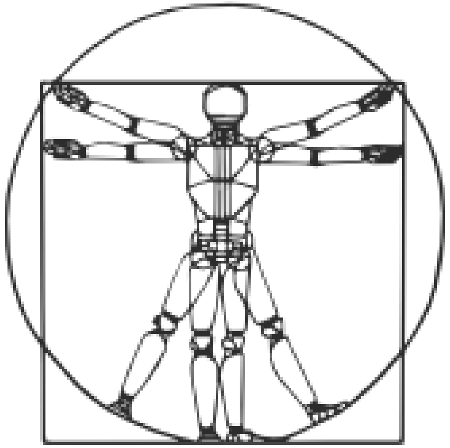
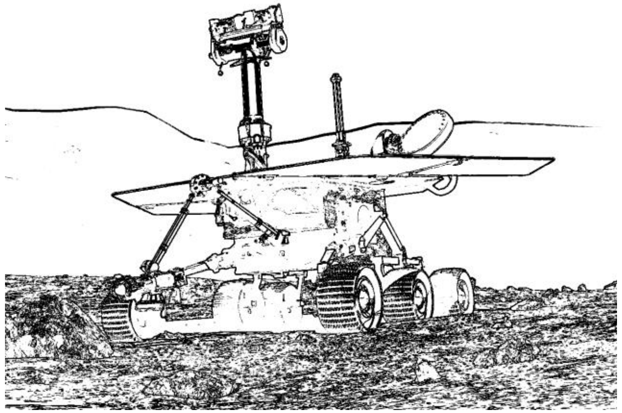




COMP417
Introduction to Robotics and Intelligent Systems

Lecture 22: Function Approximation

Florian Shkurti

Computer Science Ph.D. student

florian@cim.mcgill.ca

McGill

MRL Mobile Robotics Lab at McGill University

Main functions seen in this course

- Dynamics $p(x_t|x_{t-1}, u_{t-1})$

- Measurements $p(z_t|x_t)$

- Controller/Policy $p(u_t|x_t)$

What if we wanted to learn/estimate these functions from data/simulations/experiments?

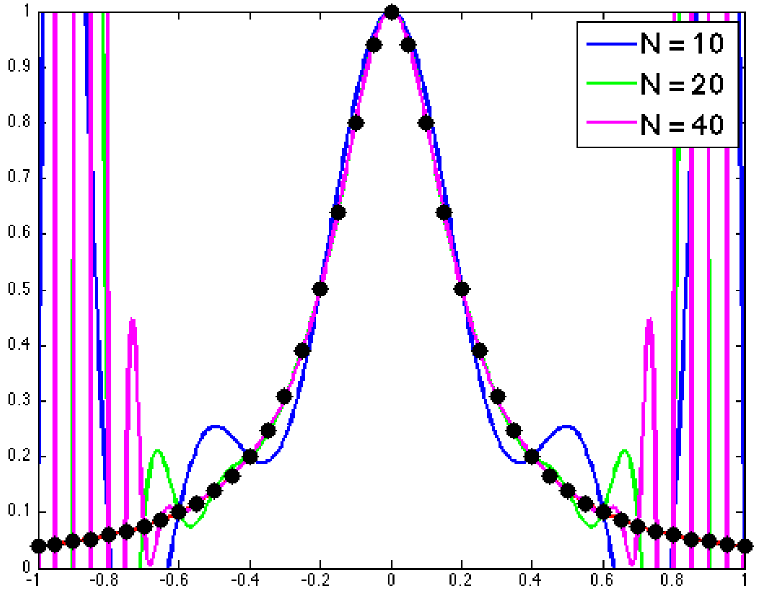
We would need a way to parameterize them and then optimize those parameters based on a cost function.

