Latent Gaussian Processes for Distribution Estimation of Multivariate Categorical Data

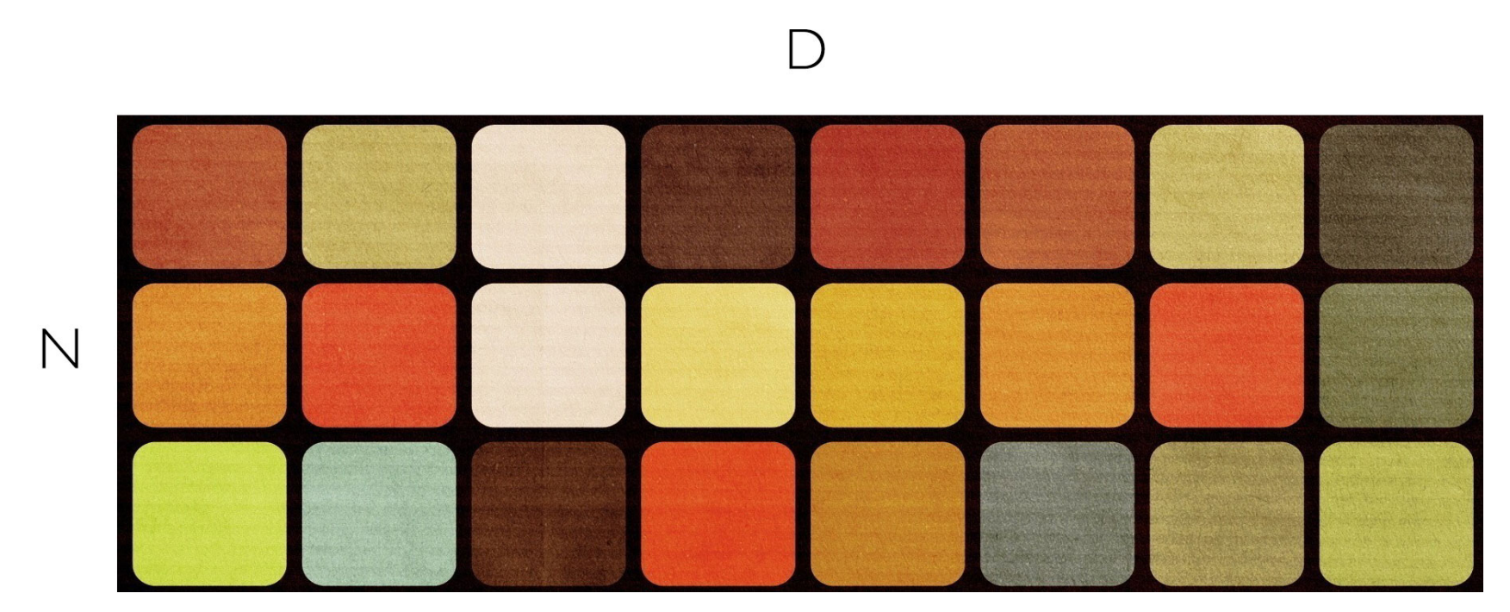
Yarin Gal, Yutian Chen, Zoubin Ghahramani

University of Cambridge

In short:

Multivariate categorical data –

- **occur frequently** in data analysis, language processing, medical diagnosis, etc.
- is challenging; the number of **possible discrete observation vectors grows exponentially** with the number of categorical variables in the vector.
- is sparsely sampled; the **diversity of data points is poor** compared to the exponentially many possible observations.



- We develop a model for **distribution estimation** of multivariate categorical data:

$$P(\mathbf{y} \mid \{\mathbf{y}_n = (y_{n1}, ..., y_{nD}) \mid n = 1, ..., N\})$$

- We use a **continuous** latent Gaussian space and learn a non-linear transformation between it and the multivariate categorical observation space.
- We derive inference for our model based on recent developments in **sampling-based variational inference and stochastic optimisation**.

Relations to other models

Existing approaches use –

- **Discrete representations**: based on frequencies of observations, but cannot handle sparse samples well (e.g. Dirichlet-Multinomial).
- **Continuous representations**: linearly transform a latent space before discretisation, but cannot capture multi-modality in the data (e.g. latent Gaussian model).

