

Compositional Model Design: High-Dimensional Multi-Output Regression

Wessel Bruinsma

University of Cambridge, CBL

29 October 2021

Model $p(x)$ for $x: \mathbb{R} \rightarrow \mathbb{R}^m$:

- ✓ Existing learning and inference routines.
- ✗ Feasible for moderate number of outputs (~ 10 – 100).

✗ computational complexity

✗ model complexity

Problem: Data may be very high-dimensional (~ 1000 – 10000).

- Example: daily temperature measurements around Europe.
- Often fewer ($\ll 1000$) underlying “mechanisms”.

Goal: Design a procedure that “wraps” $p(x)$ to scale to many outputs.

Compose $p(x)$ with a likelihood $p(y | x)$ to scale to many outputs:

$$\text{high-dim. model} \longleftarrow p(y) = \int p(y | x)p(x) dx.$$

maps into high-dim. space available model

Principled probabilistic approach: hope for well-calibrated uncertainty!

Desiderata:

- ✓ Learning and inference for $p(y)$ use existing routines for $p(x)$.
- ✓ Favourable scaling in number of outputs.
- ✓ Ability to deal with missing data.

```
from dream import DeepGP, MultiOutput

dgp = DeepGP(num_outputs=10) #  $p(x)$ 
model = MultiOutput(dgp, num_outputs=10000) #  $p(y)$ 

model.fit(x, y) # Uses `dgp.fit`.

pred = model.predict(x) # Uses `dgp.predict`.
```

Data: $y(t) \in \mathbb{R}^p$. **Model:** $x(t) \in \mathbb{R}^m$.

- Data is high-dimensional: $p \gg m$.
- Likelihood model $p(y | x)$ needs to transform $x(t)$ to $y(t)$.

Linear model:

$$\begin{aligned} \underbrace{y(t)}_{\mathbb{R}^p} &= \underbrace{h_1}_{\mathbb{R}^p} \underbrace{x_1(t)}_{\mathbb{R}} + \cdots + \underbrace{h_m}_{\mathbb{R}^p} \underbrace{x_m(t)}_{\mathbb{R}} + \underbrace{\varepsilon(t)}_{\mathbb{R}^p} \\ &= \underbrace{H}_{\mathbb{R}^{p \times m}} \underbrace{x(t)}_{\mathbb{R}^m} + \varepsilon(t). \end{aligned}$$

- Data lives around m -dimensional linear subspace $\text{col}(H) \subseteq \mathbb{R}^p$.

Full generative model:

$$\begin{aligned}x &\sim p(x), && \text{(latent model)} \\f(t) | H, x(t) &= Hx(t), && \text{(mixing mechanism)} \\y(t) | f(t) &\sim \mathcal{N}(f(t), \Sigma), && \text{(observation model)}\end{aligned}$$

x : “latent processes”,

H : “basis” or “mixing matrix”.

- ✓ Scales $p(x)$ to many outputs.
- ? Learning and inference for $p(y)$ use existing routines for $p(x)$.
- ? Favourable scaling in number of outputs.
- ? Deal with missing data.

We consider $x \sim \text{GPAR}$:

$$\begin{aligned}x_1(t) &= f_1(t), & f_1 &\sim \mathcal{GP}(0, k_1), \\x_2(t) &= f_2(t, x_1(t)), & f_2 &\sim \mathcal{GP}(0, k_2), \\x_3(t) &= f_3(t, x_1(t), x_2(t)), & f_3 &\sim \mathcal{GP}(0, k_3).\end{aligned}$$

- ✓ Captures nonlinear dependencies between outputs.
- ✓ Learning and inference exact and closed form.
- ✓ Feasible for moderate number of outputs.
- ✓ Depends on ordering of outputs. (Implicit in H !)
- ✗ Cannot be further composed with Gaussian likelihood.

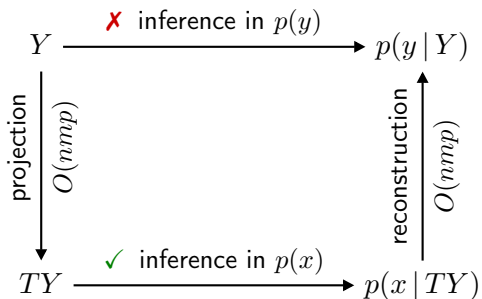
Inference in $p(y)$ in Terms of Inference in
 $p(x)$

Projection of the data:

$$T = \overbrace{y}^{\mathbb{R}^p} \mapsto \underbrace{\overbrace{(H^\top \Sigma^{-1} H)^{-1} H^\top \Sigma^{-1} y}^{\mathbb{R}^m}}_{\text{"observation for } p(x)\text{"}}$$

