Nonlinear Regression – Part II

Interactive lecture

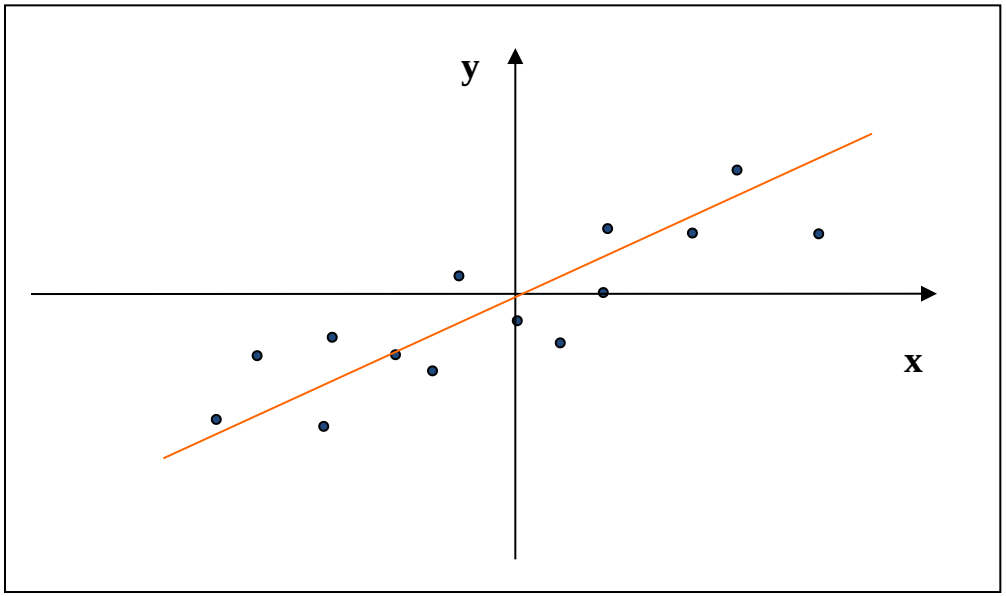
Gaussian Process Regression
&
Extensions

GP for Classification

GP for Manifold Learning - GPLVM

Linear Regression

$$y = f(x; w) = w^T x$$



It has an exact solution $w^* = \left(XX^T\right)^{-1} X\mathbf{y}$ if:

a) XX^T is not singular (it is singular with not enough datapoints)

b) Data is not noisy (otherwise no single match to $y^i = \langle w, x^i \rangle$)

Ridge Regression: computational costs

Ridge Regression computational cost grow primarily as a function of:

A. Number of datapoints
B. Dimension of the datapoints

Solution in linear case:

$$w^* = \left(XX^T + \lambda I\right)^{-1} X\mathbf{y}$$

always invertible for $\lambda > 0$.

Solution in nonlinear case:

$$y = k(X,x)\left(\underbrace{K(X,X)}_{\text{Gram Matrix in feature space}} + \lambda I\right)^{-1}\mathbf{y}, \quad k(X,x) = \begin{bmatrix} k(x^1,x) \\ \cdot \\ \cdot \\ k(x^M,x) \end{bmatrix}^T$$

In linear ridge regression, the complexity is $O(N^3)$, N: dimension of datapoint

In nonlinear ridge regression, the complexity is: $O(M^3)$, M: number of datapoints

Ridge Regression: optimality

Is the Ridge Regression optimum unique?

A. Yes
B. No

Solution in linear case:

$$w^* = \left(XX^T + \lambda I\right)^{-1} X\mathbf{y}$$

always invertible for $\lambda > 0$.

Solution in nonlinear case:

$$y = k(X,x)\left(\underbrace{K(X,X)}_{\text{Gram Matrix in feature space}} + \lambda I\right)^{-1}\mathbf{y}, \quad k(X,x) = \begin{bmatrix} k(x^1,x) \\ \cdot \\ \cdot \\ k(x^M,x) \end{bmatrix}^T$$

It is unique if λ is fixed but optimality depends on λ.

Linear Probabilistic Regression (PR) vs GPR

Which is true?

A. PR only has a closed-form solution
B. GPR only has a closed-form solution
C. Both have closed-form solutions ✓

PR estimates y given a test point x^* :

$$y^* = E\left\{ p(y \mid x^*, X, \mathbf{y}) \right\} = \frac{1}{\sigma^2} x^{*T} A^{-1} X\mathbf{y}$$

GPR estimates y given a testing point x^*

$$y = \sum_{i=1}^{M} \alpha_i k\left(x^*, x^i\right)$$

with $\alpha = \left[K(X, X) + \sigma_\varepsilon{}^2 I \right]^{-1} \mathbf{y}$

Linear Probabilistic Regression (PR) vs GPR

Which is computationally most costly?

A. PR is the most costly.

B. GPR is the most costly.

C. Both are as costly.

PR estimates y given a test point x^* :

$$y^* = E\left\{ p(y \mid x^*, X, \mathbf{y}) \right\} = \frac{1}{\sigma^2} x^{*T} A^{-1} X\mathbf{y}$$

In both the complexity is:

$O\left(M^3\right)$, M : number of datapoints

GPR estimates y given a testing point x^*

$$y = \sum_{i=1}^{M} \alpha_i k\left(x^*, x^i\right)$$

with $\alpha = \left[K\left(X, X\right) + \sigma_\varepsilon{}^2 I \right]^{-1} \mathbf{y}$

GPR requires to compute a series of exponentials which is somewhat more demanding