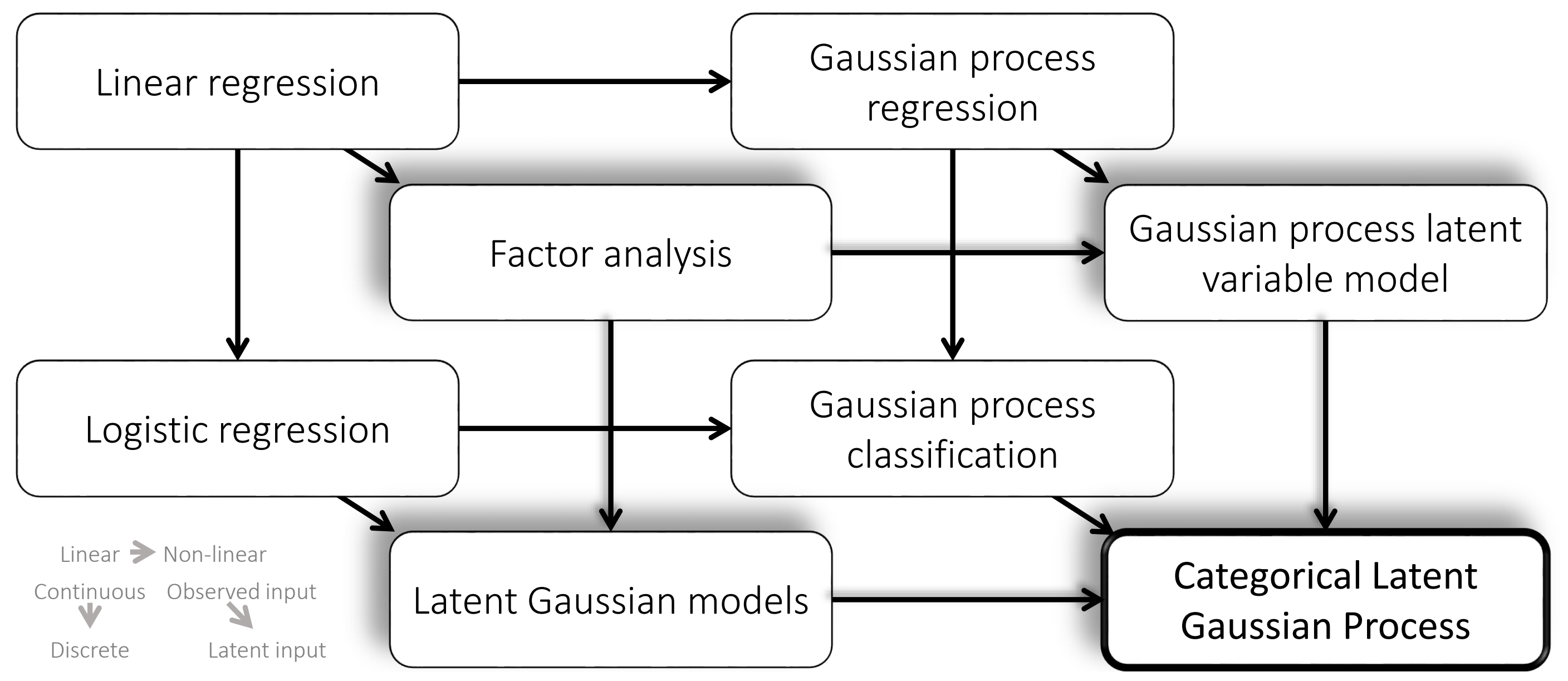

Linear regression
Gaussian process regression
Factor analysis
Gaussian process latent variable model
Logistic regression
Gaussian process classification
Linear → Non-linear
Continuous ↓ Observed input
Discrete
Latent input
Latent Gaussian models
Categorical Latent Gaussian Process

Figure 1: The model we propose can be seen as a non-linear version of the *latent Gaussian model* (left to right, Khan et al. (2012)), as a latent counterpart to the *Gaussian process (GP) classification* model (back to front, Williams and Rasmussen (2006)), or as a discrete extension of the *Gaussian process latent variable model* (top to bottom, Lawrence (2005)).

The Categorical Latent Gaussian Process

We define the generative model, with kernel $\mathbf{K}(\cdot, \cdot)$, as

$$\mathbf{x}_n \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_x^2 I), \quad (f_{ndk})_{n=1}^N \sim \mathcal{N}(0, \mathbf{K}((\mathbf{x}_n)_{n=1}^N)), \quad y_{nd} \overset{\text{iid}}{\sim} \text{Softmax}(f_{nd1}, ..., f_{ndK}).$$

Following a breast cancer diagnosis example, each **patient** is modelled by latent $\mathbf{x}_n$; for each **examination** d, $\mathbf{x}_n$ has a sequence of weights $(f_{nd1}, ..., f_{ndK})$, one weight for each possible **test result** k; Softmax returns test result y_{nd} based on these weights, resulting in a patient's **medical assessment** $\mathbf{y}_n = (y_{n1}, ..., y_{nD})$.