Parameterizing functions

- Idea # 1: Function is a weighted linear combination of (simple) basis functions
  - Polynomials $$f_{\boldsymbol{\theta}}(x) = \sum_{i=0}^{N} \theta_i x^i = [\theta_0 \ \theta_1 \ \ldots \ \theta_N] \begin{bmatrix} x^0 \\ x^1 \\ \ldots \\ x^N \end{bmatrix}$$

Parameterizing functions

- Idea # 1: Function is a weighted linear combination of (simple) basis functions
  - Polynomials $f_{\boldsymbol{\theta}}(x) = \sum_{i=0}^{N} \theta_i x^i = [\theta_0\ \theta_1\ \ldots\ \theta_N] \begin{bmatrix} x^0 \\ x^1 \\ \ldots \\ x^N \end{bmatrix}$

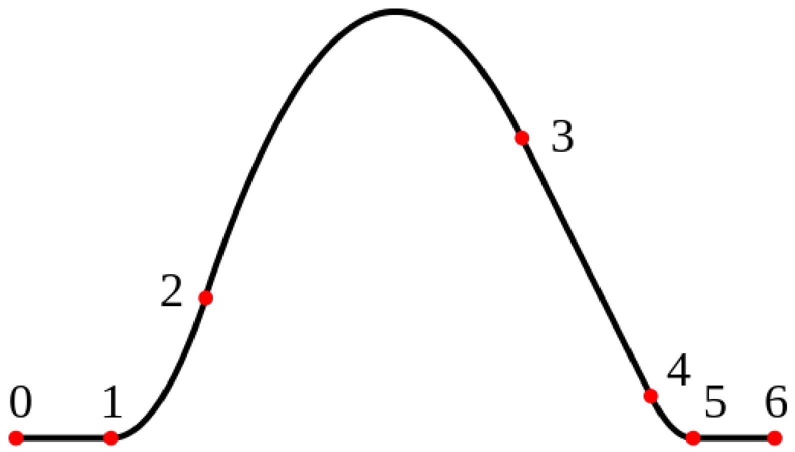


Too much oscillation at the endpoints. Also, small changes to the weights are not necessarily local changes to the shape of the polynomial.

Parameterizing functions

- Idea # 1: Function is a weighted linear combination of (simple) basis functions
  - Splines (Piecewise Polynomials):

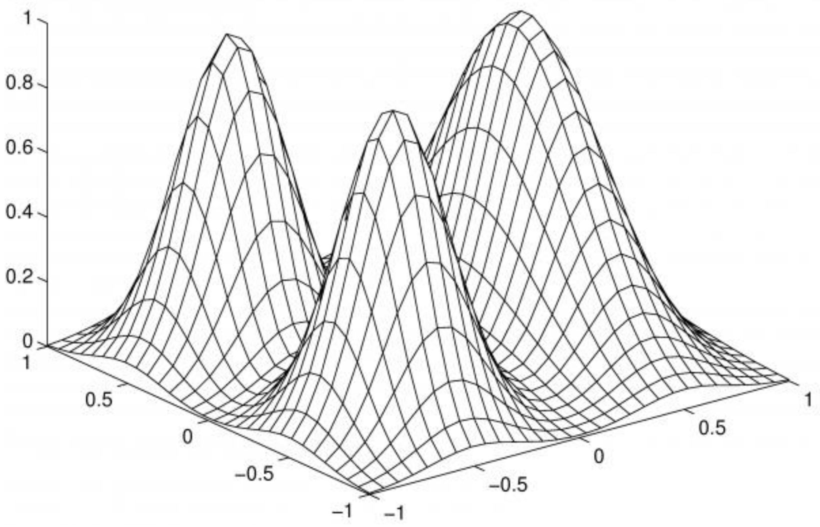
$$f_{\boldsymbol{\theta}^{(k)}}(x) = \sum_{i=0}^{N} \theta_i^{(k)} x^i \quad \text{for} \quad a^{(k)} \leq x < b^{(k)}$$



Parameterizing functions

- Idea # 2: Radial-Basis Functions (RBFs)