GPR and Probabilistic Regression

Ridge Regression

Linear case:

$$w^* = A^{-1} X \mathbf{y}$$

$$A = \left(XX^T + \lambda I \right)^{-1}$$

Nonlinear case:

$$y = \sum_{i=1}^{M} \alpha_i k\left(x^i, x\right)$$

$$\alpha = \left[K\left(X, X\right) + \lambda I \right]^{-1} \mathbf{y}$$

Probabilistic Regression

Linear case:

$$w^* = \mathbf{A}^{-1} \Sigma_{\mathbf{w}}^{-1} \mu_{\mathbf{w}} + \frac{1}{\sigma_\epsilon^2} \mathbf{A}^{-1} \mathbf{X} \mathbf{y}$$

$$A = \frac{1}{\sigma_\epsilon^2} \mathbf{X}\mathbf{X}^T + \Sigma_{\mathbf{w}}^{-1}$$

Nonlinear case (GPR):

$$y = \sum_{i=1}^{M} \alpha_i k\left(x, x^i\right)$$

with $\alpha = \left[K\left(X, X\right) + \sigma^2 I \right]^{-1} \mathbf{y}$

Ridge regression and GPR end up with similar expression for the nonlinear regressor. GPR provides a model of the full density (with variance) and has a method to estimate the hyperparameters

GPR and Probabilistic Regression

Ridge Regression

Linear case:

$$w^* = A^{-1} X\mathbf{y}$$

$$A = \left(XX^T + \lambda I \right)^{-1}$$

Nonlinear case:

$$y = \sum_{i=1}^{M} \alpha_i k\left(x^i, x\right)$$

$$\alpha = \left[K\left(X, X\right) + \lambda I \right]^{-1} \mathbf{y}$$

Probabilistic Regression

Linear case:

$$w^* = \mathbf{A}^{-1}\Sigma_{\mathbf{w}}^{-1}\mu_{\mathbf{w}} + \frac{1}{\sigma_\epsilon^2}\mathbf{A}^{-1}\mathbf{X}\mathbf{y}$$

$$A = \frac{1}{\sigma_\epsilon^2}\mathbf{X}\mathbf{X}^T + \Sigma_{\mathbf{w}}^{-1}$$

Nonlinear case (GPR):

$$y = \sum_{i=1}^{M} \alpha_i k\left(x, x^i\right)$$

with $\alpha = \left[K\left(X, X\right) + \sigma^2 I \right]^{-1} \mathbf{y}$

In the special case when the distribution is zero mean and $\Sigma_w = \tau I$, probabilistic regression reduces to ridge regression with $\lambda = \frac{\sigma_\epsilon^2}{\tau}$.

Gaussian Process Regression (GPR)

Kernel function

$$y = E\{y \mid x, X, \mathbf{y}\} = \sum_{i=1}^{M} \alpha_i k\left(x, x^i\right)$$

$$\text{with } \alpha = \left[K(X, X) + \sigma^2 I\right]^{-1} \mathbf{y}$$

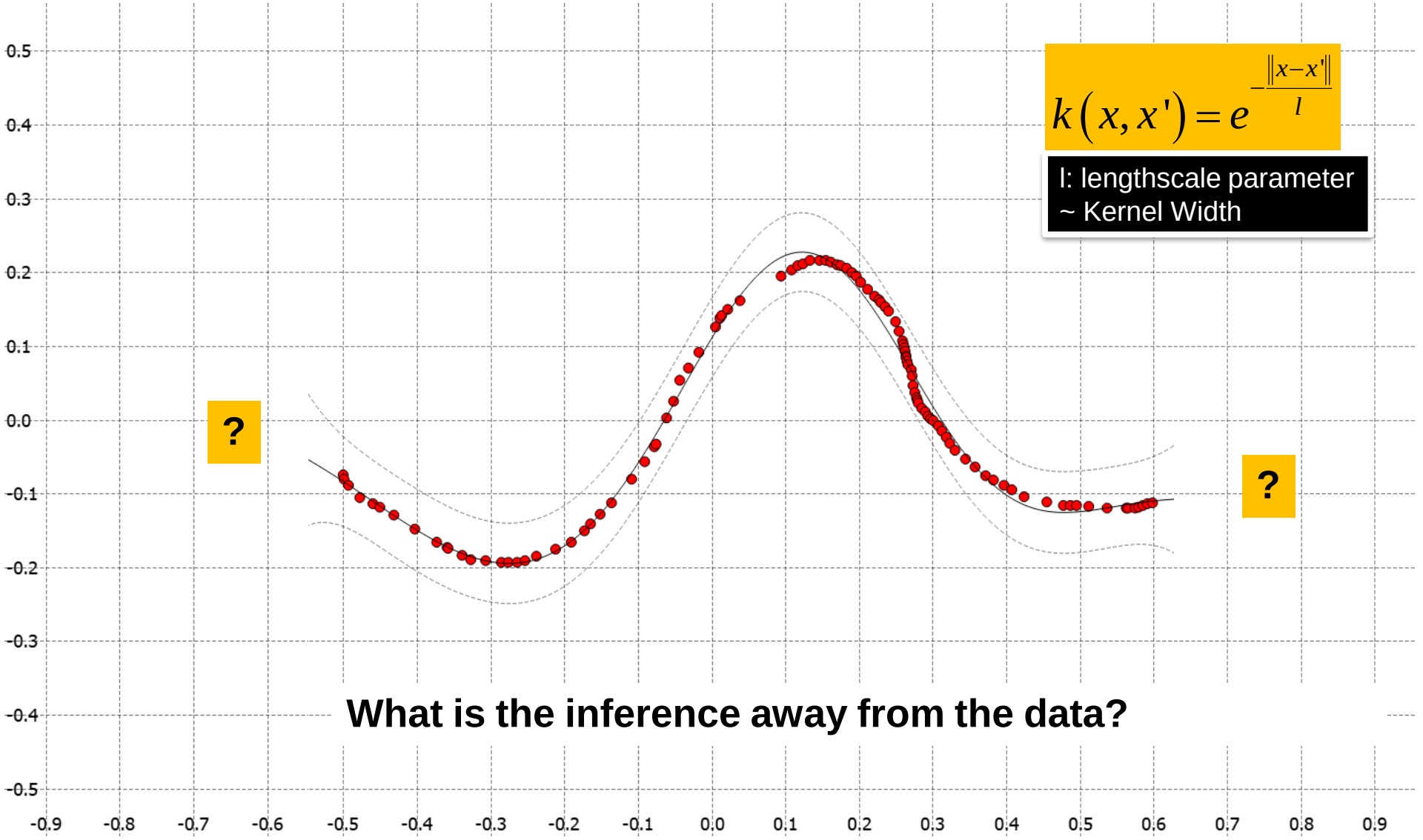
Linear: $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}' + c$ where c is the intercept

