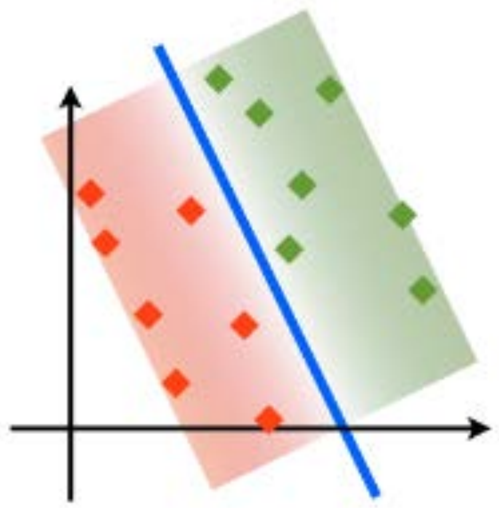
ENM 531: Data-driven Modeling and Probabilistic Scientific Computing

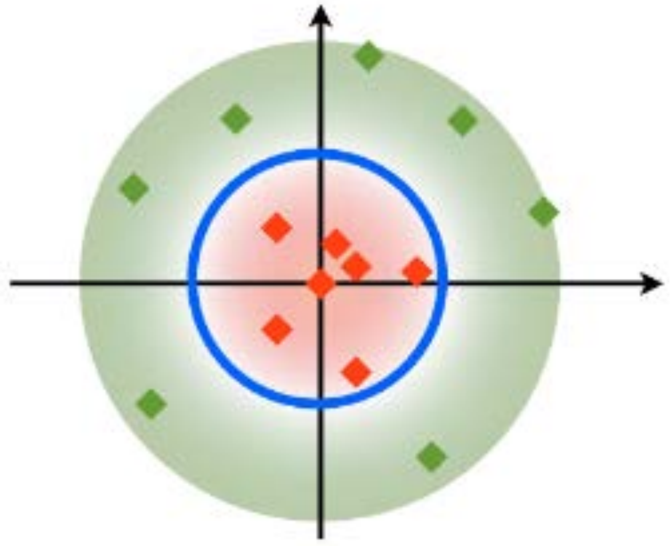
Lecture #19: Gaussian process regression

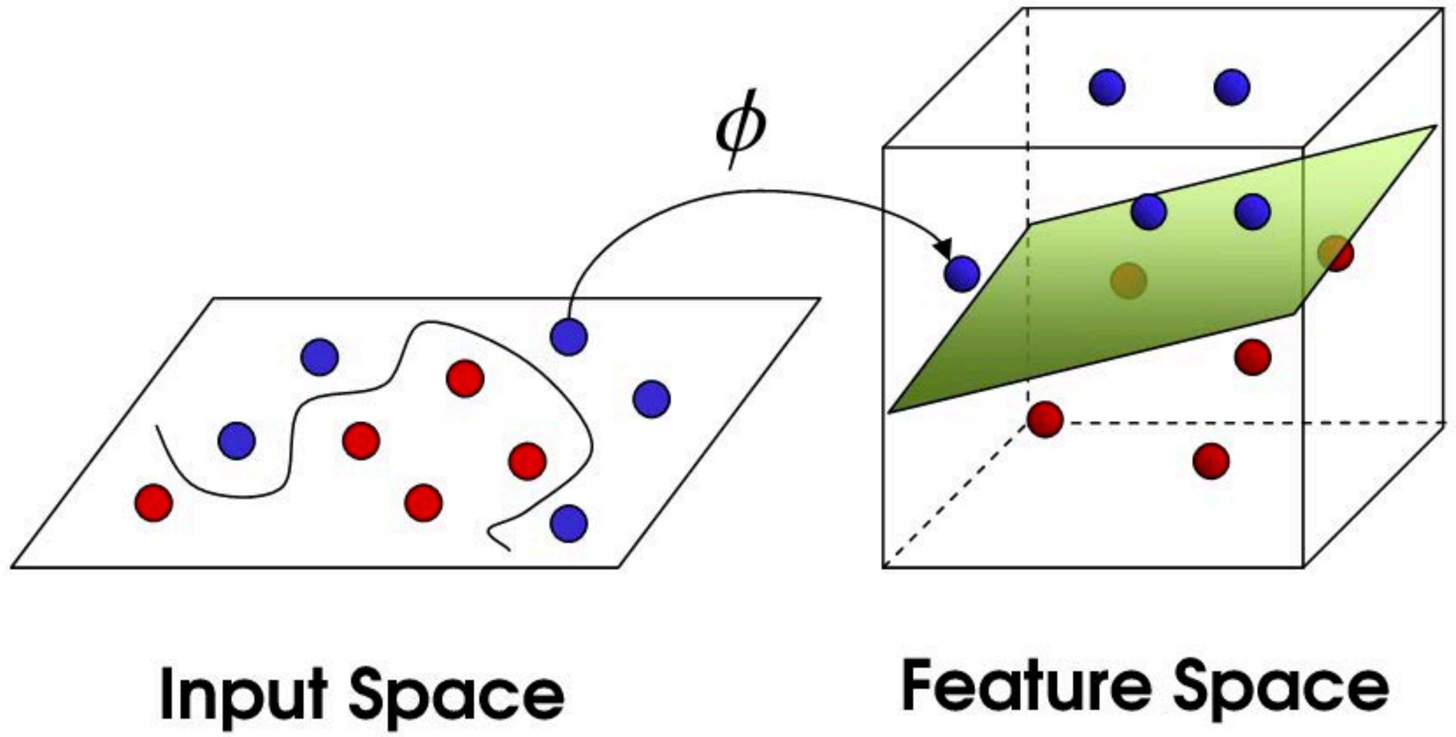
Penn
UNIVERSITY *of* PENNSYLVANIA

“Linearization” by embedding to higher dimensions

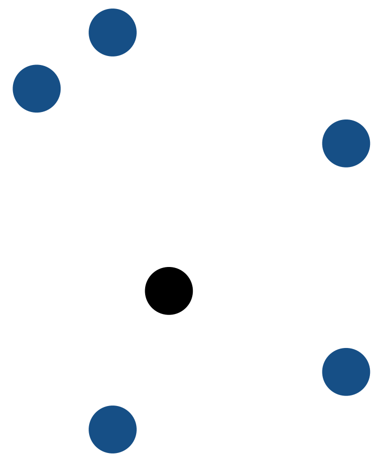
$$f(\boldsymbol{x}) = \langle \theta, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}}, \quad \phi : \mathbb{R}^d \to \mathbb{R}^m$$







Kernel methods



$$f(\boldsymbol{x}) = \langle \theta, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}}, \quad \phi : \mathbb{R}^d \to \mathbb{R}^m$$

$$m \to \infty \quad \Big\downarrow \; k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$$

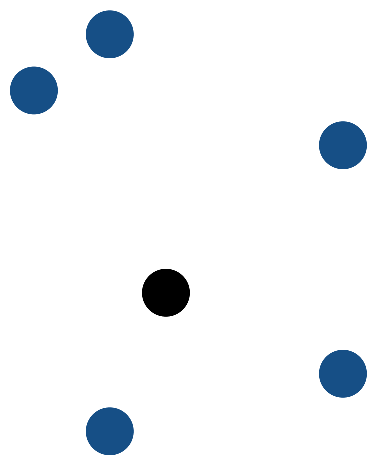
$$f(\boldsymbol{x}) = \sum_{i=1}^{n} (\boldsymbol{K}^{-1}\boldsymbol{y})_i k(\boldsymbol{x}_i, \boldsymbol{x}), \quad \boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Key take-away points:

-Kernels and representer theorems: learning with infinite-dimensional linear models can be done in time that depend on the number of observations by using a kernel function.
-Kernels on $\mathbb{R}^d$: such models include polynomials and classical Sobolev spaces (functions with square-integrable partial derivatives).
-Algorithms: convex optimization algorithms can be applied with theoretical guarantees and many dedicated developments to avoid the quadratic complexity of computing the kernel matrix.
-Analysis of well-specified models: When the target function is in the associated function space, learning can be done with rates that are independent of dimension.
-Analysis of mis-specified models: if the target is not in the the RKHS, the curse of dimensionality cannot be avoided in the worst case situations of few existing derivatives of the target function, but the methods are adaptive to any amount of intermediate smoothness.
-Sharp analysis of ridge regression: for the square loss, a more involded analysis leads to optimal rates in a variety of situations in $\mathbb{R}^d$.

https://www.di.ens.fr/~fbach/ltfp_book.pdf

Kernel methods



$$f(\boldsymbol{x}) = \langle \theta, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}}, \quad \phi : \mathbb{R}^d \to \mathbb{R}^m$$

$$m \to \infty \quad \Big\downarrow \quad k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} (\boldsymbol{K}^{-1}\boldsymbol{y})_i k(\boldsymbol{x}_i, \boldsymbol{x}), \quad \boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

$$\Big\downarrow$$

$$p(f|\mathcal{D}) = \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}$$

$$f(\boldsymbol{x}) \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'))$$

Key take-away points:

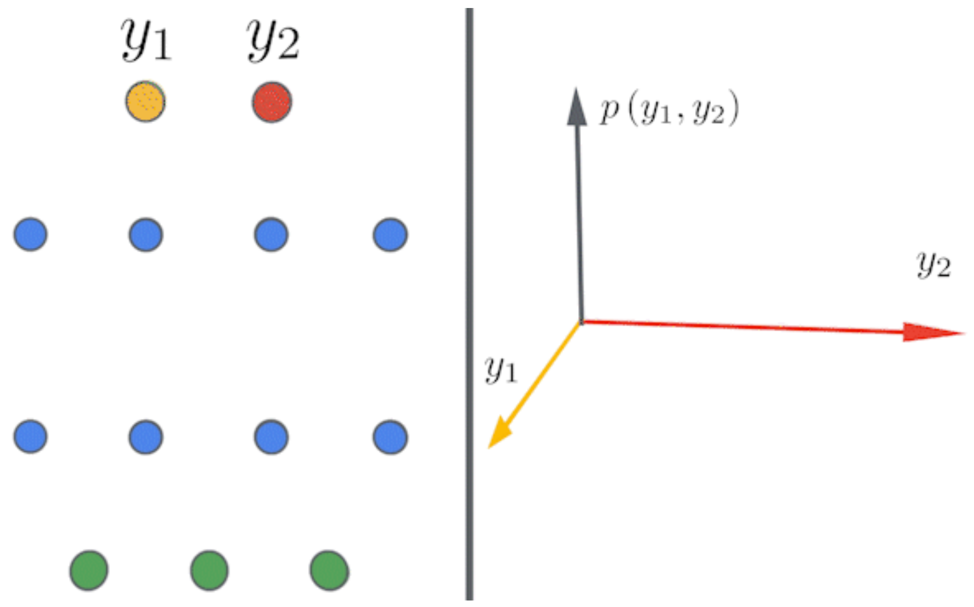
-Kernels and representer theorems: learning with infinite-dimensional linear models can be done in time that depend on the number of observations by using a kernel function.
-Kernels on $\mathbb{R}^d$: such models include polynomials and classical Sobolev spaces (functions with square-integrable partial derivatives).
-Algorithms: convex optimization algorithms can be applied with theoretical guarantees and many dedicated developments to avoid the quadratic complexity of computing the kernel matrix.
-Analysis of well-specified models: When the target function is in the associated function space, learning can be done with rates that are independent of dimension.
-Analysis of mis-specified models: if the target is not in the the RKHS, the curse of dimensionality cannot be avoided in the worst case situations of few existing derivatives of the target function, but the methods are adaptive to any amount of intermediate smoothness.
-Sharp analysis of ridge regression: for the square loss, a more involded analysis leads to optimal rates in a variety of situations in $\mathbb{R}^d$.

https://www.di.ens.fr/~fbach/ltfp_book.pdf

Infinitely wide neural networks are kernel machines

Key concepts:

- Explore the connection between deep neural networks and kernel regression methods to elucidate the training dynamics of deep learning models.
- At initialization, fully-connected networks are equivalent to Gaussian processes in the infinite-width limit.
- The evolution of an infinite-width network during training can also be described by another kernel, the so-called Neural Tangent Kernel (NTK).
- As its width tends to infinity, a deep neural network's behavior under gradient descent can become simplified and predictable, characterized by its limiting NTK.