Inference

- We use **Sparse GPs** to get linear time complexity – we condition the observations on M inducing inputs $\mathbf{Z}$ with inducing outputs $\mathbf{U}$ with a Gaussian prior.
- Our marginal log-likelihood is intractable. We lower bound the log evidence with a variational approximate posterior $q(\mathbf{X}, \mathbf{F}, \mathbf{U}) = q(\mathbf{X})q(\mathbf{U})p(\mathbf{F}|\mathbf{X}, \mathbf{U})$, with

$$\begin{aligned} x_{ni} &= m_{ni} + s_{ni}\varepsilon_{ni}^{(x)} & \varepsilon_{ni}^{(x)} &\sim \mathcal{N}(0,1) \\ \mathbf{u}_{dk} &= \boldsymbol{\mu}_{dk} + \mathbf{L}_d\boldsymbol{\varepsilon}_{dk}^{(u)} & \boldsymbol{\varepsilon}_{dk}^{(u)} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M) \\ f_{ndk} &= \mathbf{a}_n^T\mathbf{u}_{dk} + \sqrt{b_n}\varepsilon_{ndk}^{(f)} & \varepsilon_{ndk}^{(f)} &\sim \mathcal{N}(0,1) \end{aligned}$$

and

$$\mathbf{a}_n = \mathbf{K}_{MM}^{-1}\mathbf{K}_{Mn}, \quad b_n = K_{nn} - \mathbf{K}_{nM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{Mn}.$$

Then,

$$\begin{aligned} \log p(\mathbf{Y}) &= \log \int p(\mathbf{X})p(\mathbf{U})p(\mathbf{F}|\mathbf{X}, \mathbf{U})p(\mathbf{Y}|\mathbf{F})\mathrm{d}\mathbf{X}\mathrm{d}\mathbf{F}\mathrm{d}\mathbf{U} \\ &\geq -\mathrm{KL}(q(\mathbf{X})\|p(\mathbf{X})) - \mathrm{KL}(q(\mathbf{U})\|p(\mathbf{U})) \\ &\quad + \sum_{n=1}^N \sum_{d=1}^D \mathbb{E}_{\boldsymbol{\varepsilon}_n^{(x)}, \boldsymbol{\varepsilon}_d^{(u)}, \boldsymbol{\varepsilon}_{nd}^{(f)}} \log \text{Softmax}\left(\mathbf{y}_{nd} \middle| \mathbf{f}_{nd}\left(\boldsymbol{\varepsilon}_{nd}^{(f)}, \mathbf{U}_d(\boldsymbol{\varepsilon}_d^{(u)}), \mathbf{x}_n(\boldsymbol{\varepsilon}_n^{(x)})\right)\right). \end{aligned}$$

Method

1. Monte Carlo integration approximates the likelihood obtaining noisy gradients:

$$\mathbb{E}_{\boldsymbol{\varepsilon}_n^{(x)}, \boldsymbol{\varepsilon}_d^{(u)}, \boldsymbol{\varepsilon}_{nd}^{(f)}} \log \text{Softmax}(\cdot) \approx \frac{1}{T}\sum_{i=1}^T \log \text{Softmax}\left(\mathbf{y}_{nd} \middle| \mathbf{f}_{nd}\left(\boldsymbol{\varepsilon}_{nd}^{(f)}, \mathbf{U}_d(\boldsymbol{\varepsilon}_d^{(u)}), \mathbf{x}_n(\boldsymbol{\varepsilon}_n^{(x)})\right)\right)$$

2. Learning-rate free stochastic optimisation is used to optimise the noisy objective.
3. Symbolic differentiation is used to get simple and modular code:

```python
import theano.tensor as T
X = m + s * randn(N, Q)
U = mu + L.dot(randn(M, K))
Kmm, Kmn, Knn = RBF(sf2, l, Z), RBF(sf2, l, Z, X), RBFnn(sf2, l, X)
KmmInv = sT.matrix_inverse(Kmm)
A = KmmInv.dot(Kmn)
B = Knn - T.sum(Kmn * KmmInv.dot(Kmn), 0)
F = A.T.dot(U) + B[:,None]**0.5 * randn(N, K)
S = T.nnet.softmax(F)
KL_U, KL_X = get_KL_U(), get_KL_X()
LS = T.sum(T.log(T.sum(Y * S, 1))) - KL_U - KL_X
LS_func = theano.function(['''inputs'''], LS)
dLS_dm = theano.function(['''inputs'''], T.grad(LS, m)) # and others
# ... and run RMS-PROP
```

That's all.

Experiments

- **Linear models have difficulty with multi-modal distributions**
  - We use the simple XOR dataset capturing the non-linear relation based on observations of triplets such as $(1, 1, 0)$.