RBF: $k(\mathbf{x}, \mathbf{x}') = \exp - \frac{\| \mathbf{x} - \mathbf{x}' \|^2}{\ell}$ where ℓ is the kernel width (lengthscale)

Polynomial: $k(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T \mathbf{x}' + c\right)^p$ where p is the degree of polynomial

Periodic: $k(\mathbf{x}, \mathbf{x}') = \exp - \frac{2 \sin^2(\nu \| \mathbf{x} - \mathbf{x}' \| / T)}{\ell}$ where T is the period.

GPR – RBF Kernel - Inference

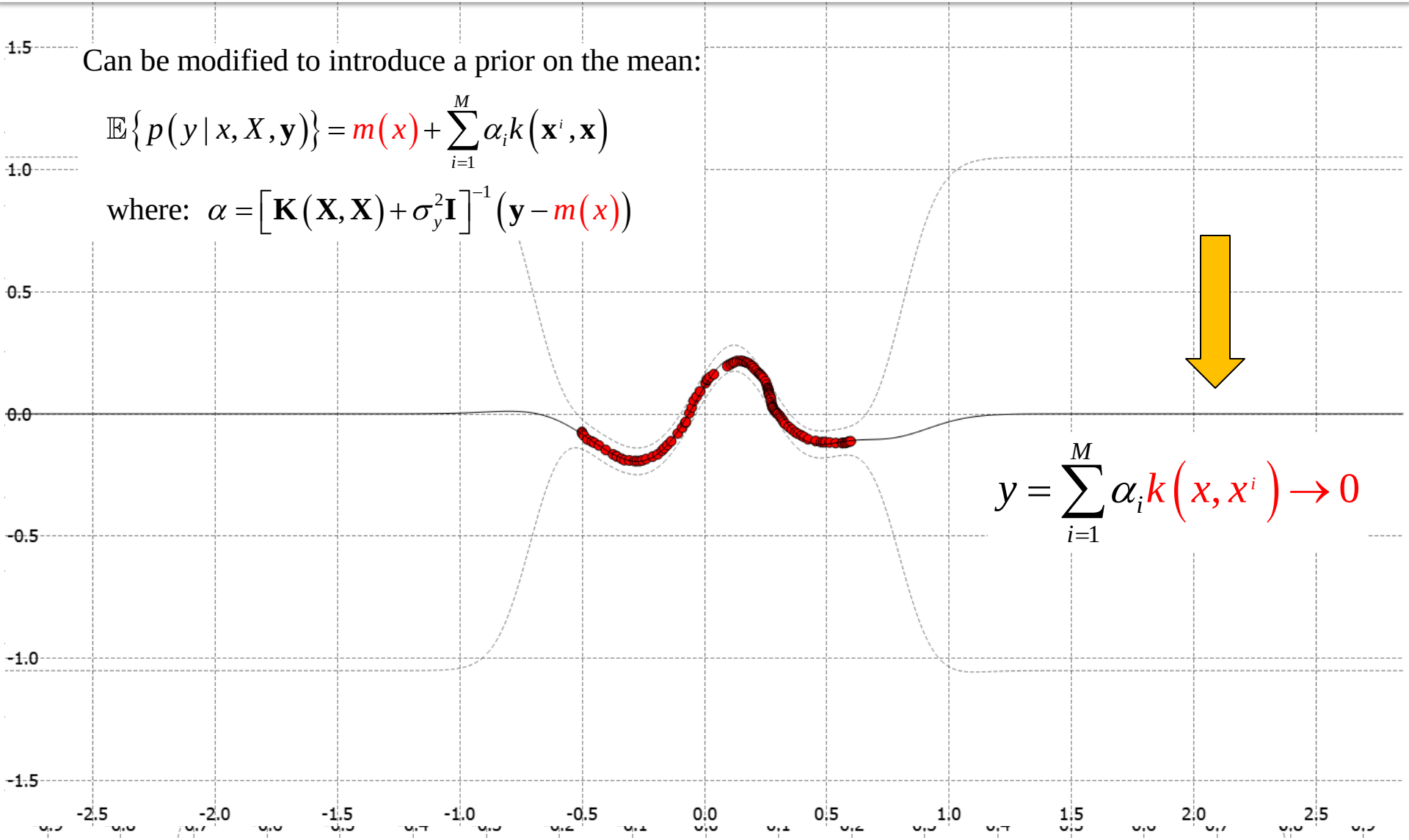


Inference away from the data – RBF kernel

Can be modified to introduce a prior on the mean:

$$\mathbb{E}\left\{p\left(y \mid x, X, \mathbf{y}\right)\right\} = m\left(x\right) + \sum_{i=1}^{M} \alpha_i k\left(\mathbf{x}^i, \mathbf{x}\right)$$