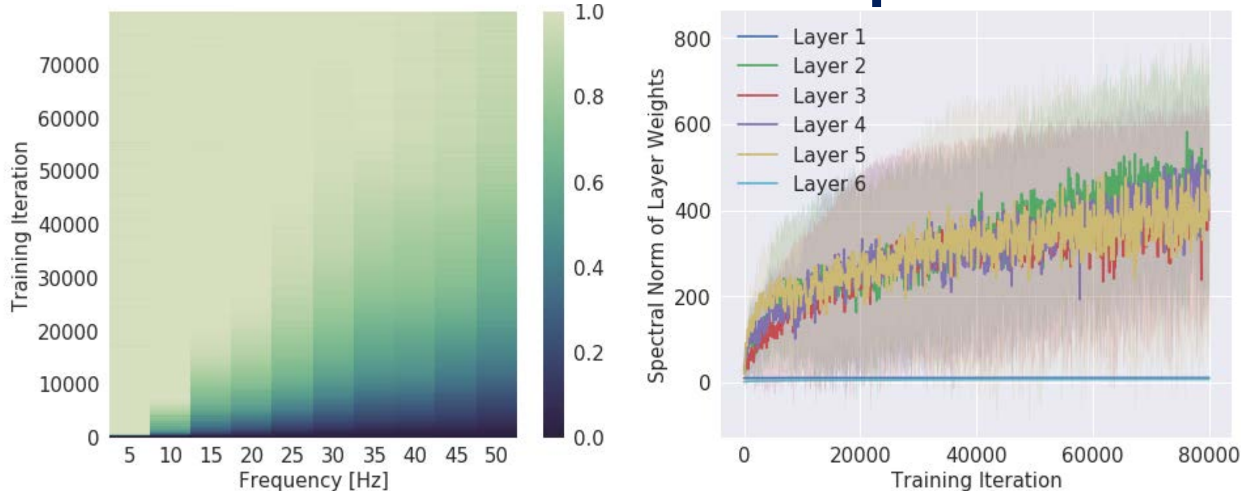


$$k_\theta(x, x') = \left\langle \frac{df_\theta(x)}{d\theta}, \frac{f_\theta(x')}{d\theta} \right\rangle$$

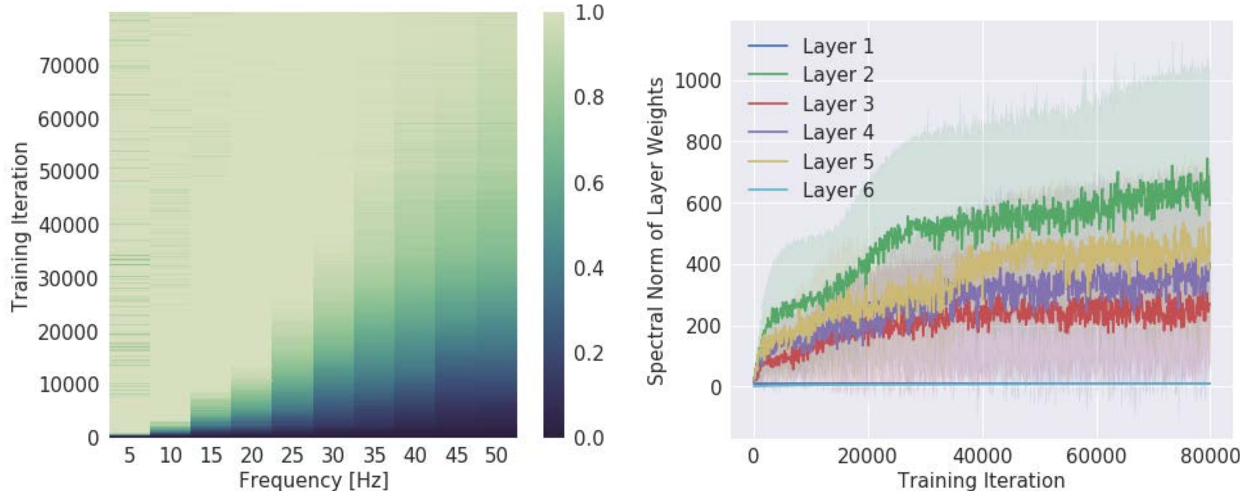
Intuition: $k(x,x')$ measures how sensitive the function value at x is to prediction errors at x′.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., & Sohl-Dickstein, J. (2017). Deep neural networks as Gaussian processes.
Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In Advances in neural information processing systems (pp. 8571-8580).
Shin, Y., Darbon, J., & Karniadakis, G. E. (2020). On the convergence and generalization of physics informed neural networks. arXiv preprint arXiv:2004.01806.

Spectral bias in MLPs

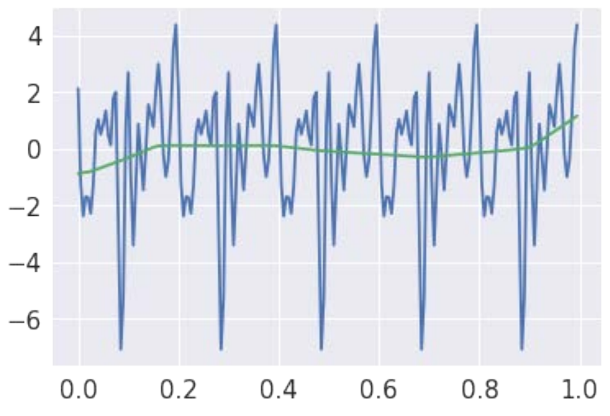


(a) Equal Amplitudes

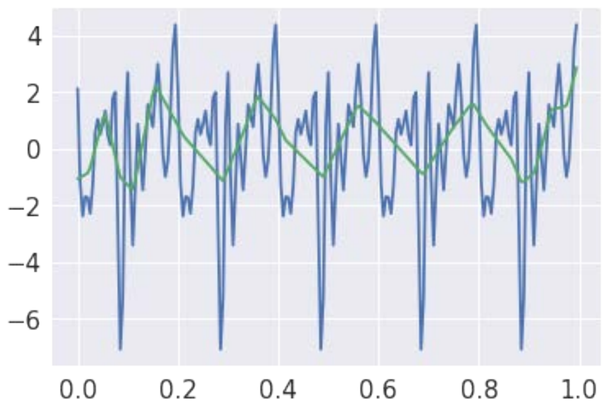


(b) Increasing Amplitudes

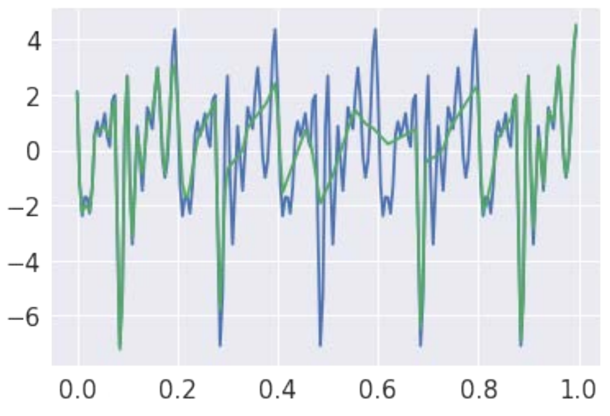
Figure 1. Left (a, b): Evolution of the spectrum (x-axis for frequency) during training (y-axis). The colors show the measured amplitude of the network spectrum at the corresponding frequency, normalized by the target amplitude at the same frequency (i.e. $|\tilde{f}_{k_i}|/A_i$) and the colorbar is clipped between 0 and 1. Right (a, b): Evolution of the spectral norm (y-axis) of each layer during training (x-axis). Figure-set (a) shows the setting where all frequency components in the target function have the same amplitude, and (b) where higher frequencies have larger amplitudes. **Gist**: We find that even when higher frequencies have larger amplitudes, the model prioritizes learning lower frequencies first. We also find that the spectral norm of weights increases as the model fits higher frequency, which is what we expect from Theorem 1.



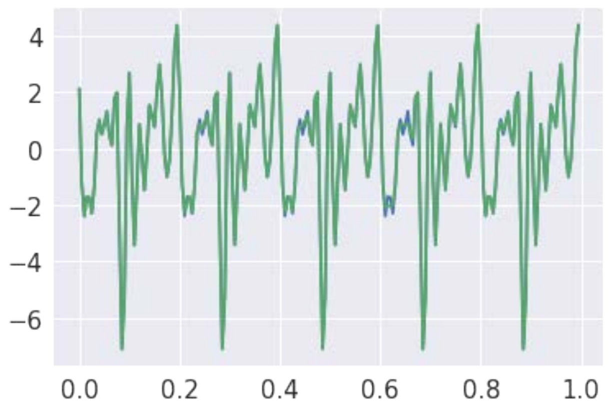
(a) Iteration 100



(b) Iteration 1000



(c) Iteration 10000



(d) Iteration 80000

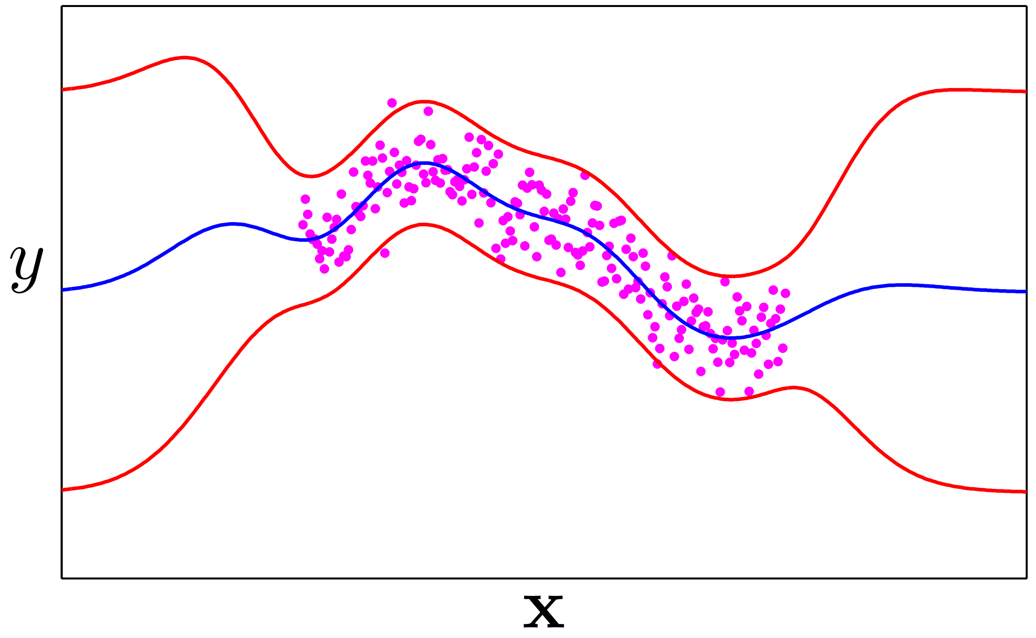
Figure 2. The learnt function (green) overlayed on the target function (blue) as the training progresses. The target function is a superposition of sinusoids of frequencies $\kappa = (5, 10, ..., 45, 50)$, equal amplitudes and randomly sampled phases.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y. and Courville, A., 2019, May. On the spectral bias of neural networks. In International Conference on Machine Learning (pp. 5301-5310). PMLR.