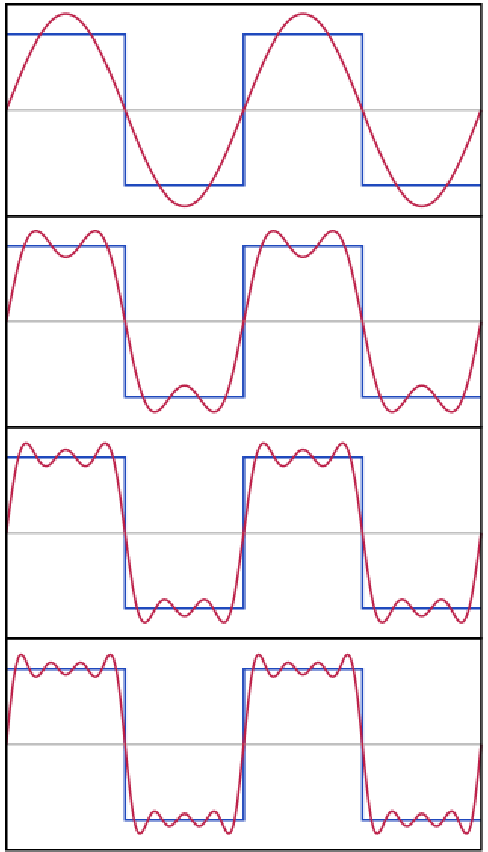
$$f_{\boldsymbol{\theta},\mathbf{c}}(x) = \sum_{i=0}^{N} \theta_i \exp(-||x - c_i||^2) = [\theta_0 \; \theta_1 \; \ldots \; \theta_N] \begin{bmatrix} \exp(-||x - c_0||^2) \\ \exp(-||x - c_1||^2) \\ \ldots \\ \exp(-||x - c_N||^2) \end{bmatrix}$$

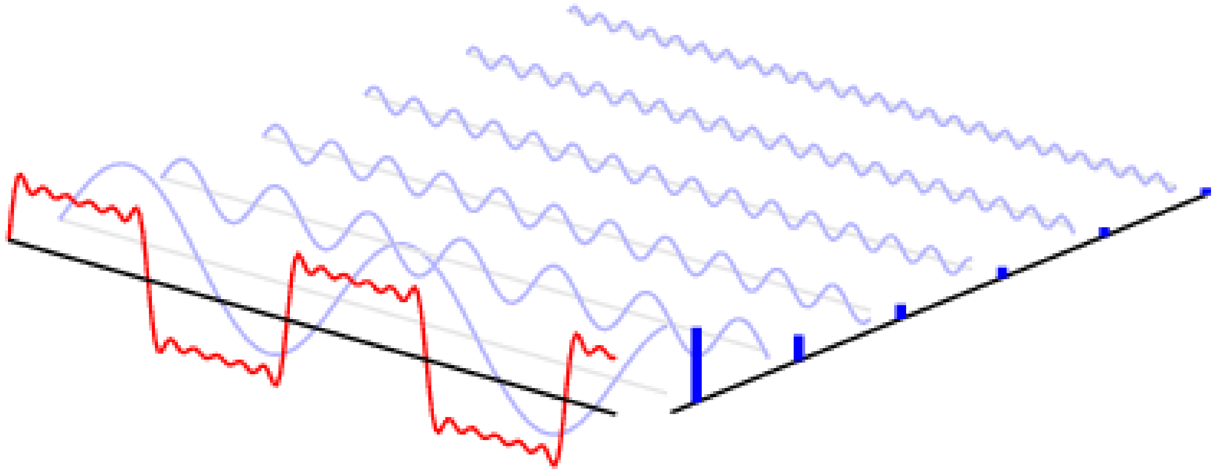


Weighted sums of unnormalized Gaussians

Parameterizing functions

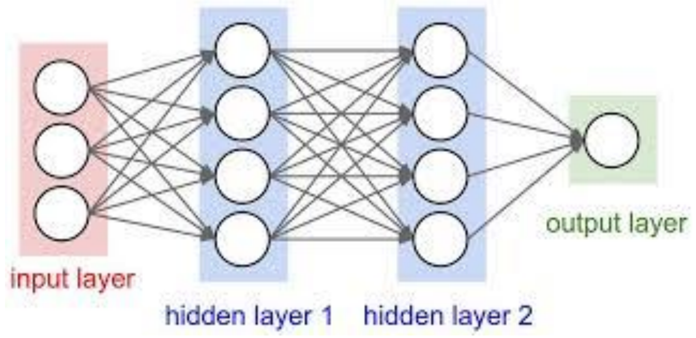
- Idea # 3: Fourier series (for periodic functions) and Fourier transform





Parameterizing functions

- Idea # 4: Multi-Layer Perceptron (MLP, type of neural network)



weight matrices
(an entry per edge)

