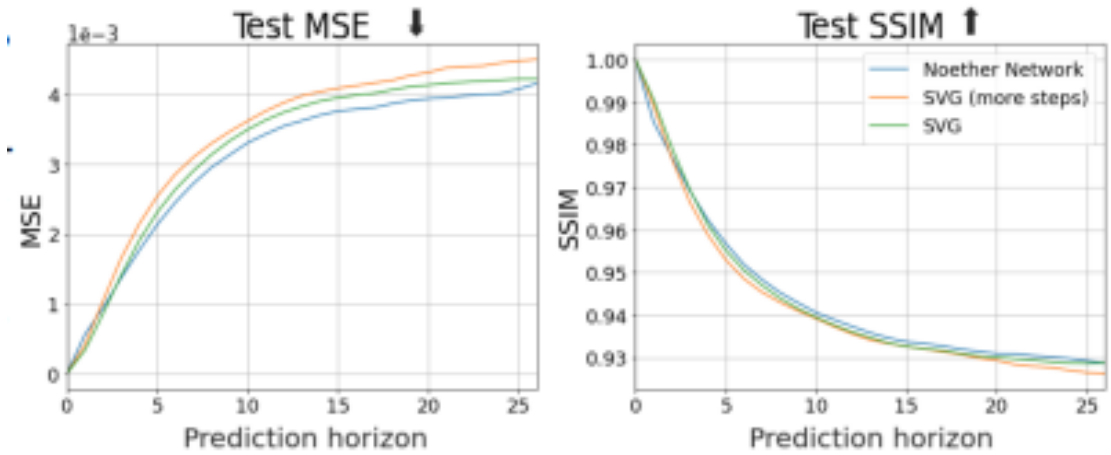
Are Noether Networks useful in settings with controlled dynamics?

- Add a Noether conservation loss to the stochastic video generation (SVG) model but (1) remove the prior sampling and (2) concatenate each action to the corresponding frame encoder output and feed the result to the LSTM

Experiments and Results

Are Noether Networks useful in settings with controlled dynamics?

- Add a Noether conservation loss to the stochastic video generation (SVG) model but (1) remove the prior sampling and (2) concatenate each action to the corresponding frame encoder output and feed the result to the LSTM
- On pixel pendulum with controls by videos of episodes in OpenAI Gym's pendulum swing up environment; Each model receives four history frames and a sequence of 26 policy actions starting from the current timestep, and must predict 26 future frames.



Experiments and Results

Can Noether Networks parameterize useful conserved quantities from raw pixel information?

- Physics 101 dataset containing videos of various real-world objects sliding down an incline and colliding with a second object.

Experiments and Results

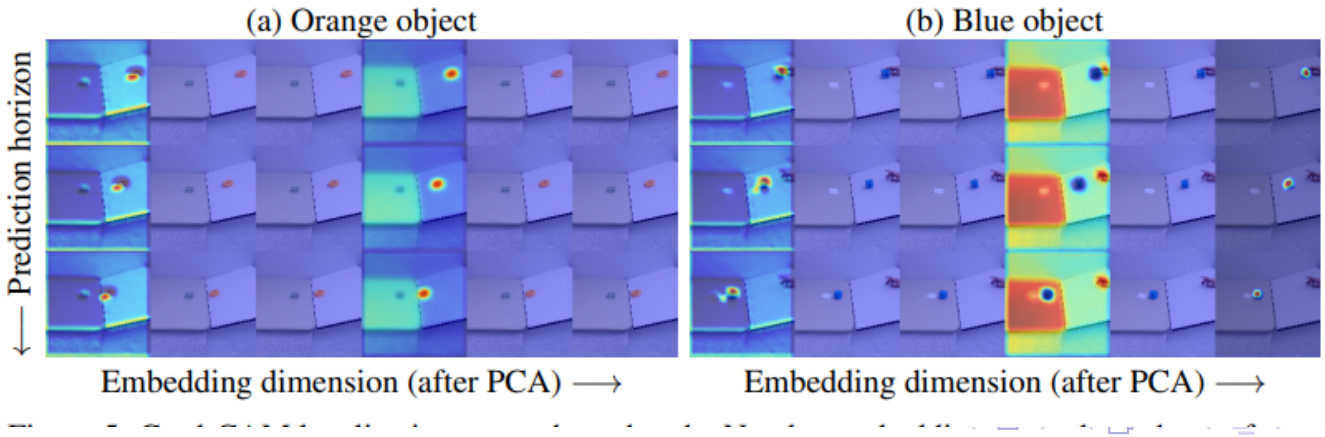
Can Noether Networks parameterize useful conserved quantities from raw pixel information?