Then use inference for $p(x)$!**Proposition:**

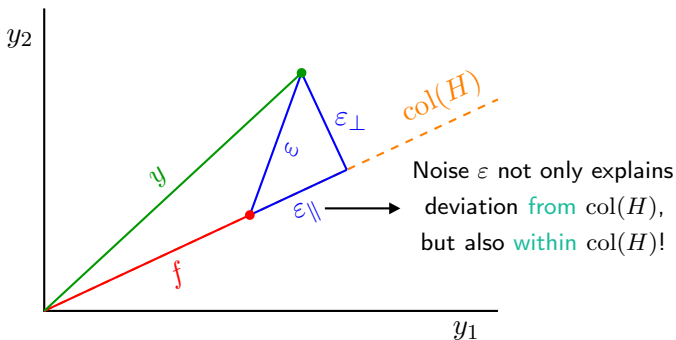
- ① $p(f | \overbrace{Y}^{\mathbb{R}^{p \times n}}) = p(f | \overbrace{TY}^{\mathbb{R}^{m \times n}})$ with $Ty_i | x_i \sim \mathcal{N}(x_i, \Sigma_T)$. “projected noise”
↑
- ② $p(Y) = \left[\prod_{i=1}^n \frac{\mathcal{N}(y_i | 0, \Sigma)}{\mathcal{N}(Ty_i | 0, \Sigma_T)} \right] \int p(x) \prod_{i=1}^n \mathcal{N}(Ty_i | x_i, \Sigma_T) dx.$



$$\begin{aligned}
 & \text{likelihood under } p(x) \text{ with additional noise} \\
 \log p(Y) \simeq & \log \int p(x) \prod_{i=1}^n \mathcal{N}(Ty_i | x_i, \Sigma_T) dx \\
 & - \underbrace{\frac{1}{2} \sum_{i=1}^n \|y_i - HTy_i\|_{\Sigma}^2}_{\text{data "lost" by projection (reconstruction error)}} - \underbrace{\frac{1}{2} n \log \frac{|\Sigma|}{|\Sigma_T|}}_{\text{noise "lost" by projection}}.
 \end{aligned}$$

- Learn $H \implies$ learn $T \implies$ learn a transform of the data!
 - "Regularisation terms" prevent underfitting.
- ✗ Requires additional projected noise Σ_T .
- Can we eliminate Σ_T ?

- Consider case $y(t) \in \mathbb{R}^2$ and $x(t) \in \mathbb{R}$:



- ε_{\parallel} is responsible for Σ_T !
- Idea: Set $\varepsilon_{\parallel} = 0$ to obtain $\Sigma_T = 0$.


- Consider $\Sigma = \sigma^2 I$. Then $T = H^\dagger$.

1 Decompose

$$\sigma^2 I = \underbrace{\sigma_{\parallel}^2 U U^\top}_{\text{orth proj. onto col}(H)} + \sigma_{\perp}^2 \overbrace{N N^\top}^{\text{orth proj. onto col}(H)^\perp}.$$

- 2 Take $\sigma_{\parallel}^2 \rightarrow 0$. Then $\Sigma_T \rightarrow 0$.

- 3 Optimise over σ_{\perp}^2 : measure of goodness of fit for H

$$\sigma_{\perp}^2 = \frac{1}{n(p-m)} \|(I - H H^\dagger) Y\|_F^2.$$


$$\log p_y(Y) \simeq \log p_x(H^\dagger Y) - \frac{1}{2}n \log |H^\top H| - \frac{1}{2}n(p-m) \log(\|(I - HH^\dagger)Y\|_F^2)$$

- ✓ Learning and inference for $p(y)$ use existing routines for $p(x)$.
- ✓ Linear scaling in number of outputs.
- Possible to generalise to general Σ .

Missing data is tricky:

- 1 Cannot compute $H^\dagger Y$.
- 2 No mechanism to “tell $p(x)$ ”.

Proposition: Missing data is equivalent to adding noise to $p(x)$:

$$\Sigma_{\text{miss}} = \sigma_{\perp}^2 \left[((L^T H)^T (L^T H))^{-1} - (H^T H)^{-1} \right].$$

$x \sim \text{GPAR}$:

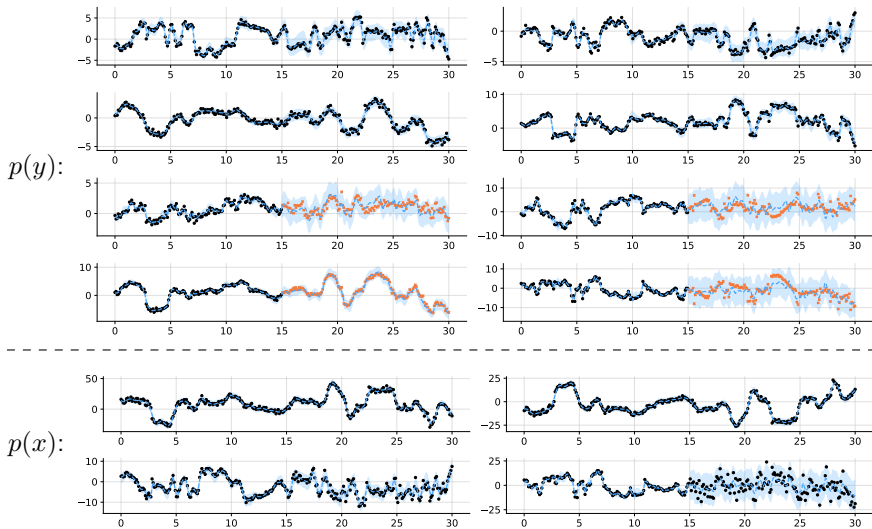
- ✗ Learning requires subtle approximation.
- ✓ Can produce exact posterior samples.