$W^{(1)}$ 3 x 4 $W^{(2)}$ 4 x 4 $W^{(3)}$ 4 x 1

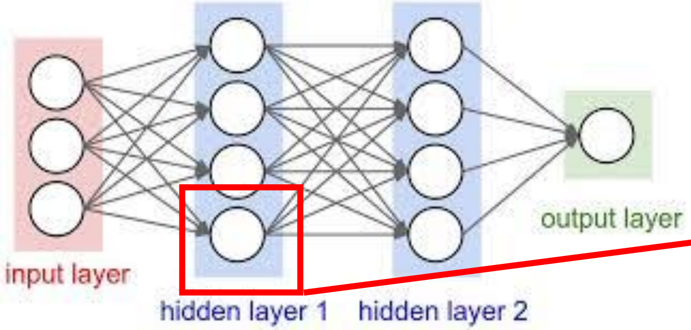
bias vectors
(an entry per node)

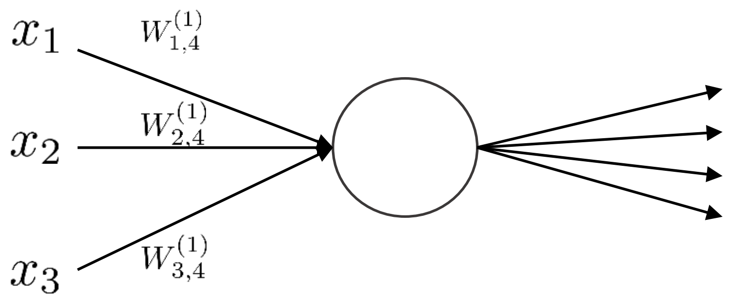
$b^{(1)}$ 4 x 1 $b^{(2)}$ 4 x 1 $b^{(3)}$ 1 x 1

optimizable
parameters

Parameterizing functions

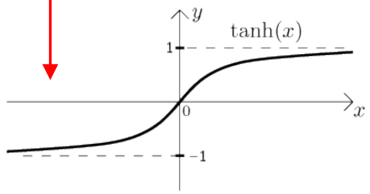
- Idea # 4: Multi-Layer Perceptron (MLP, type of neural network)

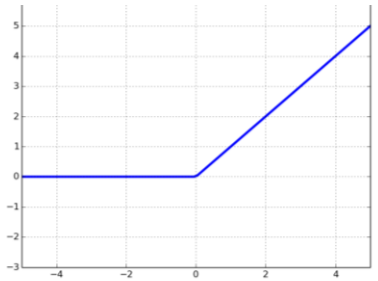




$$\sigma(b^{(1)}_4 + \sum_{i=1}^{3} W^{(1)}_{i,4} x_i)$$

usually a nonlinear activation function





weight matrices
(an entry per edge)

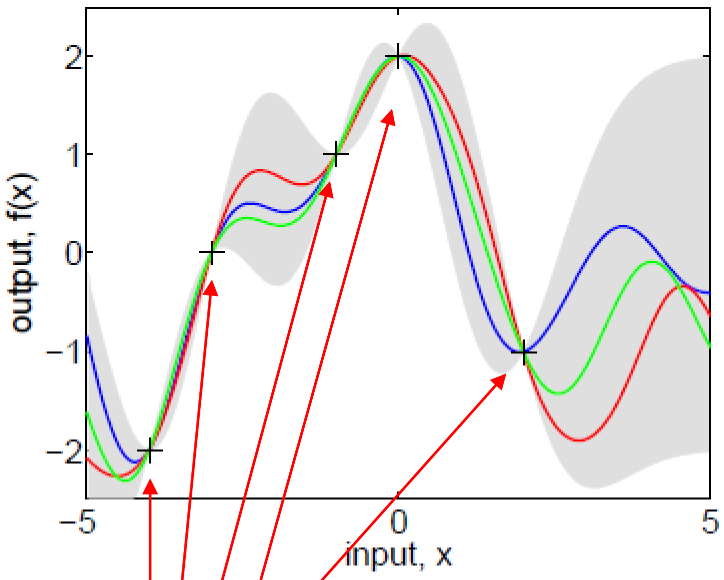
$W^{(1)}$ 3 x 4 $\quad W^{(2)}$ 4 x 4 $\quad W^{(3)}$ 4 x 1

bias vectors
(an entry per node)