- Physics 101 dataset containing videos of various real-world objects sliding down an incline and colliding with a second object.
- g_ϕ as 2D CNN's followed by a fully-connected projection layer (output as 64 dim embedding) and Gradient-weighted Class Activation Mapping (Grad-CAM) to compute importance maps for sequences in the test set.

Experiments and Results

Can Noether Networks parameterize useful conserved quantities from raw pixel information?

- Physics 101 dataset containing videos of various real-world objects sliding down an incline and colliding with a second object.
- g_ϕ as 2D CNN's followed by a fully-connected projection layer (output as 64 dim embedding) and Gradient-weighted Class Activation Mapping (Grad-CAM) to compute importance maps for sequences in the test set.
- Perform principal component analysis (PCA) before Grad-CAM to reduce the dimensionality of the embedding



Connection to our work!

- Molecules indeed have certain conserved quantities over time

Connection to our work!

- Molecules indeed have certain conserved quantities over time
- One can define a neural embedding (unsupervised loss) over a molecule or a portion of molecule, such that

Connection to our work!

- Molecules indeed have certain conserved quantities over time
- One can define a neural embedding (unsupervised loss) over a molecule or a portion of molecule, such that
  - We can embed the roto-translation invariance by using momentum terms leading to an invariant graph

Connection to our work!

- Molecules indeed have certain conserved quantities over time
- One can define a neural embedding (unsupervised loss) over a molecule or a portion of molecule, such that
  - We can embed the roto-translation invariance by using momentum terms leading to an invariant graph
  - Sub-graph aggregation or new atoms/rings adding to molecules in chemical reactions by following certain conservation's by neural embedding

Connection to our work!

- Molecules indeed have certain conserved quantities over time
- One can define a neural embedding (unsupervised loss) over a molecule or a portion of molecule, such that
  - We can embed the roto-translation invariance by using momentum terms leading to an invariant graph
  - Sub-graph aggregation or new atoms/rings adding to molecules in chemical reactions by following certain conservation's by neural embedding
  - Within a graph/molecule, we can have neural embedding over relational inference over interactions and regularizes the interactions to be physically probable!; can have application in RL over decision/action strategy molecule structure over unsupervised loss.

Mathematical Formulation

Given a graph $\mathcal{G} = (V, E)$ where V are vertices (nodes) and E are edges of the graph.
There exist a spectral density $\mathcal{S}(\omega_i, \theta_i)$ (parameterized by ω_i, θ_i) over:

- Over each node
- Over Subgraph-structure (e.g. Rings etc.)

Mathematical Formulation

Given a graph $\mathcal{G} = (V, E)$ where V are vertices (nodes) and E are edges of the graph.
There exist a spectral density $\mathcal{S}(\omega_i, \theta_i)$ (parameterized by ω_i, θ_i) over:

- Over each node
- Over Subgraph-structure (e.g. Rings etc.)

The total spectral density over graph can be written as:

$$\mathcal{S}(\Omega, \Theta) = \prod \mathcal{S}(\omega_i, \theta_i)$$

where $\Theta = \{\theta_i\}$ and $\Omega = \{\omega_i\}$. The parameters θ_i may also depend on the individual features of nodes.

Kernel Defining

Each node/subgraph having a spectral density $\mathcal{S}(\omega_i, \theta_i)$; one can define kernel as "mixture of experts"

$$\mathcal{K}_i(\psi_i) = \int_{-\infty}^{\infty} \mathcal{S}(\omega_i, \theta_i) e^{2\pi\omega\psi_i} d\omega + \textit{Interaction term}$$

The first term is a local contribution term defined by each node/subgraph features.

Kernel Defining

Each node/subgraph having a spectral density $\mathcal{S}(\omega_i, \theta_i)$; one can define kernel as "mixture of experts"

$$\mathcal{K}_i(\psi_i) = \int_{-\infty}^{\infty} \mathcal{S}(\omega_i, \theta_i) e^{2\pi\omega\psi_i} d\omega + \textit{Interaction term}$$

The first term is a local contribution term defined by each node/subgraph features.
There is an open choice in choosing the interaction term. One option is to have a whole graph contribution towards each node/subgraph kernels.

$$\textit{Interaction term} = \int_{-\infty}^{\infty} \mathcal{S}(\Omega, \Theta) e^{2\pi\Omega\psi_i} d\Omega$$

Examples of choosing (I)

Let us consider $\mathcal{S}(\omega_i, \theta_i)$ defined by a mixture of gaussians (SM kernel) as $\{\alpha_i \mathcal{N}(\mu_i, \sigma_i^2)\}$ given as

$$\mathcal{S}(\omega_i, \theta_i = (\mu_i, \sigma_i^2, \alpha_i)) = \sum_{k=1}^{Q} \alpha_i \mathcal{N}(\mu_i, \sigma_i^2)$$

$$\mathcal{K}_i(\psi_i) = \sum_{k=1}^{Q} \alpha_i e^{-2\pi\sigma_i^2\psi_i^2} cos(2\pi\psi_i\mu_i)$$

One can also calculate the interaction term given by whole graph.

