Compositional Model Design: High-Dimensional Multi-Output Regression

Wessel Bruinsma

University of Cambridge, CBL

13 May 2020

Problem Setup

✗ computational complexity

✗ model complexity

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

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

Problem: Data may be very high-dimensional ($\sim$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.

Approach

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

$$\text{high-dim. model} \longleftarrow p(y) = \int \underbrace{p(y\,|\,x)}_{\text{maps into high-dim. space}} \underbrace{p(x)}_{\text{available model}}\,\mathrm{d}x.$$

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.

The Dream


```
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`.
```

Likelihood Model: $p(y \mid x)$

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

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

Linear model:

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

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

Likelihood Model: $p(y \mid x)$ (2)

Full generative model:

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

$$
\begin{aligned}
x &: \text{“latent processes”}, \\
H &: \text{“basis” or “mixing matrix”}.
\end{aligned}
$$

- ✓ 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.

Gaussian Process Autoregressive Model

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

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

- ✓ 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 $p(x)$

Key Result

Projection of the data:

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

Then use inference for $p(x)$!

Proposition:

"projected noise"

1. $p(f \mid \overbrace{Y}^{\mathbb{R}^{p\times n}}) = p(f \mid \overbrace{TY}^{\mathbb{R}^{m\times n}})$ with $Ty_i \mid x_i \sim \mathcal{N}(x_i, \Sigma_T)$.
2. $p(Y) = \left[\prod_{i=1}^{n} \frac{\mathcal{N}(y_i \mid 0, \Sigma)}{\mathcal{N}(Ty_i \mid 0, \Sigma_T)}\right] \int p(x) \prod_{i=1}^{n} \mathcal{N}(Ty_i \mid x_i, \Sigma_T)\, \mathrm{d}x.$

Key Result (2)

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
Y \arrow[r, "\text{✗ inference in } p(y)"] \arrow[d, "\text{projection}"', "O(nmp)"] & p(y \,|\, Y) \\
TY \arrow[r, "\text{✓ inference in } p(x)"] & p(x \,|\, TY) \arrow[u, "\text{reconstruction}", "O(nmp)"']
\end{tikzcd}
$$

Key Result (3)

$$\log p(Y) \simeq \overbrace{\log \int p(x) \prod_{i=1}^{n} \mathcal{N}(Ty_i \,|\, x_i, \Sigma_T)\,\mathrm{d}x}^{\text{likelihood under } p(x) \text{ with additional noise}}$$
$$- \underbrace{\frac{1}{2}\sum_{i=1}^{n} \|y_i - HTy_i\|^2_{\Sigma}}_{\substack{\text{data ``lost'' by projection} \\ \text{(reconstruction error)}}} - \underbrace{\frac{1}{2} n \log \frac{|\Sigma|}{|\Sigma_T|}}_{\substack{\text{noise ``lost'' by} \\ \text{projection}}}.$$

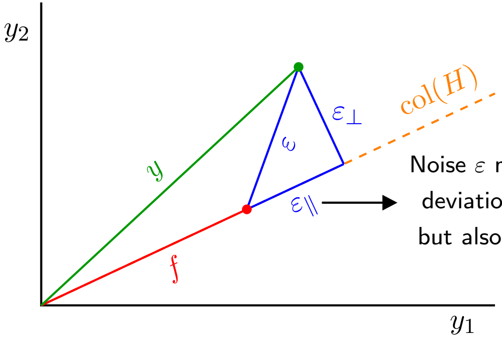
- Learn $H \implies$ learn $T \implies$ learn a transform of the data!
  - "Regularisation terms" prevent underfitting.

✗ Requires additional projected noise Σ_T.

- Can we eliminate Σ_T?

Orthogonal Noise Model

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



Noise ε not only explains deviation from $\mathrm{col}(H)$, but also within $\mathrm{col}(H)$!

- $\varepsilon_\|$ is responsible for Σ_T!
- Idea: Set $\varepsilon_\| = 0$ to obtain $\Sigma_T = 0$.

Orthogonal Noise Model (2)

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

1. Decompose

$$\sigma^2 I = \sigma_\parallel^2 \underbrace{UU^\mathsf{T}}_{\text{orth proj. onto } \mathrm{col}(H)} + \sigma_\perp^2 \overbrace{NN^\mathsf{T}}^{\text{orth proj. onto } \mathrm{col}(H)^\perp}.$$

2. Take $\sigma_\parallel^2 \to 0$. Then $\Sigma_T \to 0$.
3. Optimise over $\sigma_\perp^2$:

measure of goodness of fit for H

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

Orthogonal Noise Model (3)

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

- ✓ 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

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 $p(x)$:

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

$x \sim \mathrm{GPAR}$:

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

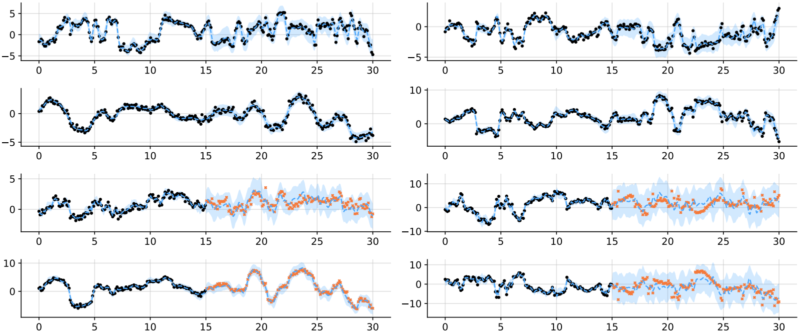
Experiment: Missing Data

- $y(t) \in \mathbb{R}^8$ 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.

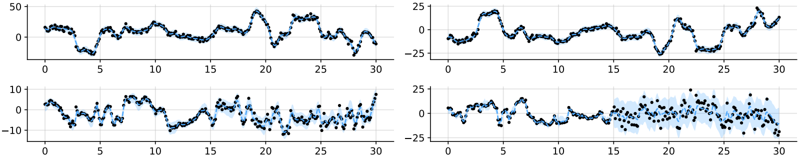
- Task: Impute outputs 5–8 from outputs 1–4.
- Parameters randomly initialised and learned.

Experiment: Missing Data (2)

$p(y)$:



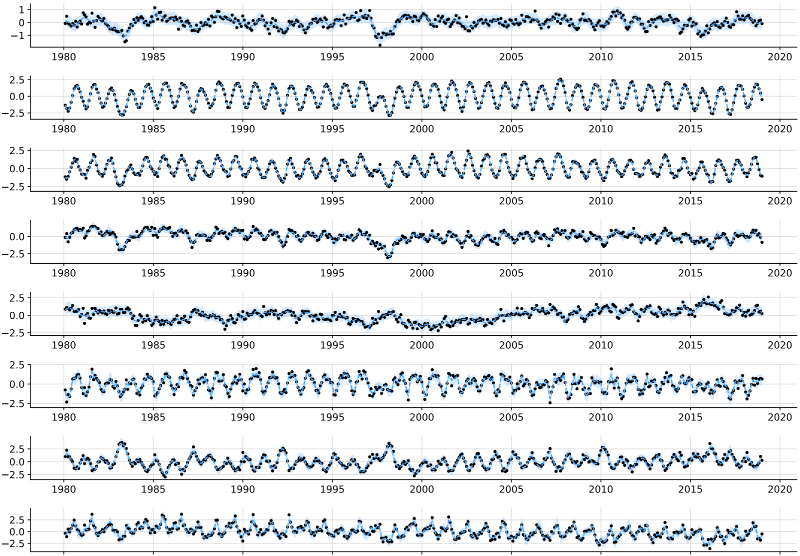
$p(x)$:



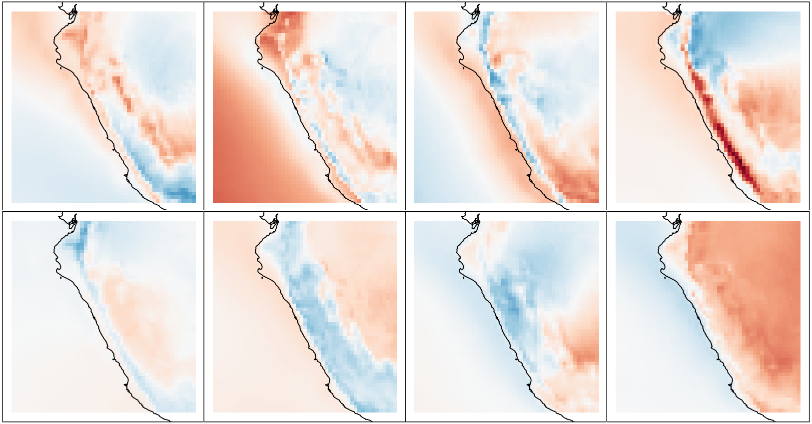
Experiment: Temperature in Peru

- 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 $h_1, \ldots, h_8$ constrained to be orthogonal.

Experiment: Temperature in Peru (2)



Experiment: Temperature in Peru (3)



Summary

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

$$\text{high-dim. model} \longleftarrow p(y) = \int \underbrace{p(y \mid x)}_{\text{maps into high-dim. space}} \underbrace{p(x)}_{\text{available model}} \, \mathrm{d}x.$$

- Likelihood is linear and uses orthogonal noise:

$$y(t) = h_1 x_1(t) + \cdots + h_m x_m(t) + \varepsilon_\perp(t).$$

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

These slides: `https://wessel.page.link/compositional`.