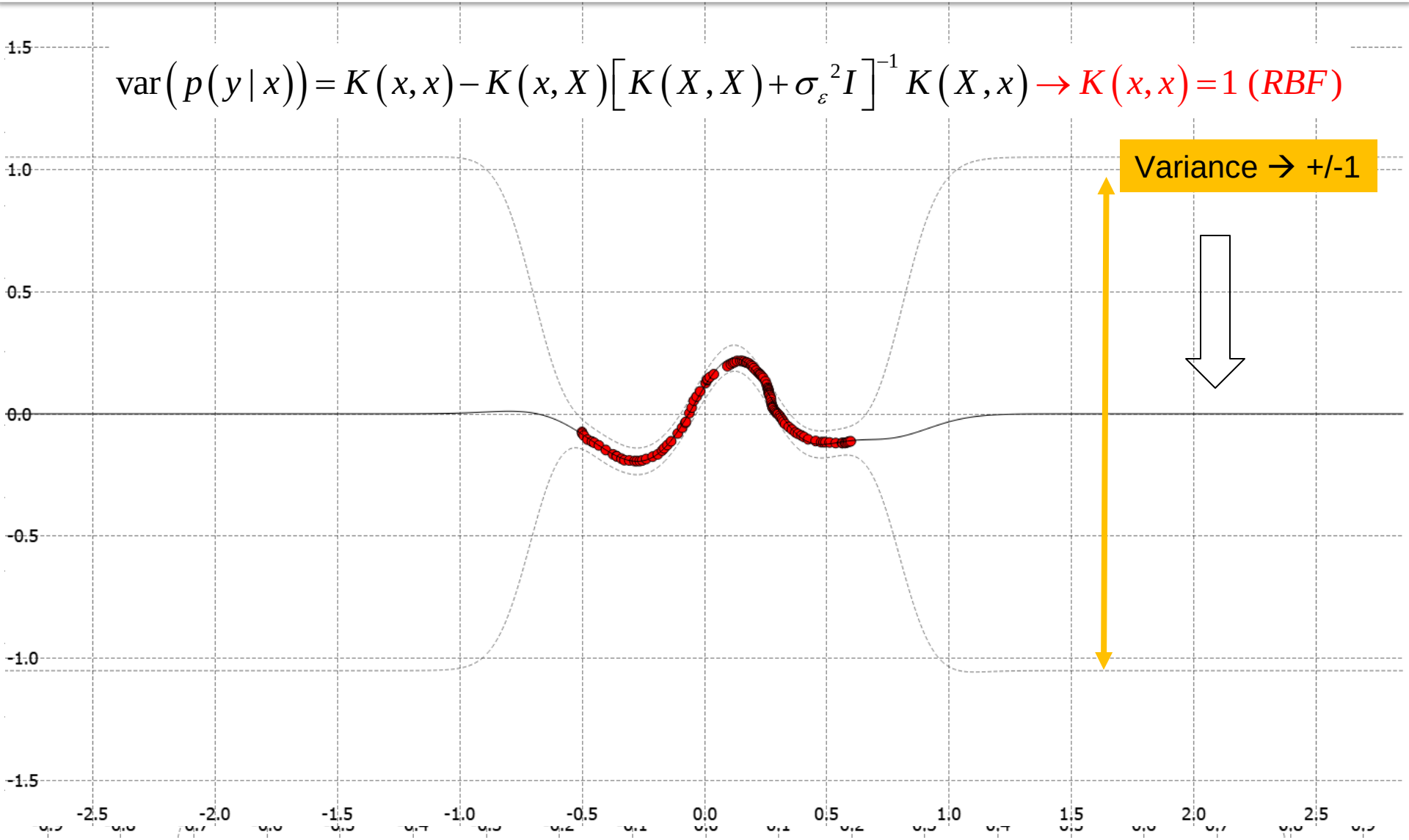
where: $\alpha = \left[\mathbf{K}\left(\mathbf{X}, \mathbf{X}\right) + \sigma_y^2 \mathbf{I}\right]^{-1} \left(\mathbf{y} - m\left(x\right)\right)$



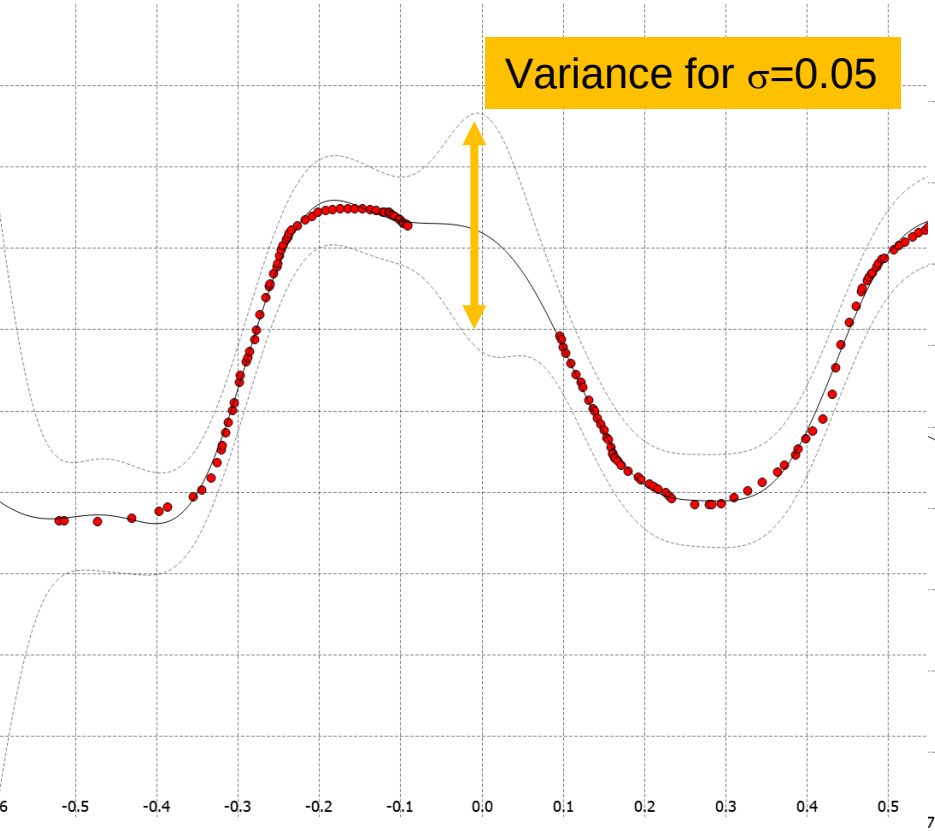
$$y = \sum_{i=1}^{M} \alpha_i k\left(x, x^i\right) \to 0$$