Nonlinear regression

Consider the problem of nonlinear regression:

You want to learn a function f with error bars from data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$



A Gaussian process defines a distribution over functions $p(f)$ which can be used for Bayesian regression:

$$p(f|\mathcal{D}) = \frac{p(f)p(\mathcal{D}|f)}{p(\mathcal{D})}$$

Carl Friedrich Gauss (1777–1855)

Paying Tolls with A Bell

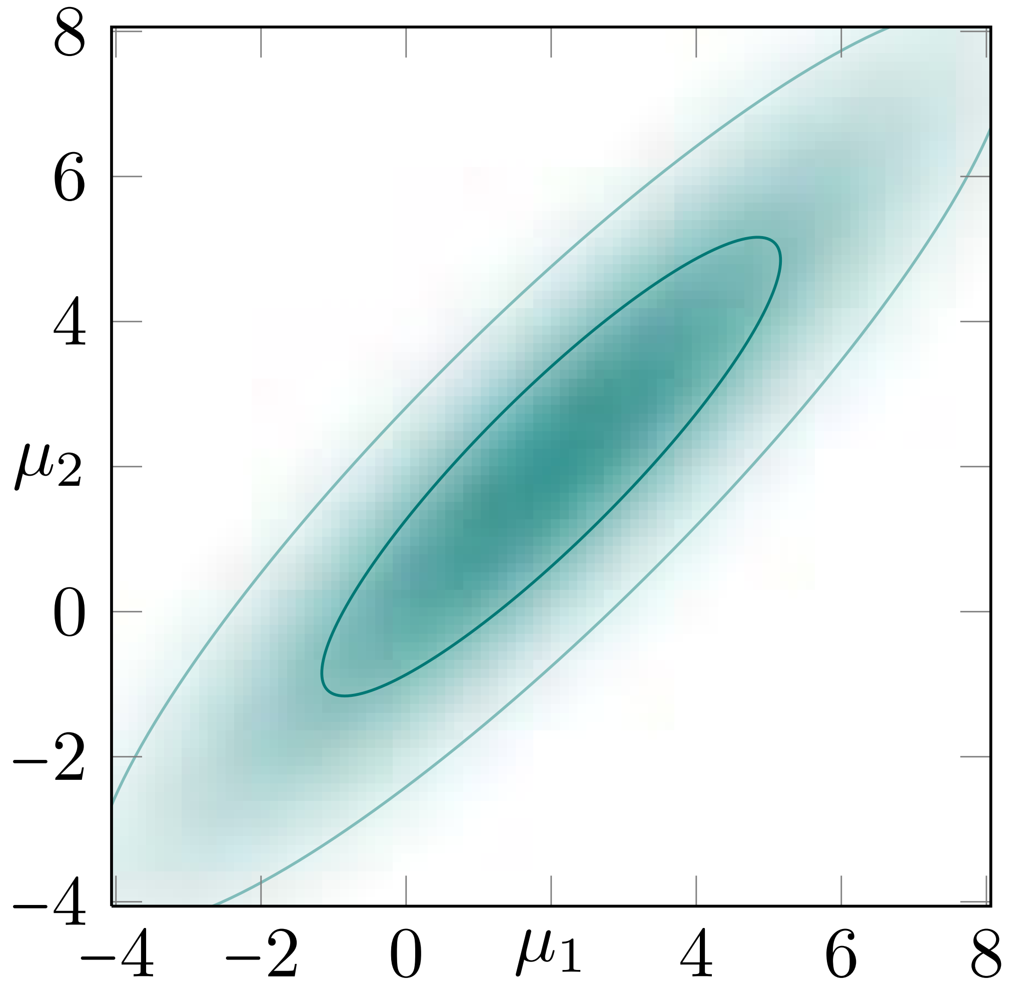
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

The Gaussian distribution

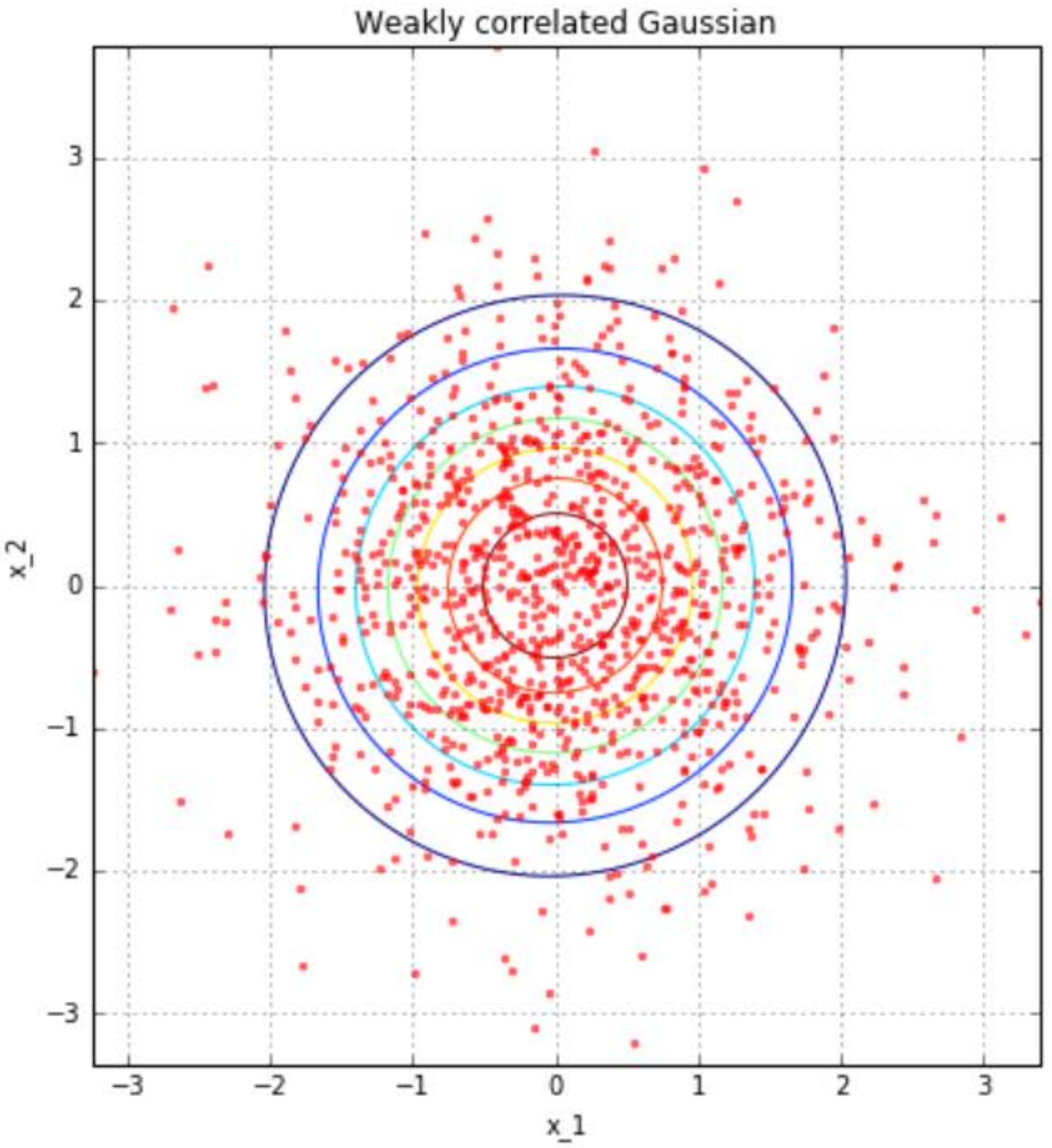
Multivariate Form

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right]$$

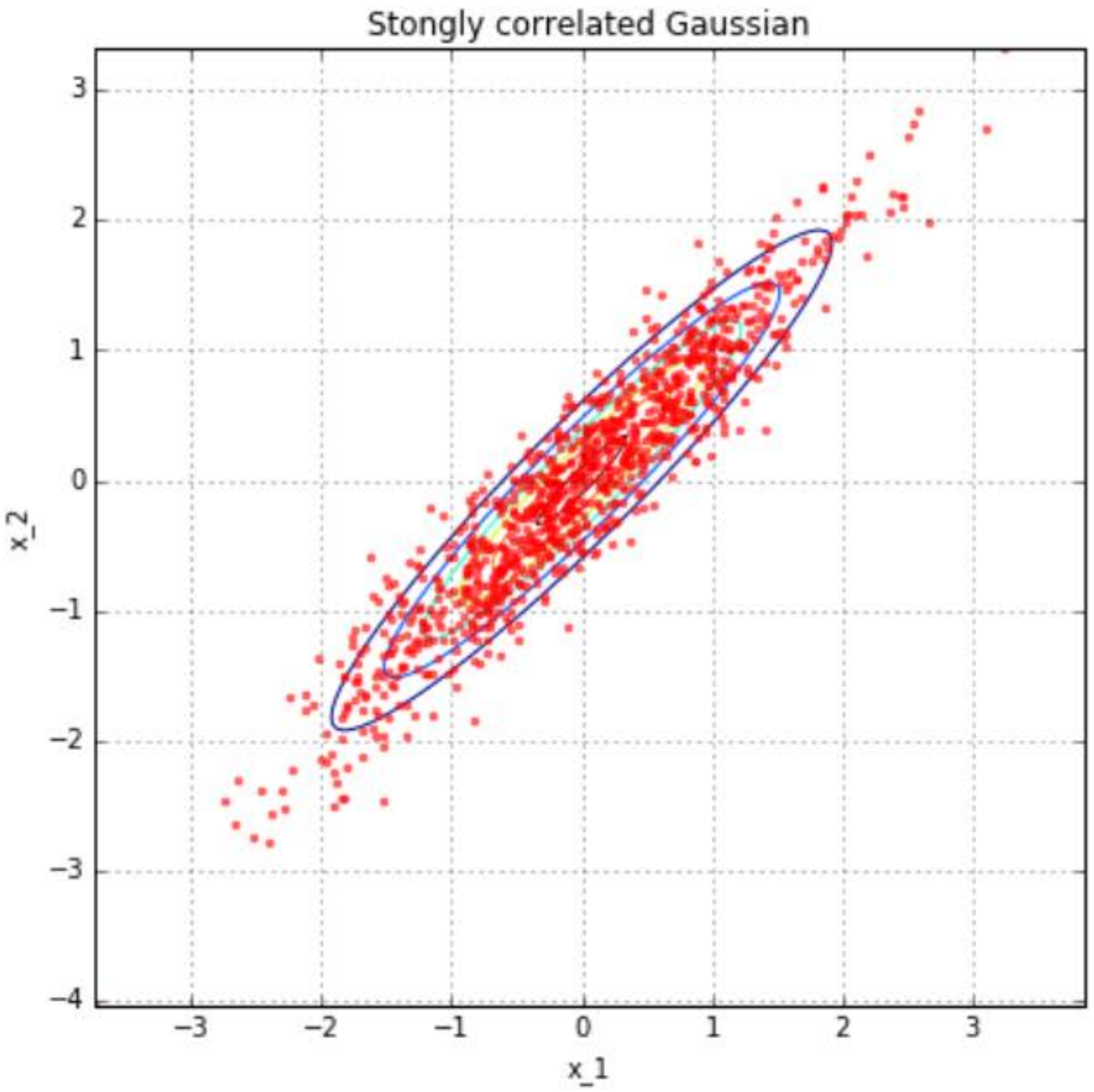


- $x, \mu \in \mathbb{R}^N$, $\Sigma \in \mathbb{R}^{N \times N}$
- Σ is positive semidefinite, i.e.
  - $v^\top \Sigma v \geq 0$ for all $v \in \mathbb{R}^N$
  - Hermitian, all eigenvalues ≥ 0