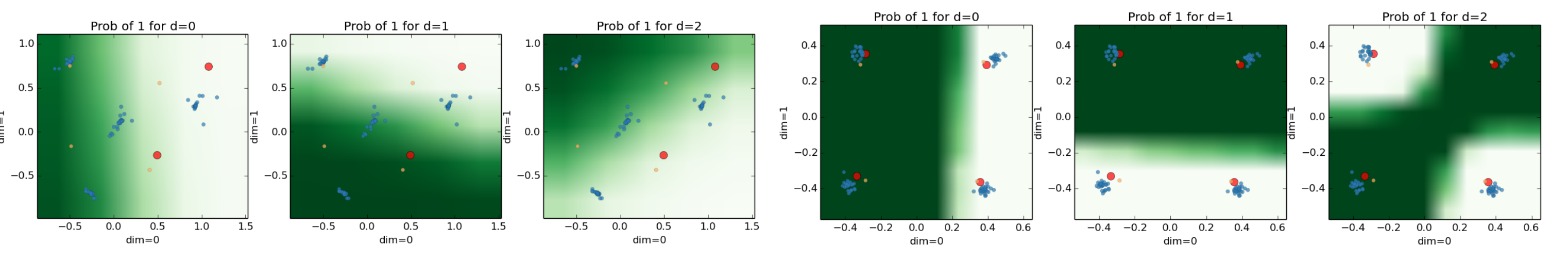


Figure 2: **Density over the latent space as predicted by the linear model (left 3 panels, LGM), and non-linear model (right 3 panels, CLGP).** Each panel shows the density over the same latent space corresponding to a different single variable taking value 1.

- **Real-world sparse small data domain** on the Wisconsin breast cancer dataset.
  - The number of observations is small (683) and costly to obtain,
  - Usual task is simple supervised classification, predicting the development of breast cancer in patients from 9 categorical variables each taking 10 values,
  - We look instead for **which tests are needed or can be deduced from others** in an attempt to reduce the number of unneeded examinations.

Split	Baseline	Multinomial	Uni-Dir-Mult	Bi-Dir-Mult	LGM	**CLGP**
1	8.68	4.41	4.41	3.41	3.57 ± 0.208	$\mathbf{2.86 \pm 0.119}$
2	8.68	∞	4.42	$\mathbf{3.49}$	$\mathbf{3.47 \pm 0.252}$	$\mathbf{3.36 \pm 0.186}$
3	8.85	4.64	4.64	3.67	12.13 ± 9.705	$\mathbf{3.34 \pm 0.096}$

Figure 4: **Model perplexity on Breast cancer dataset, predicting randomly missing categorical test results**. The models compared are: *Baseline* predicting uniform probability for all values, *Multinomial* – predicting the probability for a missing value based on its frequency, *Uni-Dir-Mult* – Unigram Dirichlet Multinomial with concentration parameter $\alpha = 0.01$, *Bi-Dir-Mult* – Bigram Dirichlet Multinomial with concentration parameter $\alpha = 1$, *LGM*, and the proposed model (*CLGP*).

- **LGM over-fitting and inference robustness**

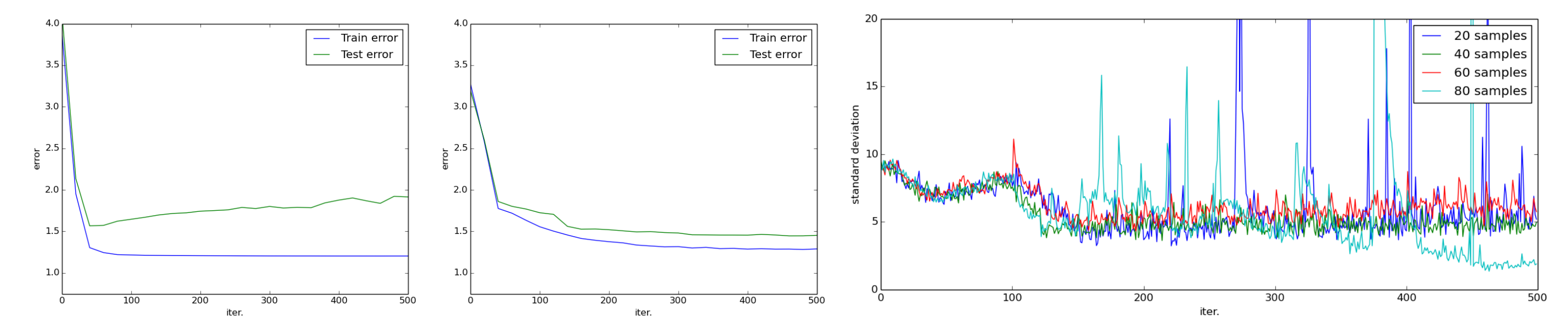


Figure 5: Train and test error for LGM (left) and the CLGP model (middle) for one of the splits of the breast cancer dataset; Standard deviation per iteration on the XOR dataset (right).

  - The train error of LGM (left) decreases while the test error starts increasing.
  - We see a slight decrease in variance with more samples (right); as the variational distribution gets closer to the true posterior, the variance seems to decrease.

Closing remarks

The entire code, consisting of 95 lines of Python + optimiser, is available online at `github.com/yaringal/CLGP`.