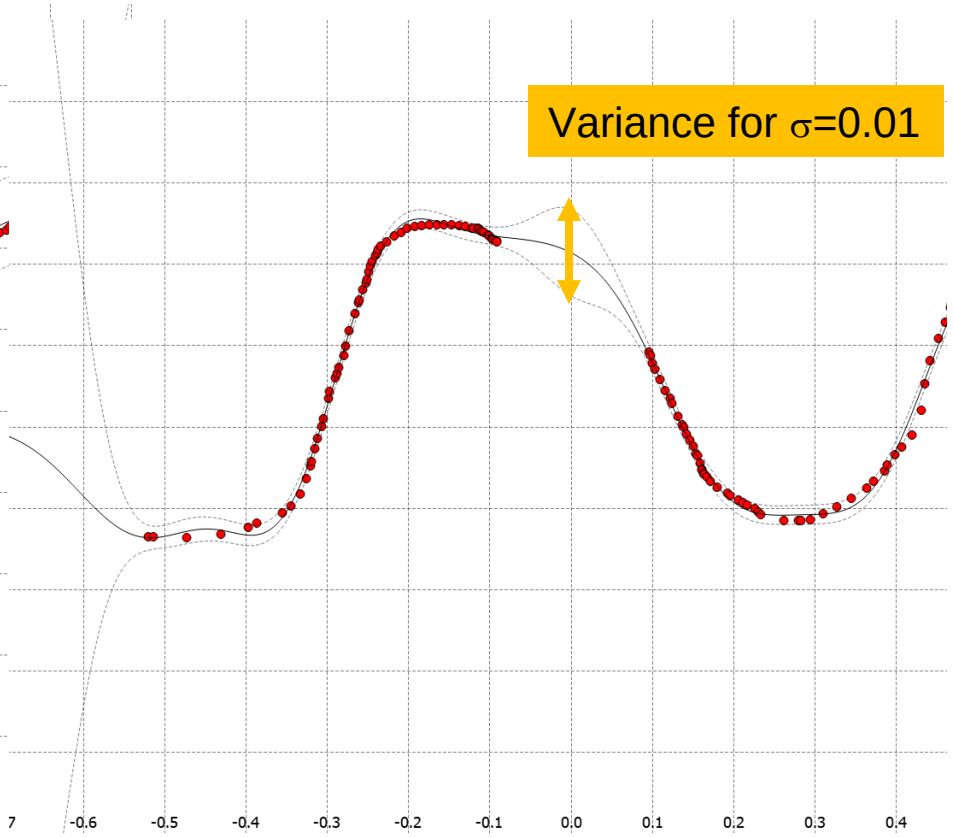
Inference away from the data – RBF kernel

$$\mathrm{var}\left(p\left(y \mid x\right)\right) = K\left(x,x\right) - K\left(x,X\right)\left[K\left(X,X\right) + \sigma_{\varepsilon}{}^{2} I\right]^{-1} K\left(X,x\right) \rightarrow K\left(x,x\right) = 1 \; (RBF)$$