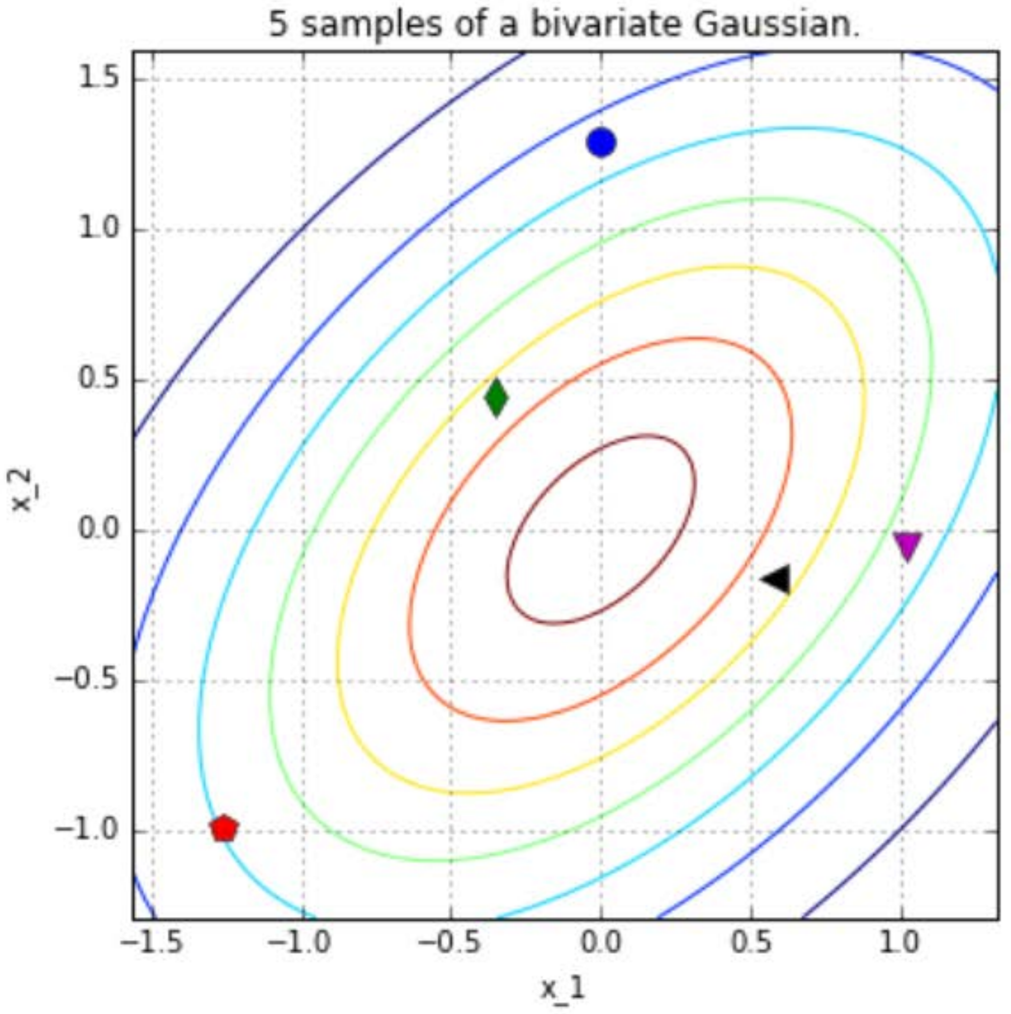
http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