Examples of choosing (II)

One can choose to represent $\mathcal{S}(\omega_i, \theta_i)$ by a *Quadric Surface*

Quadric Surface

- Simplest surface, represented by a quadratic equation

$$(\mathbf{r}A).\mathbf{r} + \mathbf{b}.\mathbf{r} + c = 0$$

where A is a symmetric matrix , $\mathbf{r} = (x, y, z)$ and b$\in \mathbb{R}^3$ vector

$$A = \begin{bmatrix} a_1 & a_4 & a_6 \\ a_4 & a_2 & a_5 \\ a_6 & a_5 & a_3 \end{bmatrix}$$

$$b = (b_1, b_2, b_3)$$

Quadric Surface

- Simplest surface, represented by a quadratic equation

$$(\mathbf{r}A).\mathbf{r} + \mathbf{b}.\mathbf{r} + c = 0$$

where A is a symmetric matrix , $\mathbf{r} = (x, y, z)$ and b$\in \mathbb{R}^3$ vector

$$A = \begin{bmatrix} a_1 & a_4 & a_6 \\ a_4 & a_2 & a_5 \\ a_6 & a_5 & a_3 \end{bmatrix}$$

$$b = (b_1, b_2, b_3)$$

Equation can be written as

$$a_1x^2 + a_2y^2 + a_3z^2 + 2a_4xy + 2a_5yz + 2a_6xz + b_1x + b_2y + b_3z + c = 0$$

By having a constraint over parameters, it can represent surfaces like ellipsoid, hyperbloid, quadric-cone and many more 2-D curves.

Examples of choosing (II)

One can choose to represent $\mathcal{S}(\omega_i, \theta_i)$ by a *Quadric Surface* and can learn or constraint over the parameters to represent various densities in 2-D and 3-D space.

Evolution

- The evolution of total spectral density can be characterized by:

$$\frac{d\mathcal{S}(\Omega, \Theta, t+1)}{dt} = \mathcal{F}(\mathcal{S}(\Omega, \Theta, t), G, \Theta)$$

Or

$$\mathcal{S}(\Omega, \Theta, t+1) = \mathcal{S}(\Omega, \Theta, t) + \mathcal{F}(\mathcal{S}(\Omega, \Theta, t), G, \Theta)$$

Choosing **f**

We have defined $\mathcal{N}(\mathcal{V})$ or $\mathcal{N}(g_i)$ (g_i is subgraph of $\mathcal{G}$) different G.P. over the graph. One way of sampling can be done via LMC (Linear Model of Coregionalisation) or can build deep gaussian process having recurrent function compositions.

Choosing **f**

We have defined $\mathcal{N}(\mathcal{V})$ or $\mathcal{N}(g_i)$ (g_i is subgraph of $\mathcal{G}$) different G.P. over the graph. One way of sampling can be done via LMC (Linear Model of Coregionalisation) or can build deep gaussian process having recurrent function compositions.
Sample S_q functions from Q separate process/nodes (each node has its defined GP with $K(\psi_i))$ $u_q^{(s)} \sim \mathcal{GP}(0, K(\psi_i))$.
It can be defined as a weighted sum,

$$f(x) = \sum_{q=1}^{Q} \sum_{s=1}^{S} a_q^{(s)} u_q^{(s)}(x)$$

where
$f(x) = [f_1(x), f_2(x), ..., f_P(x)]^T$, $a_q^{(s)} = [a_{q,1}^{(s)}, a_{q,2}^{(s)}, ..., a_{q,P}^{(s)}]^T$

Choosing $\mathbf{f}$ (LMC)

- One can define the number of neighbouring nodes (process) from which one can have a weighted contribution such that it captures the interactions with neighbours as well.

Choosing $\mathbf{f}$ (LMC)

- One can define the number of neighbouring nodes (process) from which one can have a weighted contribution such that it captures the interactions with neighbours as well.

For 2 processes:

$$\begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \sum_{q=1}^{Q} B_q \bigotimes K_q)$$

where $B_q = \sum_{s=1}^{S} a_q^{(s)} a_q^{(s)^T}$

Choosing **f** (LMC)

- One can define the number of neighbouring nodes (process) from which one can have a weighted contribution such that it captures the interactions with neighbours as well.

For 2 processes:

$$\begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \sum_{q=1}^{Q} B_q \bigotimes K_q)$$

where $B_q = \sum_{s=1}^{S} a_q^{(s)} a_q^{(s)^T}$

Another way: Function compositions using Deep Gaussian Processes

THANK YOU