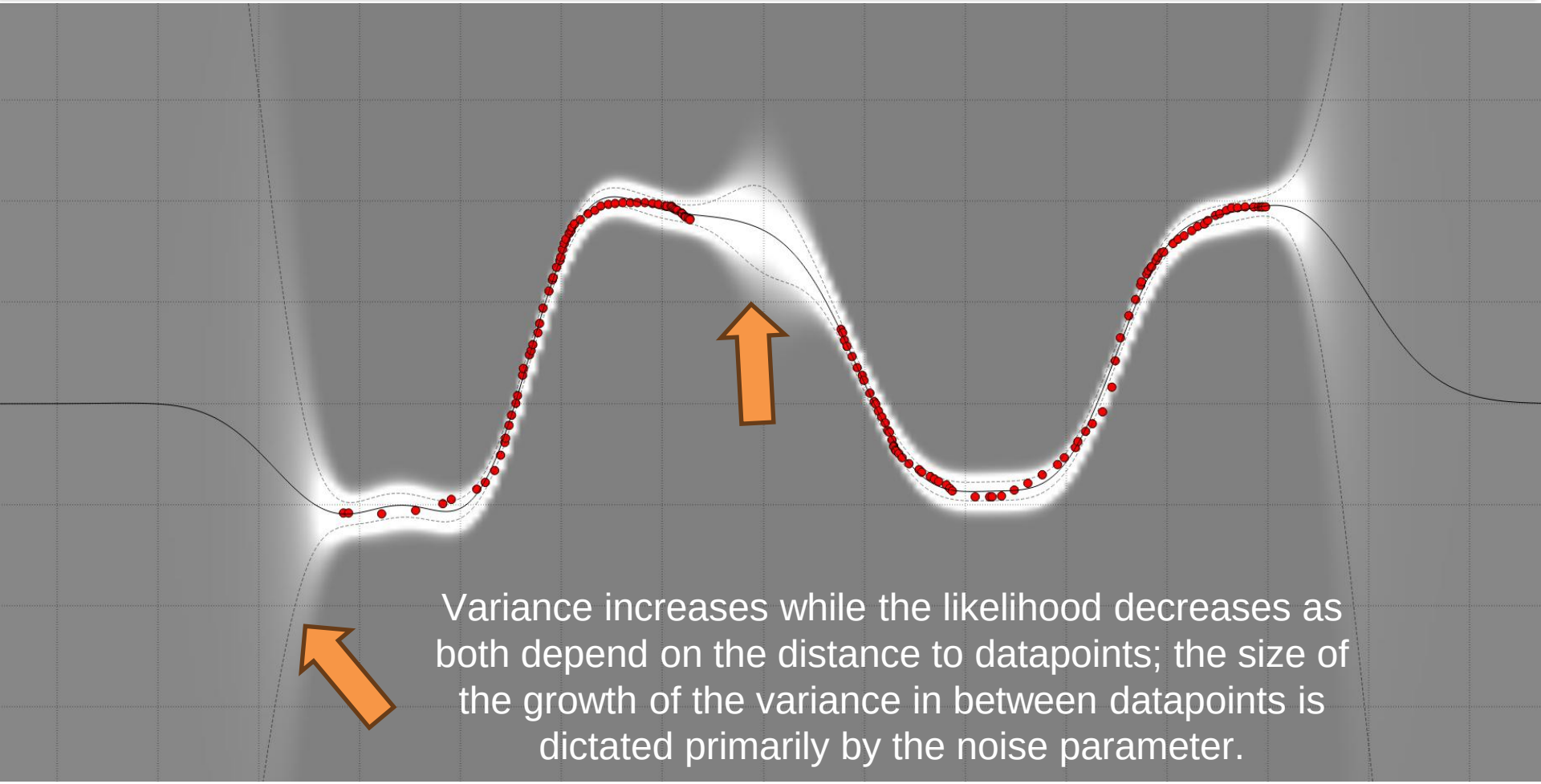


$$\mathrm{var}\left(p(y \mid x)\right) = K(x,x) - K(x,X)\left[K(X,X) + \sigma_\varepsilon{}^2 I\right]^{-1} K(X,x)$$





Variance versus Likelihood



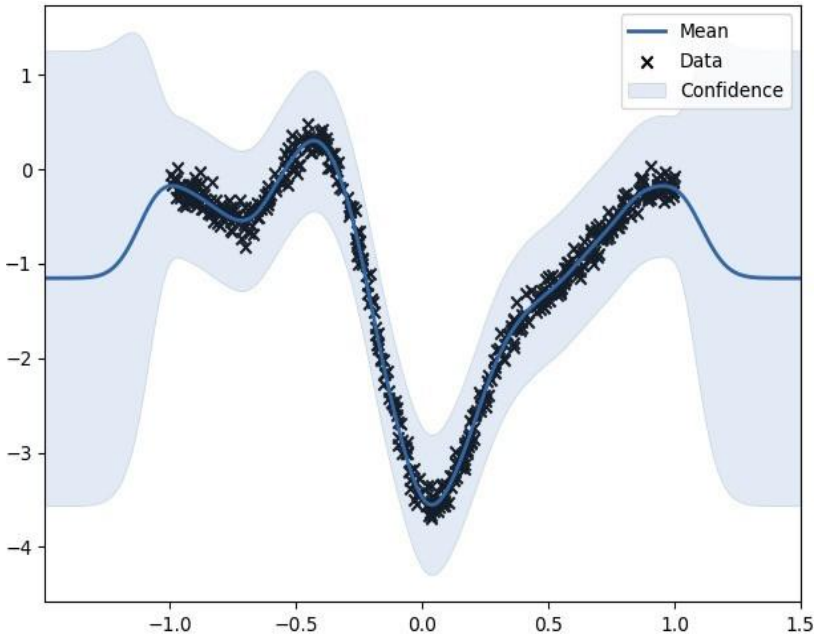
Variance increases while the likelihood decreases as both depend on the distance to datapoints; the size of the growth of the variance in between datapoints is dictated primarily by the noise parameter.

GPR with RBF – Effect of kernel width

Which plot corresponds to the smallest kernel width?