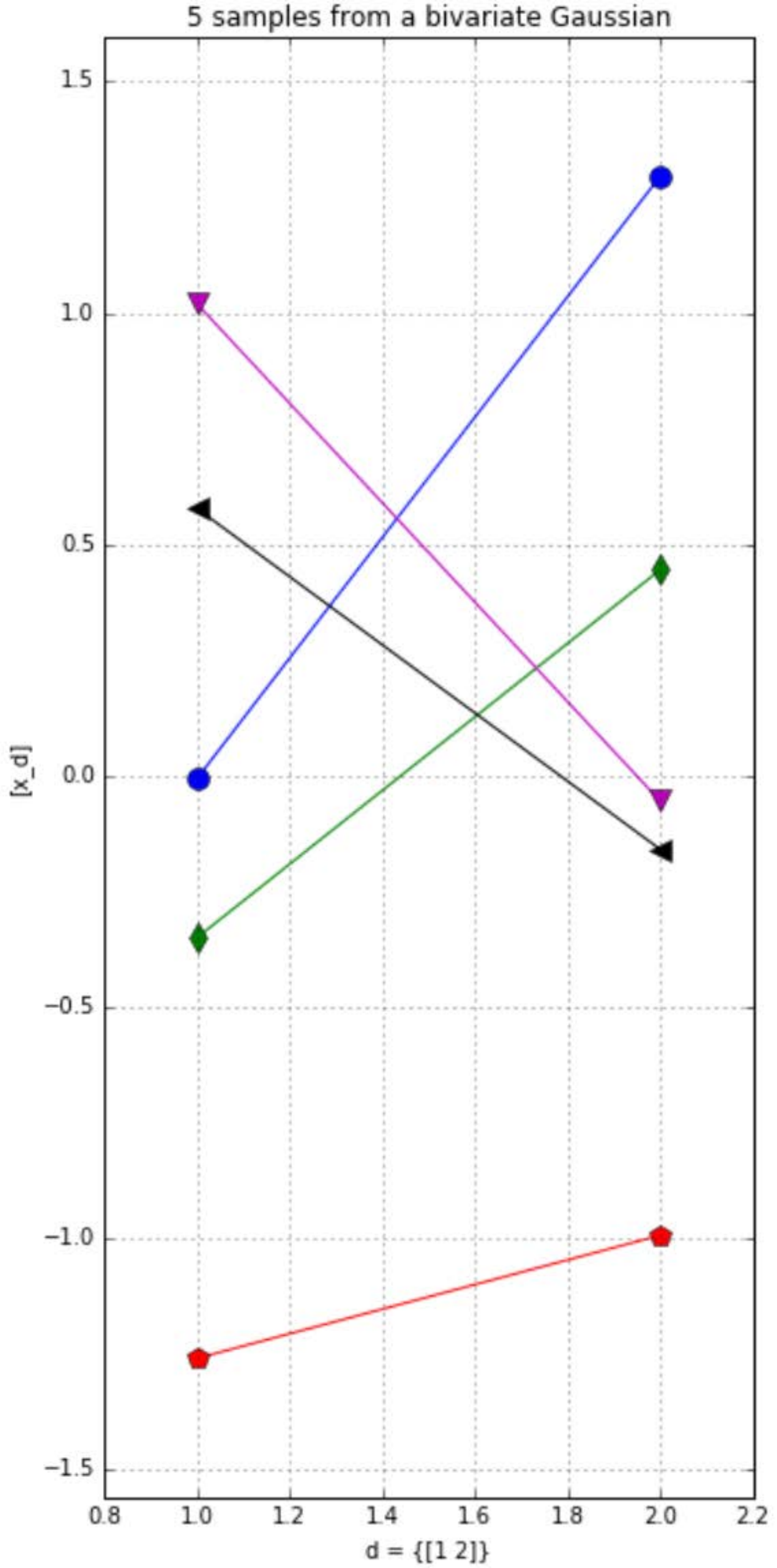


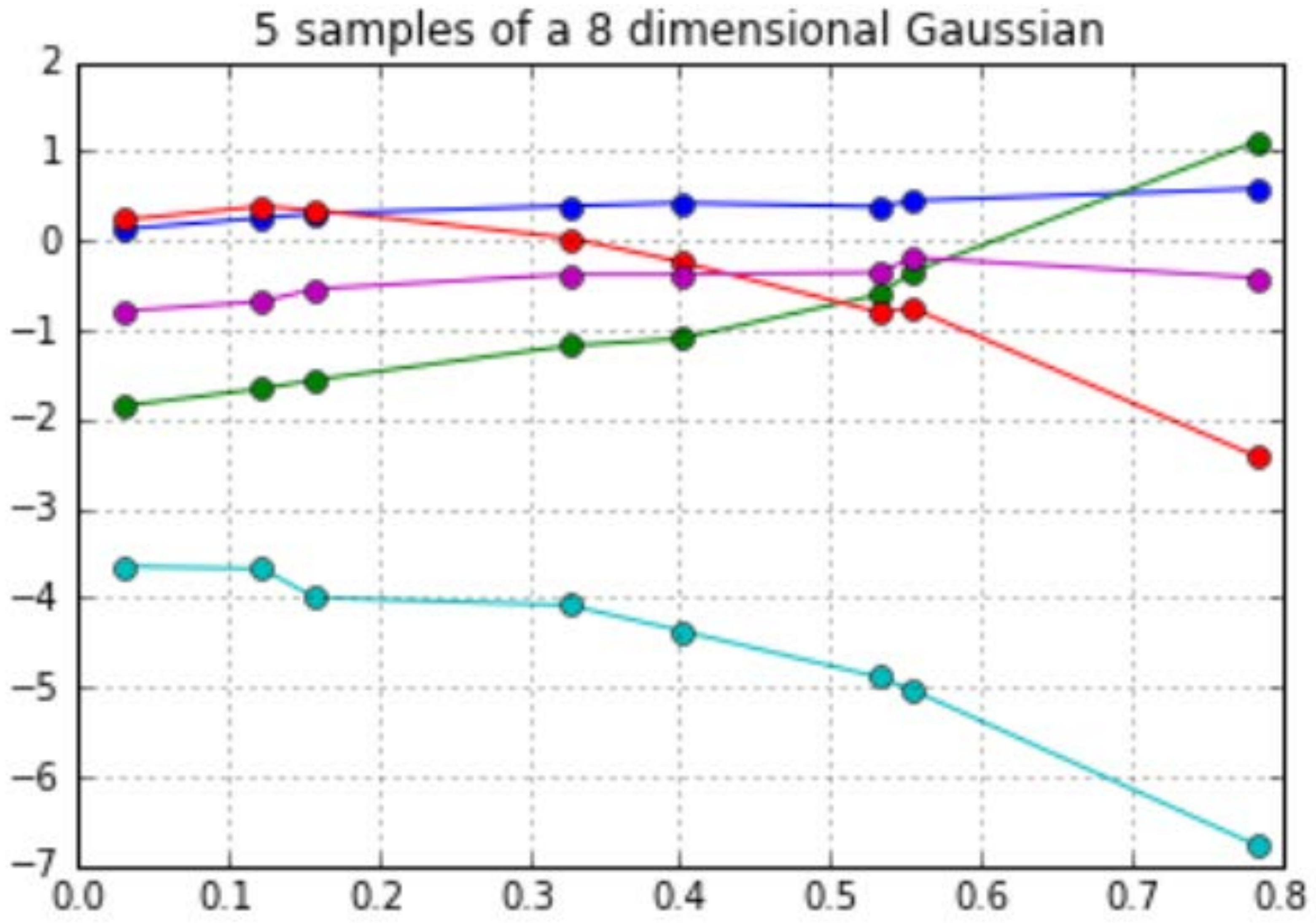
Weakly correlated Gaussian
x_1
x_2

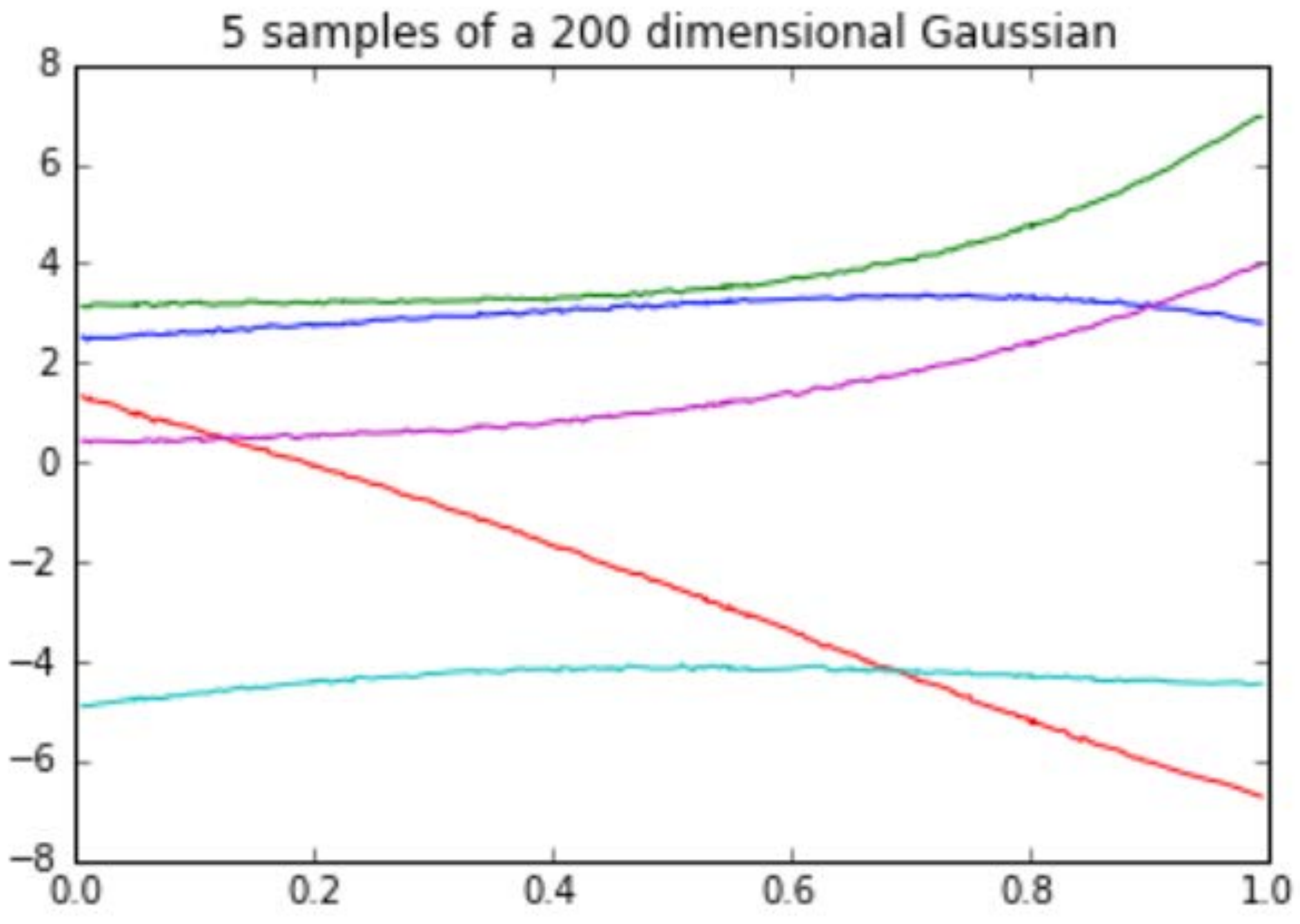


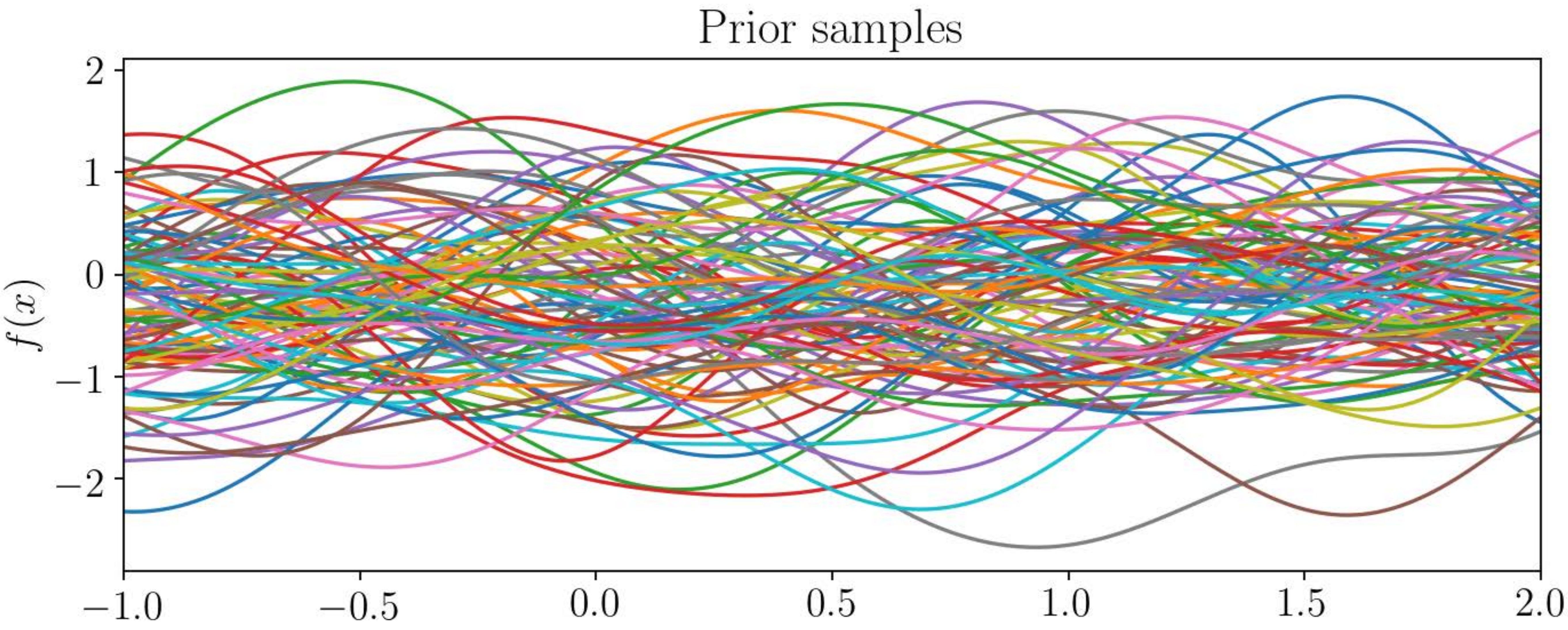
Stongly correlated Gaussian
x_1
x_2


5 samples of a bivariate Gaussian.
x_1
x_2


5 samples from a bivariate Gaussian
[x_d]
d = {[1 2]}


5 samples of a 8 dimensional Gaussian
2
1
0
−1
−2
−3
−4
−5
−6
−7
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


5 samples of a 200 dimensional Gaussian
8
6
4
2
0
−2
−4
−6
−8
0.0
0.2
0.4
0.6
0.8
1.0


Prior samples
f(x)
2
1
0
−1
−2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

From linear regression to GPs:

- Linear regression with inputs x_i and outputs y_i: $$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$
- Linear regression with M basis functions: $$y_i = \sum_{m=1}^{M} \beta_m \, \phi_m(x_i) + \epsilon_i$$
- Bayesian linear regression with basis functions:
$$\beta_m \sim \mathrm{N}(\cdot|0, \lambda_m) \quad \text{(independent of } \beta_\ell, \forall \ell \neq m\text{)}, \qquad \epsilon_i \sim \mathrm{N}(\cdot|0, \sigma^2)$$
- Integrating out the coefficients, β_j, we find:
$$E[y_i] = 0, \qquad Cov(y_i, y_j) = K_{ij} \overset{\text{def}}{=} \sum_{m=1}^{M} \lambda_m \, \phi_m(x_i) \, \phi_m(x_j) + \delta_{ij}\sigma^2$$

This is a Gaussian process with covariance function $K(x_i, x_j) = K_{ij}$.

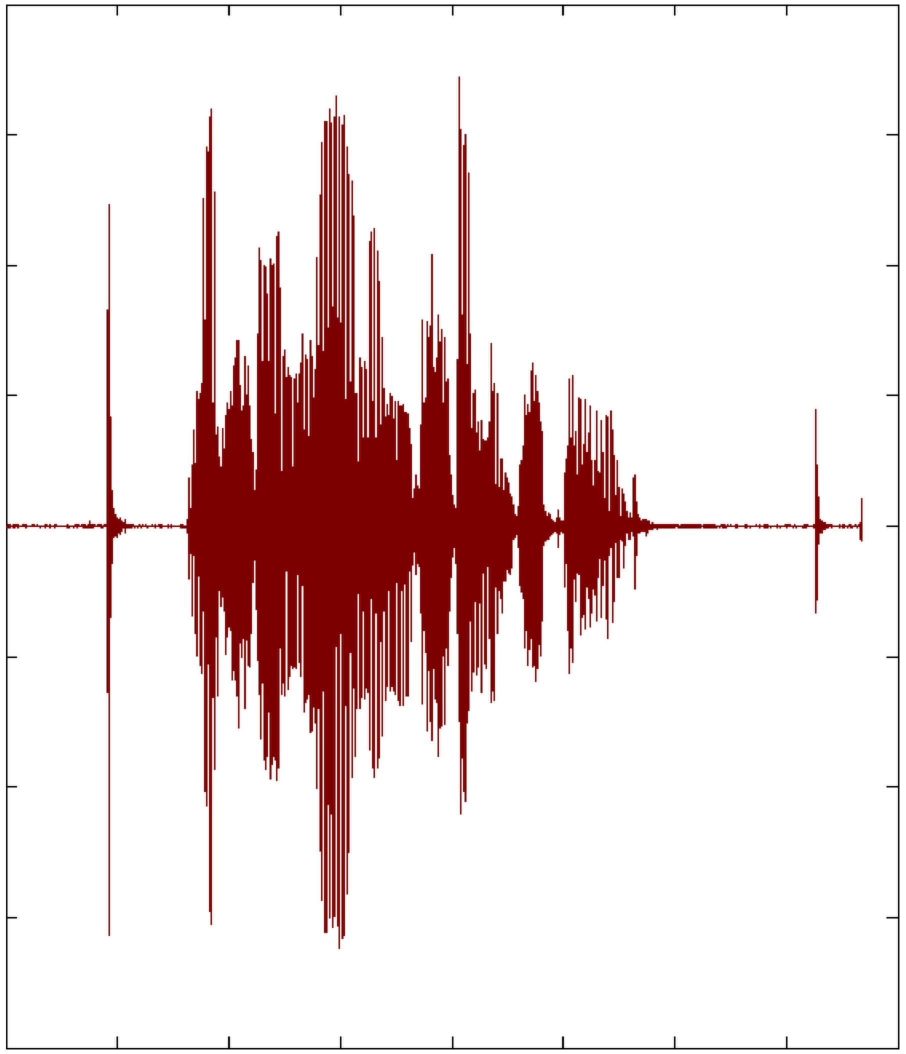
This GP has a finite number (M) of basis functions. Many useful GP kernels correspond to infinitely many basis functions (i.e. infinite-dim feature spaces).