- $y(t) \in \mathbb{R}^8$ and $x(t) \in \mathbb{R}^4$.
- **Markov** GPAR: $x_i(t)$ depends nonlinearly only on $(t, x_{i-1}(t))$.
- Data generated by sample from model with random H .

- **Task:** Impute outputs 5–8 from outputs 1–4.
- Parameters randomly initialised and learned.

Experiment: Missing Data (2)

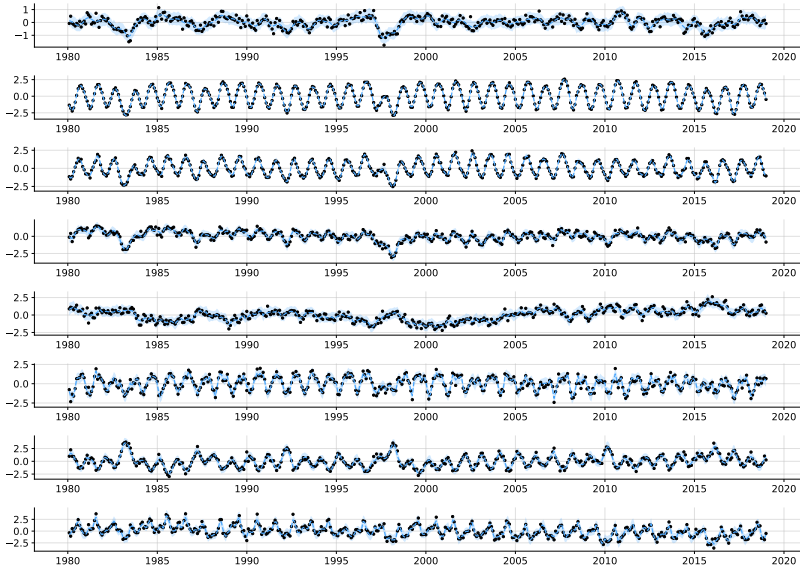
15/19



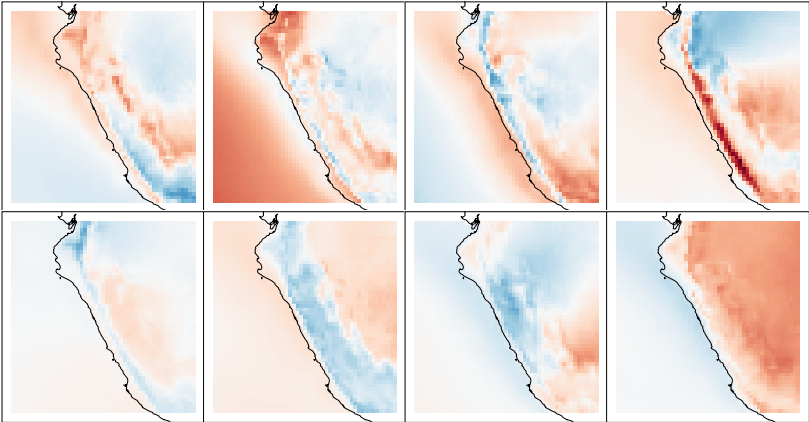
- Average monthly temp. in Peru from 1980 to 2018 ($n = 468$).
- Measured at $p = 2808$ locations.
- GPAR with $m = 8$ outputs and nonlinear dependencies.
- Basis h_1, \dots, h_8 constrained to be orthogonal.

Experiment: Temperature in Peru (2)

17/19



Experiment: Temperature in Peru (3)



- Compose $p(x)$ with a likelihood $p(y | x)$ to scale to many outputs:

high-dim. model ← $p(y) = \int p(y | x)p(x) dx$.

maps into high-dim. space ← $p(y | x)$ → available model

- Likelihood is linear and uses **orthogonal noise**:

$$y(t) = h_1x_1(t) + \dots + h_mx_m(t) + \varepsilon_{\perp}(t).$$

- ✓ Learning and inference for $p(y)$ use existing routines for $p(x)$.

These slides: <https://wesselb.github.io/pdf/compositional>.