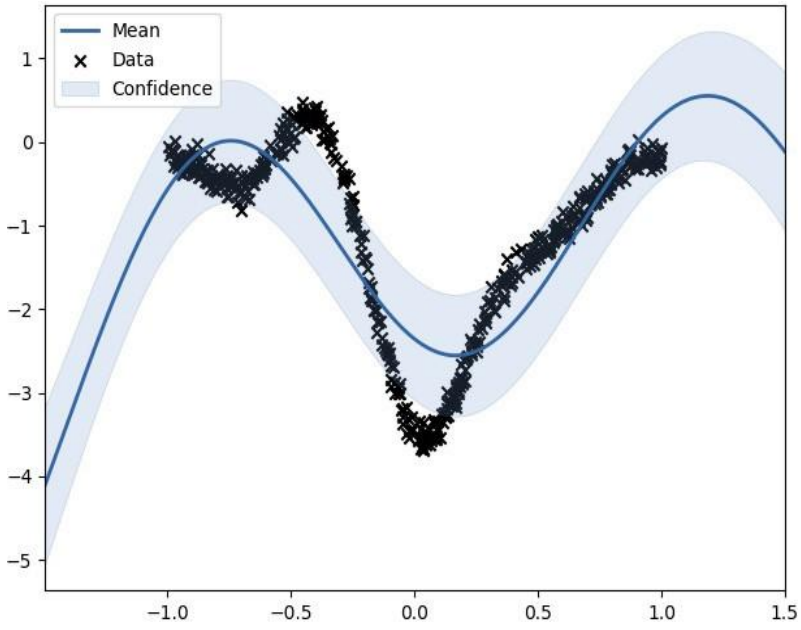
A. 2
B. 1

A



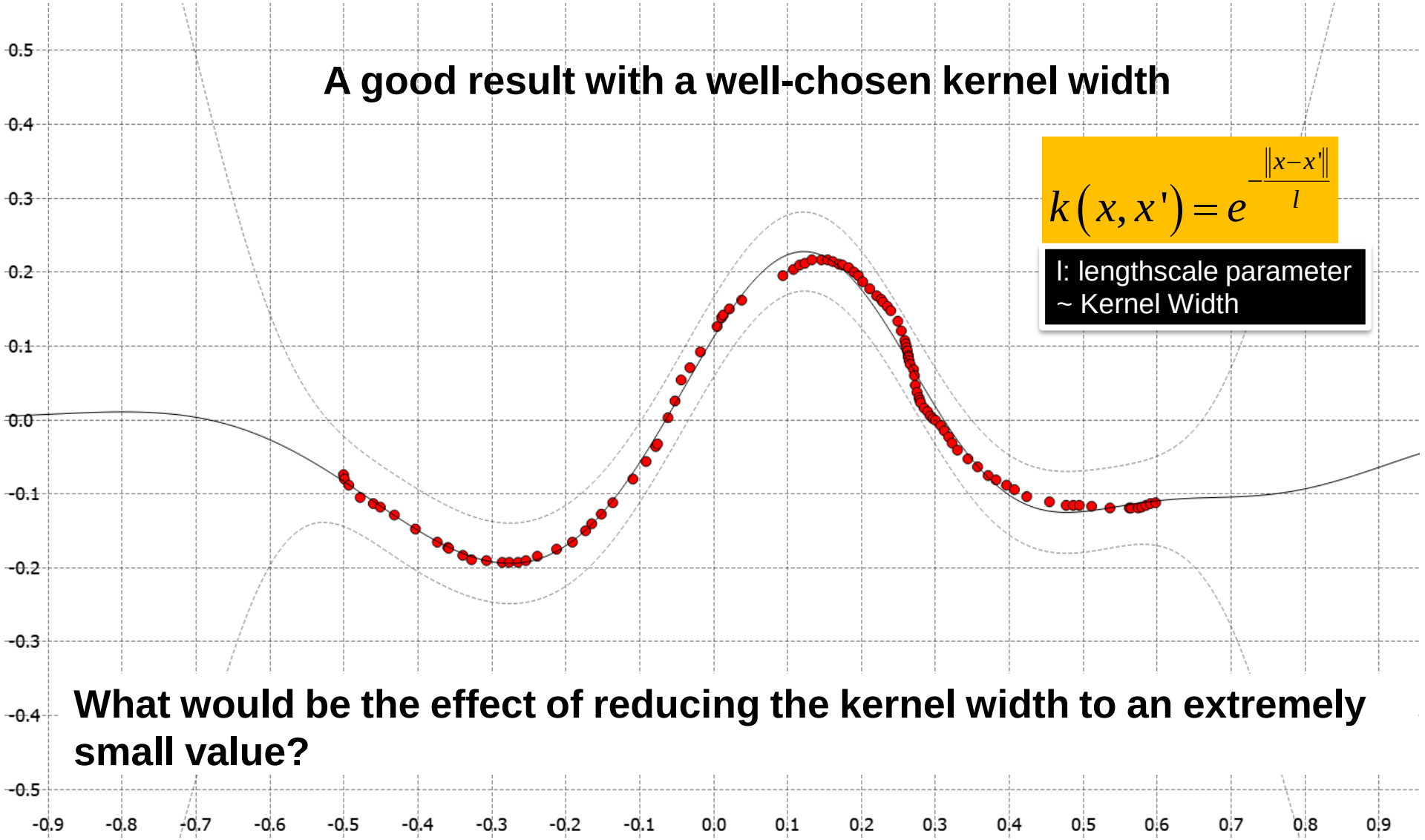
Kernel width = 0.1

B



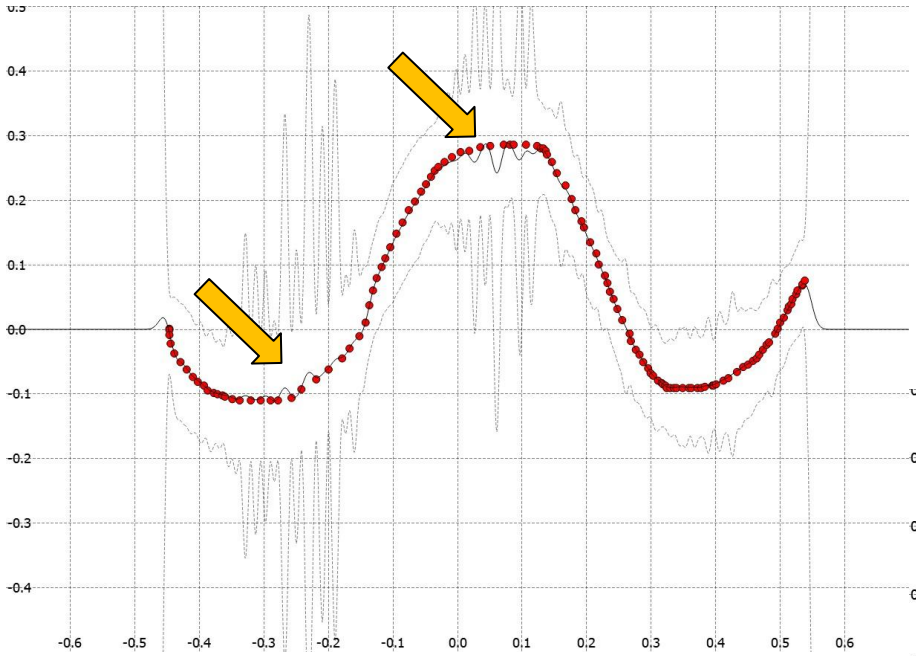
Kernel width = 1

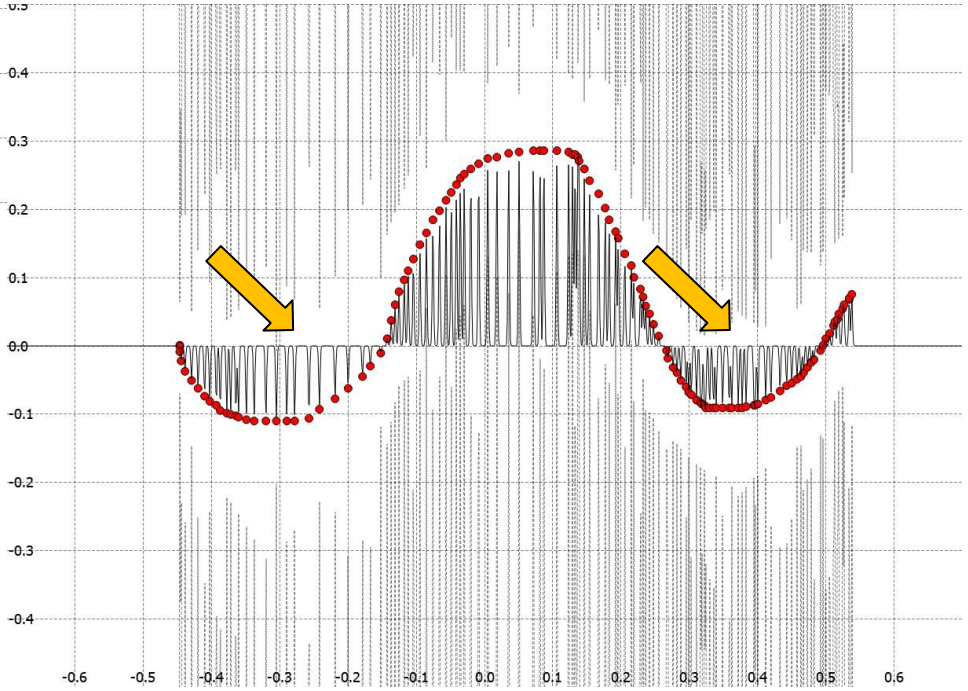
GPR – RBF Kernel – kernel width



GPR – RBF Kernel – kernel width

Too small a kernel width





Gaussian Process Regression (GPR)

Kernel function

$$y = E\{y \mid x, X, \mathbf{y}\} = \sum_{i=1}^{M} \alpha_i k\left(x, x^i\right)$$

with $\alpha = \left[K\left(X, X\right) + \sigma^2 I\right]^{-1} \mathbf{y}$