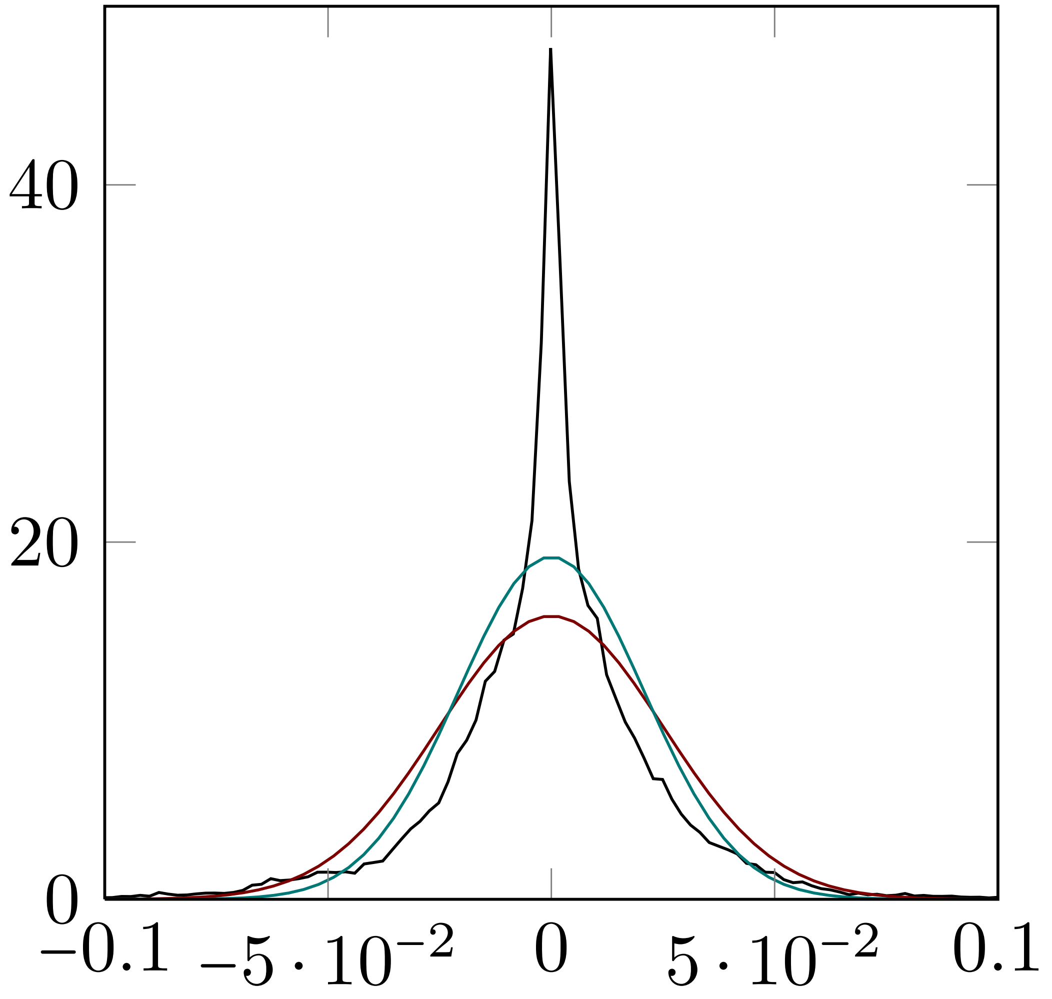
A multilayer perceptron (neural network) with infinitely many hidden units and Gaussian priors on the weights $\rightarrow$ a GP (Neal, 1996)

http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Why Gaussian?

an experiment





- nothing in the real world is Gaussian (except sums of i.i.d. variables)
- But nothing in the real world is linear either!

Gaussians are for inference what linear maps are for algebra.

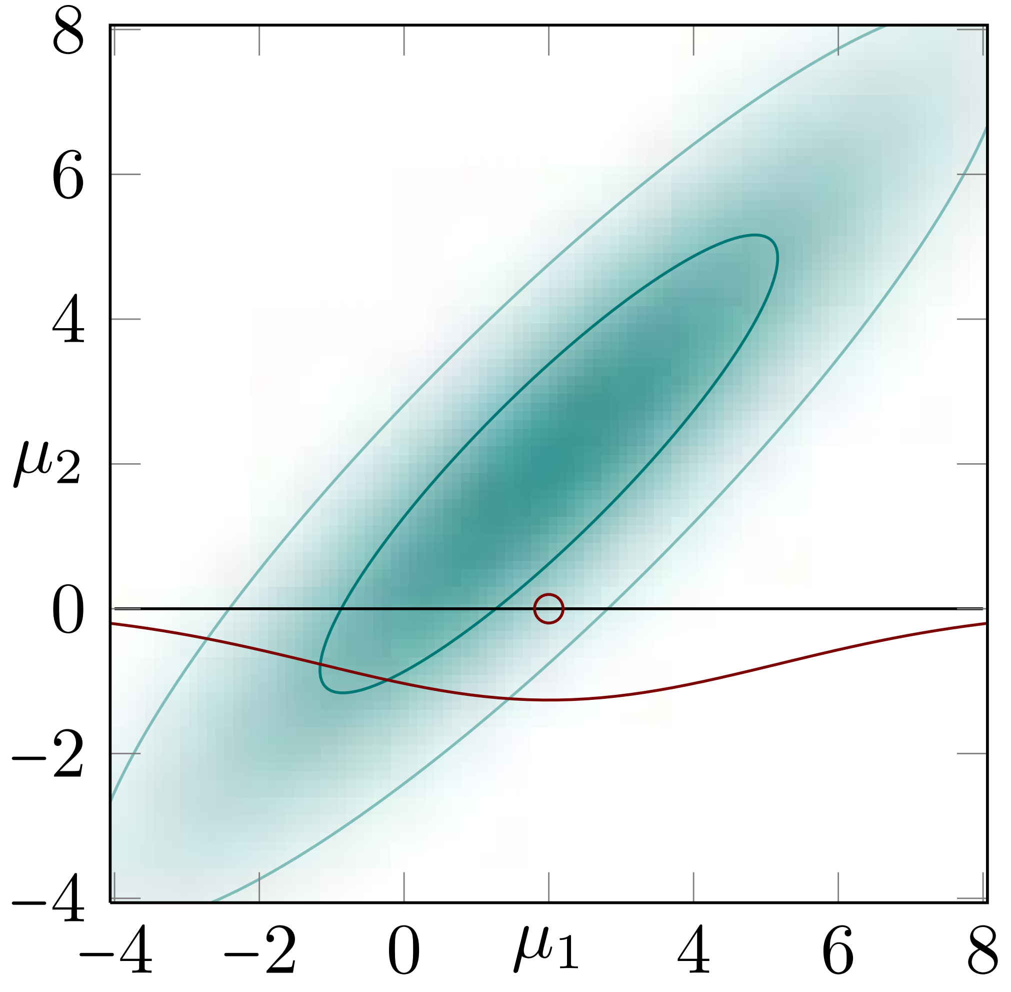
http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

Closure under Marginalization

projections of Gaussians are Gaussian

- projection with $A = \begin{pmatrix} 1 & 0 \end{pmatrix}$

$$\int \mathcal{N}\left[\begin{pmatrix} x \\ y \end{pmatrix}; \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix}\right] \mathrm{d}y = \mathcal{N}(x; \mu_x, \Sigma_{xx})$$



- this is the sum rule

$$\int p(x,y)\,\mathrm{d}y = \int p(y \mid x) p(x)\,\mathrm{d}y = p(x)$$

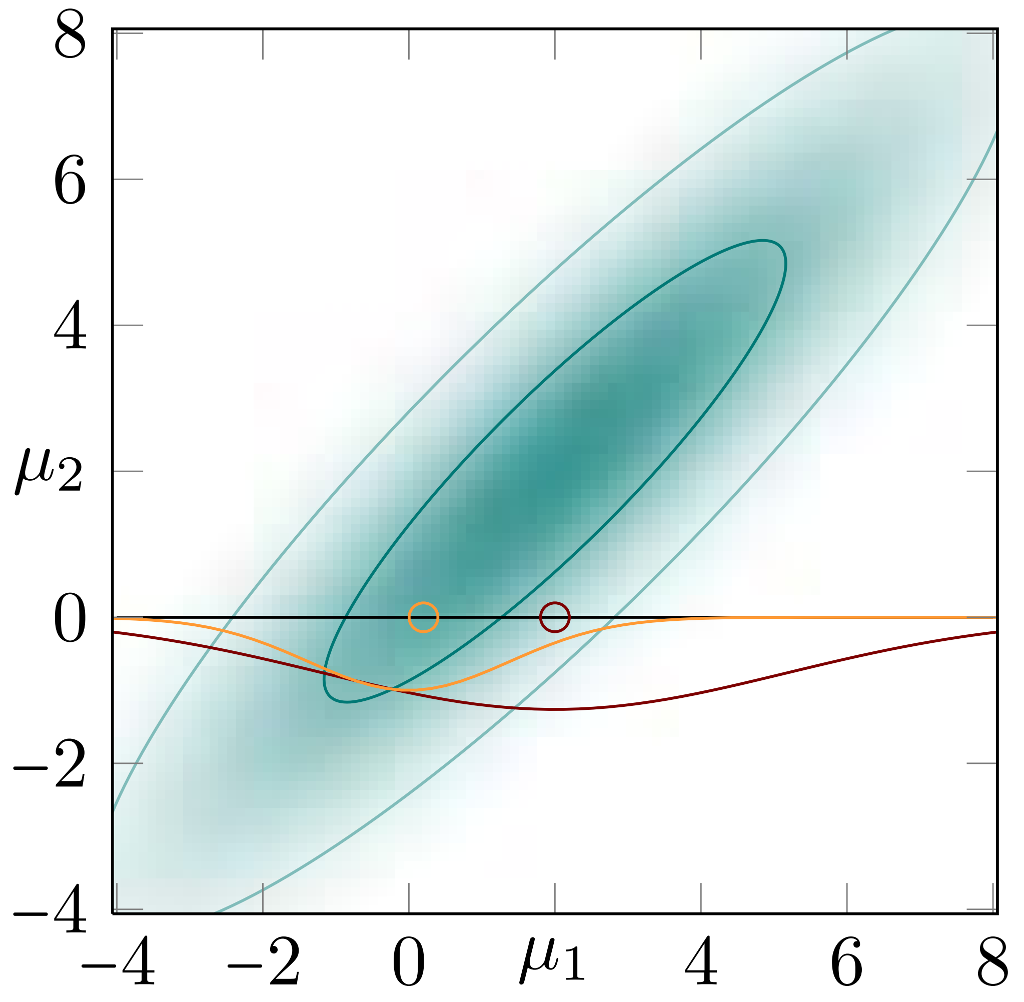
- so every finite-dim Gaussian is a marginal of infinitely many more

http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

Closure under Conditioning

cuts through Gaussians are Gaussians

$$p(x \mid y) = \frac{p(x,y)}{p(y)} = \mathcal{N}\left(x; \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y), \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\right)$$



- this is the product rule
- so Gaussians are closed under the rules of probability

http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

Gaussian process covariance functions (kernels)

$p(f)$ is a Gaussian process if for *any* finite subset $\{x_1, \ldots, x_n\} \subset \mathcal{X}$, the marginal distribution over that finite subset $p(\mathbf{f})$ has a multivariate Gaussian distribution.