$b^{(1)}$ 4 x 1 $\quad b^{(2)}$ 4 x 1 $\quad b^{(3)}$ 1 x 1

Parameterizing functions

- Idea # 5: Gaussian Processes

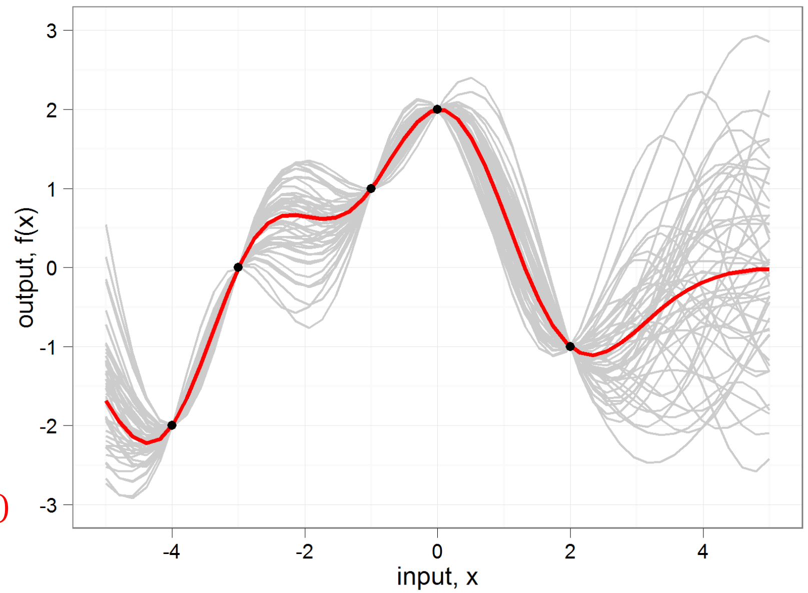


Training set (X, Y)

Not a single function, but a
distribution over functions

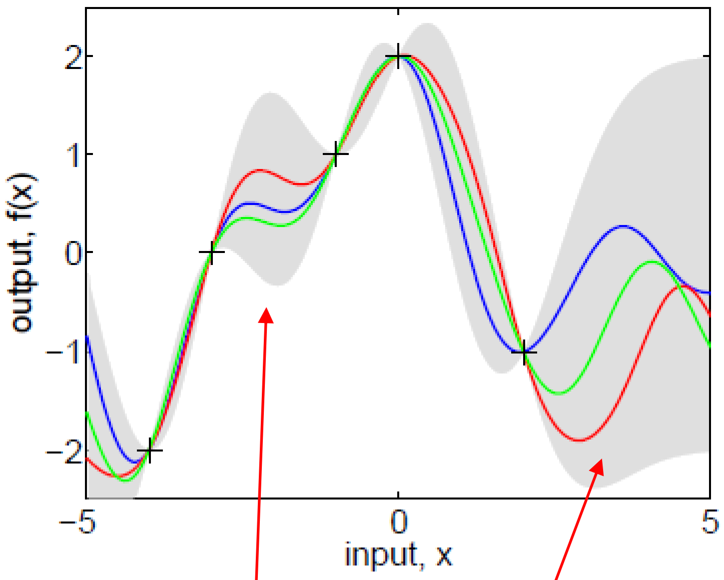
$$f(x) \sim \mathcal{N}(0, K(\theta, x, x'))$$

Covariance at x determined by
distance of point x to all other
points x' in the training set (X, Y)