Gaussian processes (GPs) are parameterized by a mean function, $\mu(x)$, and a covariance function, or kernel, $K(x, x')$.

$$p(f(x), f(x')) = \mathsf{N}(\mu, \Sigma)$$

where

$$\mu = \begin{bmatrix} \mu(x) \\ \mu(x') \end{bmatrix} \quad \Sigma = \begin{bmatrix} K(x,x) & K(x,x') \\ K(x',x) & K(x',x') \end{bmatrix}$$

and similarly for $p(f(x_1), \ldots, f(x_n))$ where now μ is an $n \times 1$ vector and Σ is an $n \times n$ matrix.

http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Gaussian process covariance functions

Gaussian processes (GPs) are parameterized by a mean function, $\mu(x)$, and a covariance function, $K(x, x')$.

An example covariance function:

$$K(x_i, x_j) = v_0 \exp \left\{ - \left(\frac{|x_i - x_j|}{r} \right)^{\alpha} \right\} + v_1 + v_2 \, \delta_{ij}$$

with parameters $(v_0, v_1, v_2, r, \alpha)$

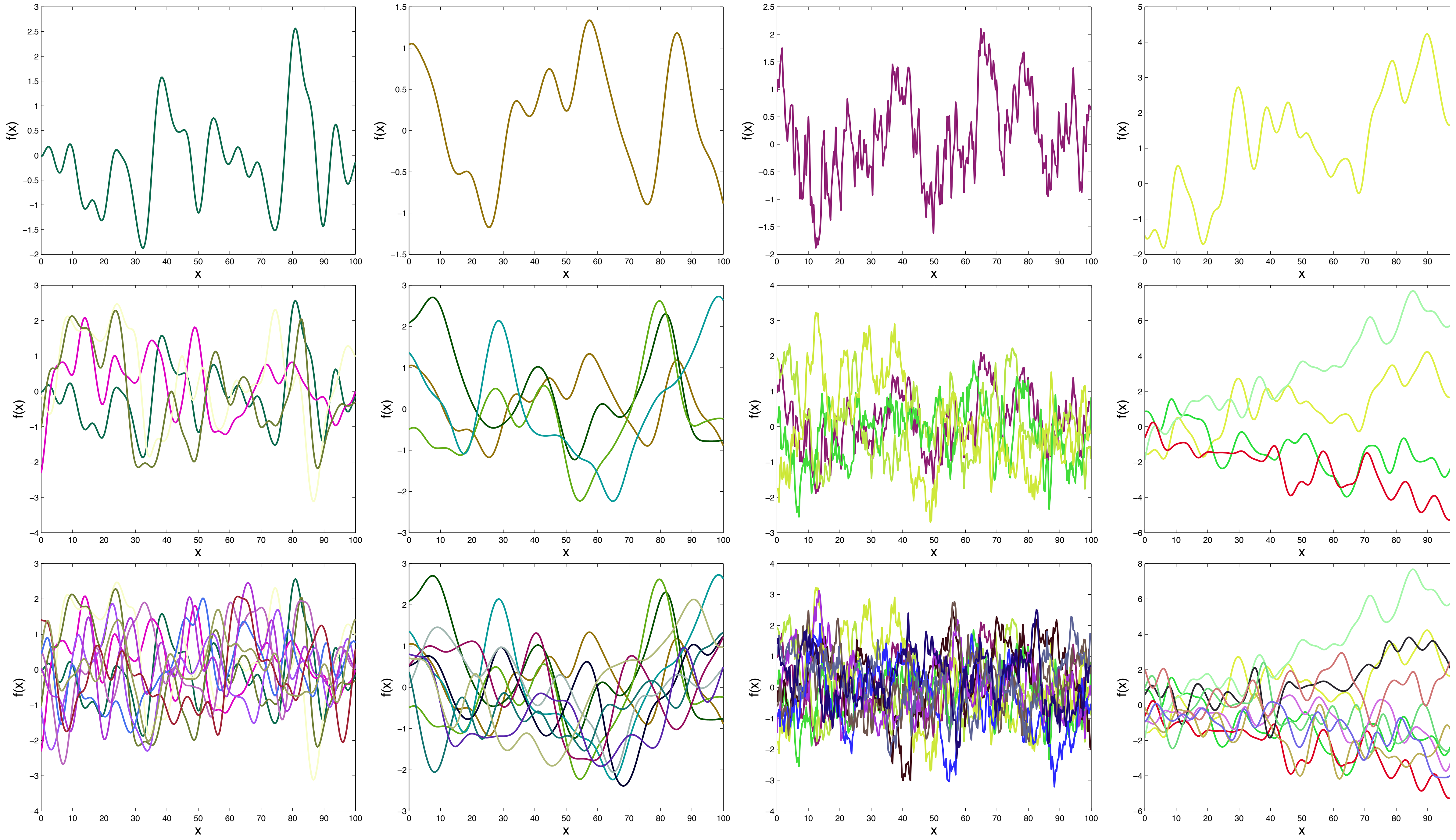
These kernel parameters are interpretable and can be learned from data:

v_0	signal variance
v_1	variance of bias
v_2	noise variance
r	lengthscale
α	roughness

Once the mean and covariance functions are defined, everything else about GPs follows from the basic rules of probability applied to mutivariate Gaussians.

http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

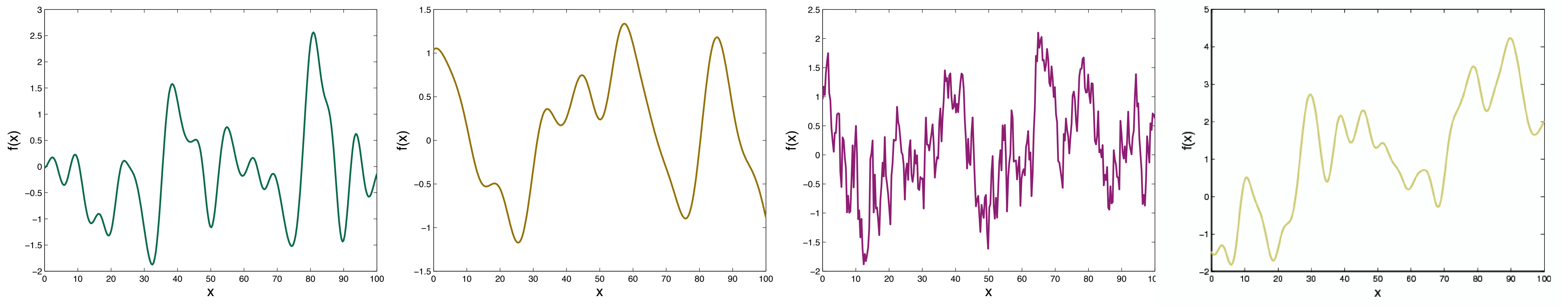
Samples from GPs with different $K(x, x')$



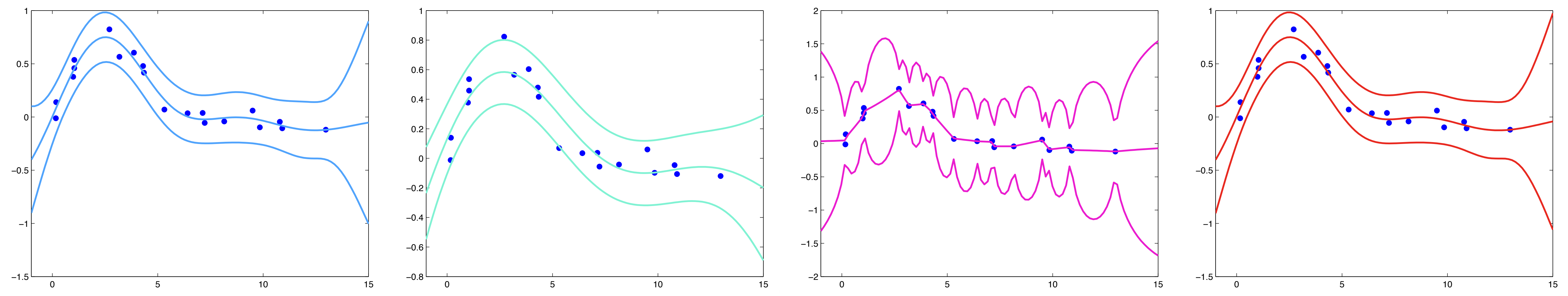
http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Prediction using GPs with different $K(x, x')$

A sample from the prior for each covariance function:



Corresponding predictions, mean with two standard deviations:



http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Using Gaussian processes for nonlinear regression

Imagine observing a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\} = (\mathbf{X}, \mathbf{y})$.

Model:
$$\begin{aligned} y_i &= f(\mathbf{x}_i) + \epsilon_i \\ f &\sim \mathsf{GP}(\cdot|0, K) \\ \epsilon_i &\sim \mathsf{N}(\cdot|0, \sigma^2) \end{aligned}$$

Prior on f is a GP, likelihood is Gaussian, therefore posterior on f is also a GP.

We can use this to make predictions

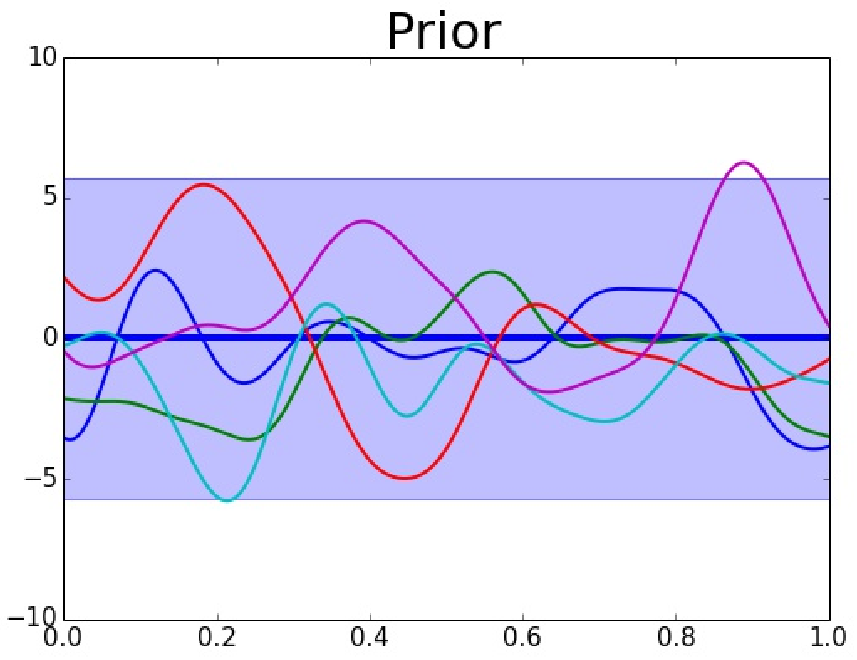
$$p(y_*|\mathbf{x}_*, \mathcal{D}) = \int p(y_*|\mathbf{x}_*, f, \mathcal{D})\, p(f|\mathcal{D})\, df$$

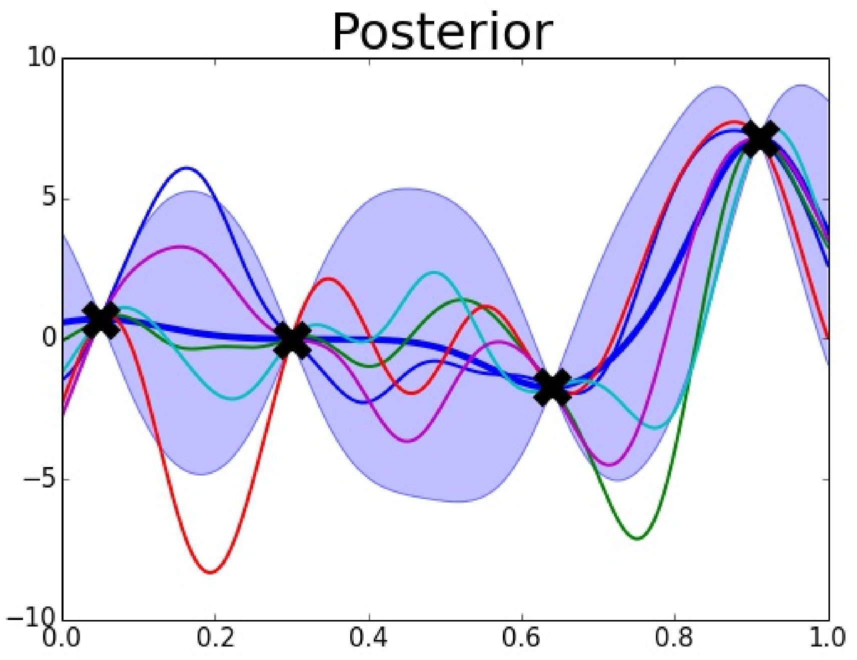
We can also compute the marginal likelihood (evidence) and use this to compare or tune covariance functions

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|f, \mathbf{X})\, p(f)\, df$$

Data-driven modeling with Gaussian processes

$$y = f(\boldsymbol{x}) + \epsilon \qquad f \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}))$$





<u>Training via maximizing the marginal likelihood</u>

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\log|\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I}| - \frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{y} - \frac{N}{2}\log 2\pi$$

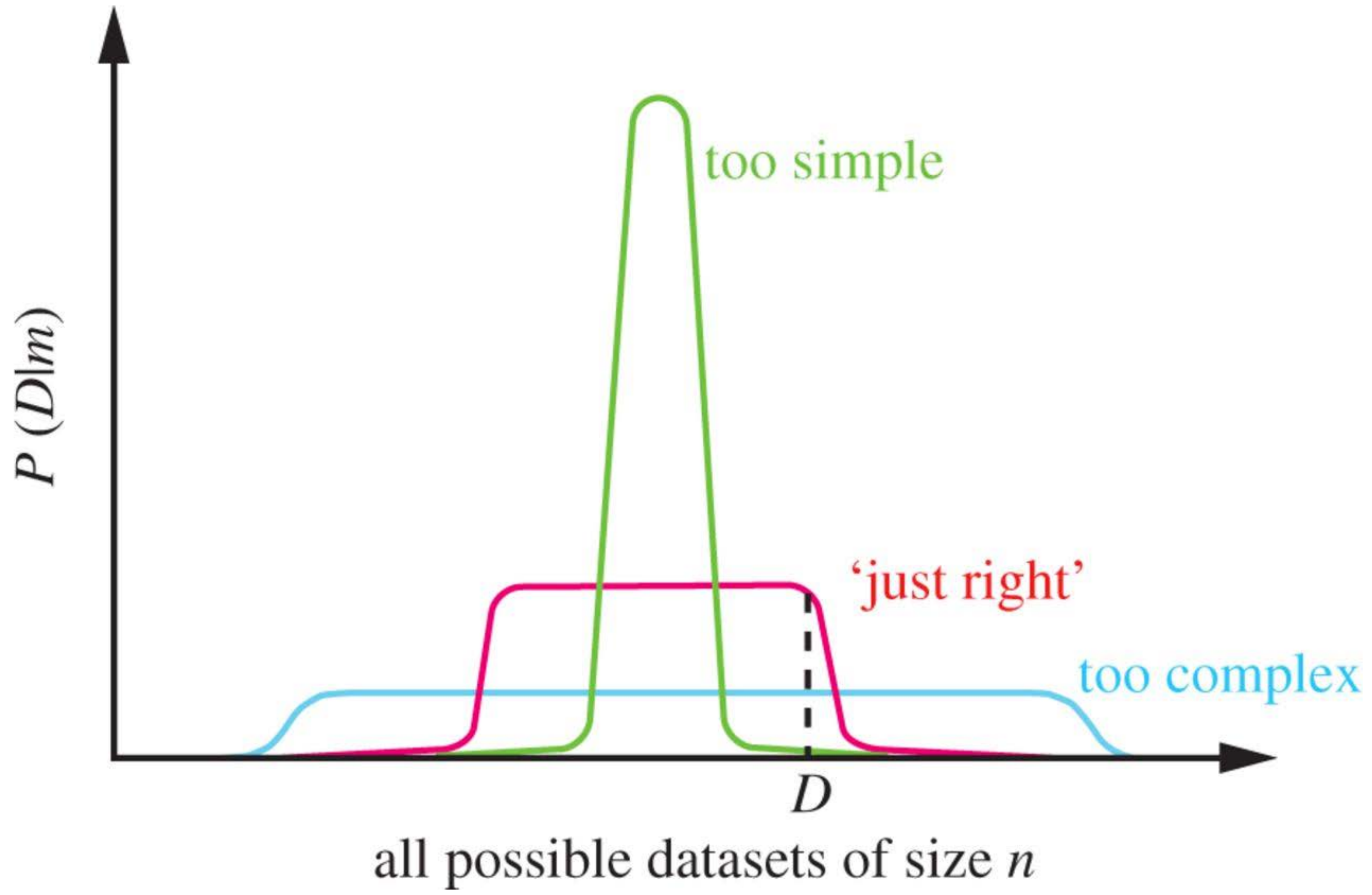
<u>Prediction via conditioning on available data</u>

$$\begin{aligned} p(f_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*) &= \mathcal{N}(f_*|\mu_*, \sigma_*^2), \\ \mu_*(\boldsymbol{x}_*) &= \boldsymbol{k}_{*N}(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{y}, \\ \sigma_*^2(\boldsymbol{x}_*) &= \boldsymbol{k}_{**} - \boldsymbol{k}_{*N}(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{k}_{N*}, \end{aligned}$$

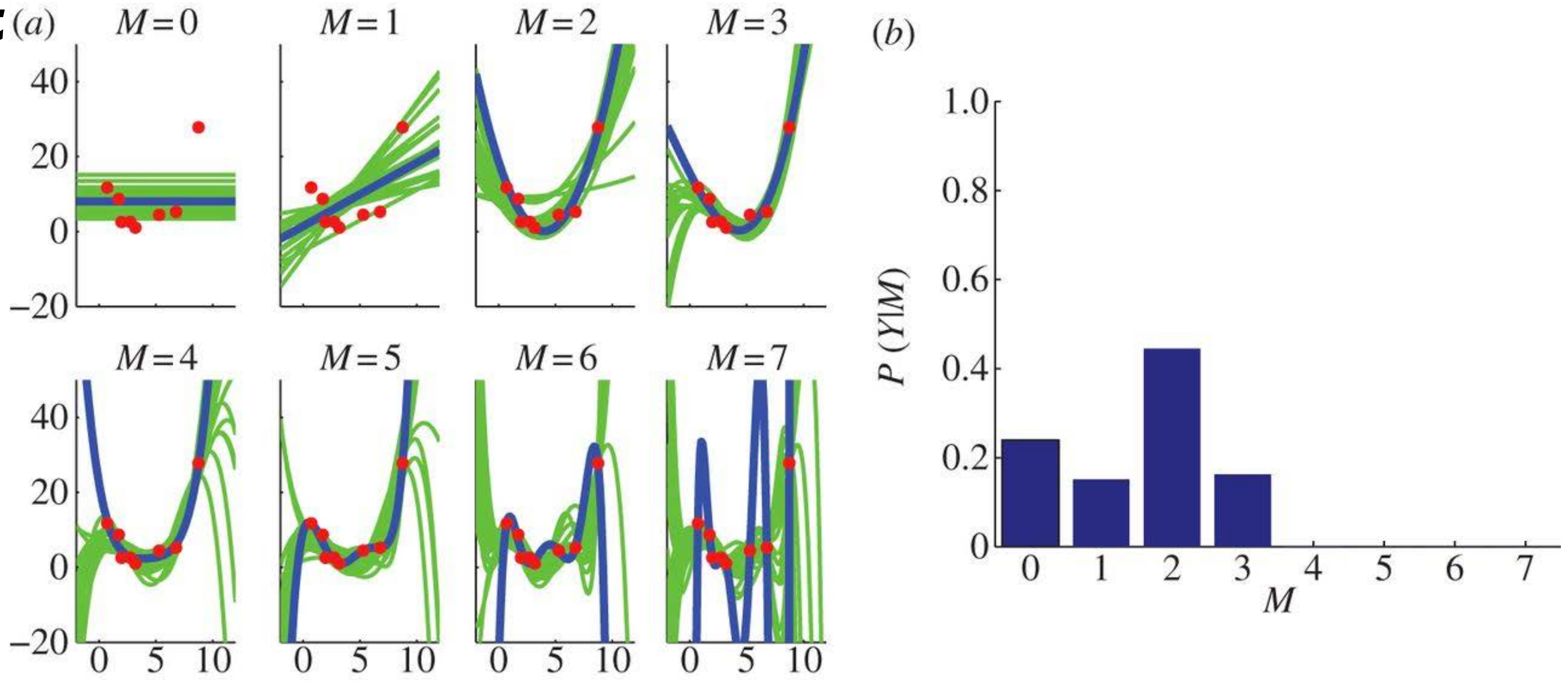
Demo code: https://github.com/PredictiveIntelligenceLab/GPTutorial

Occam's razor

William of Ockham (~1285-1347 A.D)



"plurality should not be posited without necessity."



Ghahramani, Z. (2013). Bayesian non-parametrics and the probabilistic approach to modelling. Phil. Trans. R. Soc. A, 371(1984), 20110553.

MLE vs Marginal MLE

Use MLE when:
- You have lots of data relative to features
- Computational speed is critical
- You don't need uncertainty estimates
- The problem is well-conditioned

Use MLM when it's computable <u>and</u>:
- You have limited data
- You need uncertainty estimates
- You want automatic regularization
- You need to compare different models
- You suspect some features might be irrelevant