Parameterizing functions

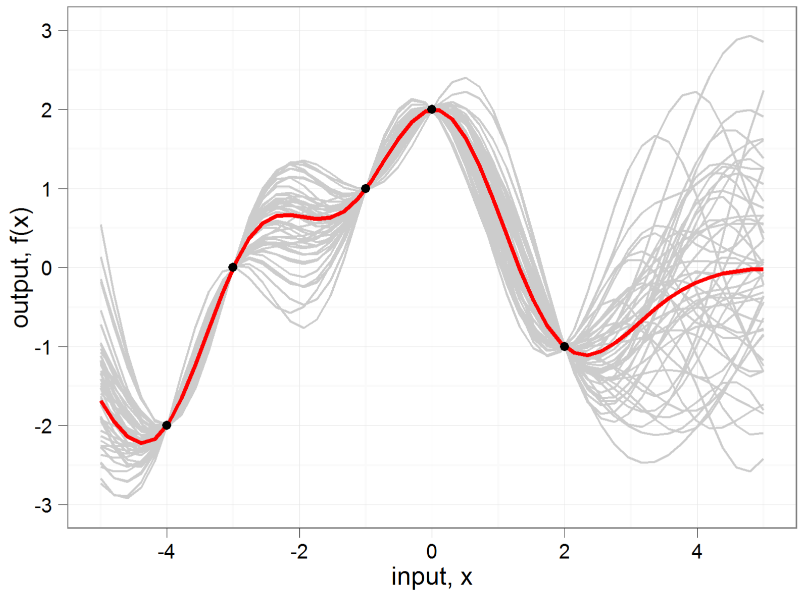
- Idea # 5: Gaussian Processes



Not a single function, but a **distribution over functions**

$$f(x) \sim \mathcal{N}(0, K(\theta, x, x'))$$

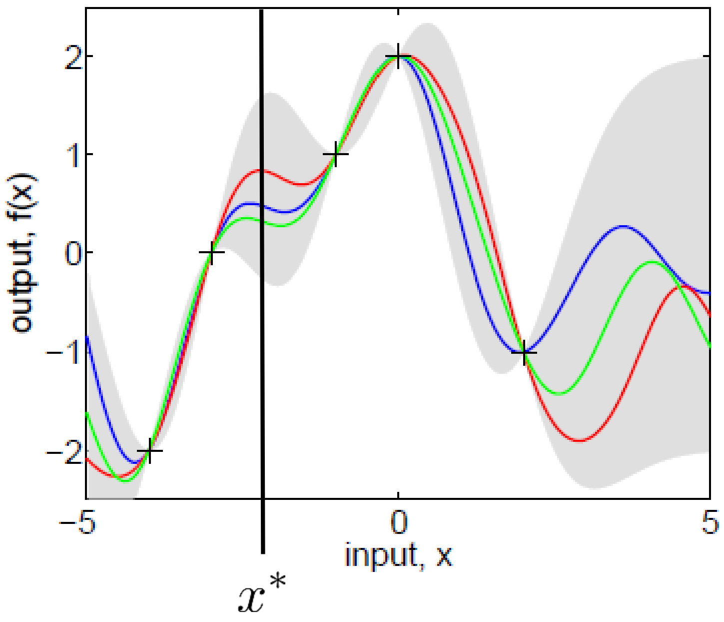
The average function is zero



High uncertainty where there are no training data

Parameterizing functions

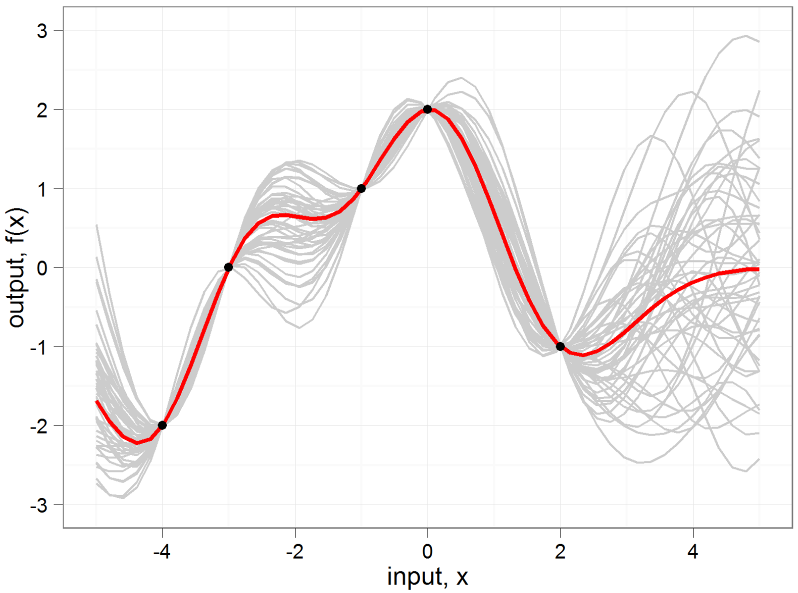
- Idea # 5: Gaussian Processes



New data point
(to be predicted)

Not a single function, but a
distribution over functions

$$f(x) \sim \mathcal{N}(0, K(\theta, x, x'))$$



Predicting the value of the function at an unseen point: $p(f^*|x^*, X, Y, \theta)$

Main drawback for GP prediction: involves matrix multiplication of matrices the size of the training set.
So, naïve implementations of GPs are limited to small datasets.