Other kernels

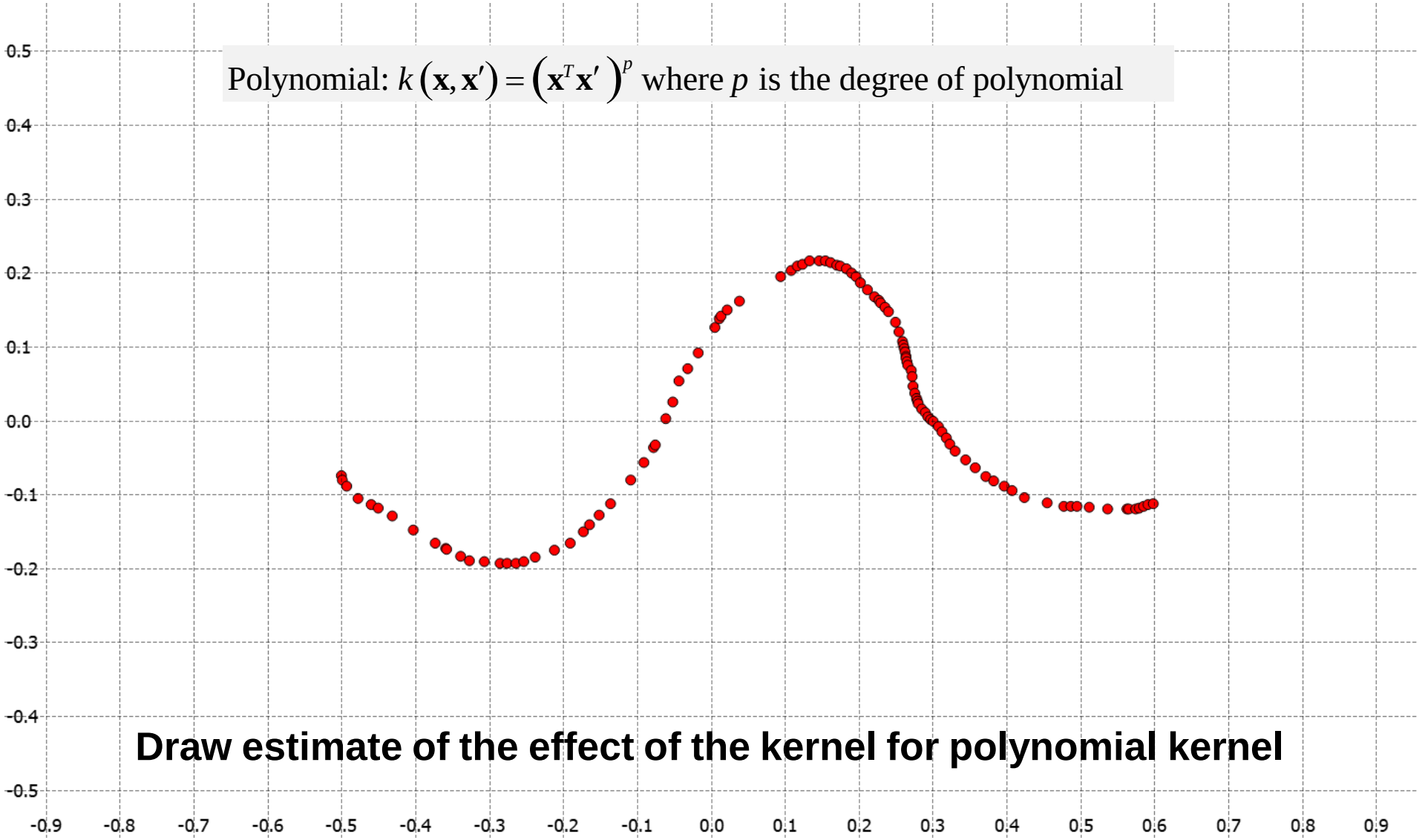
Linear: $k\left(\mathbf{x}, \mathbf{x}'\right) = \mathbf{x}^T \mathbf{x}' + c$ where c is the intercept

Polynomial: $k\left(\mathbf{x}, \mathbf{x}'\right) = \left(\mathbf{x}^T \mathbf{x}' + c\right)^p$ where p is the degree of polynomial

Periodic: $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{2\sin^2(\pi \| \mathbf{x} - \mathbf{x}' \| / T)}{\ell}\right)$ where T is the period.

GPR - Kernels

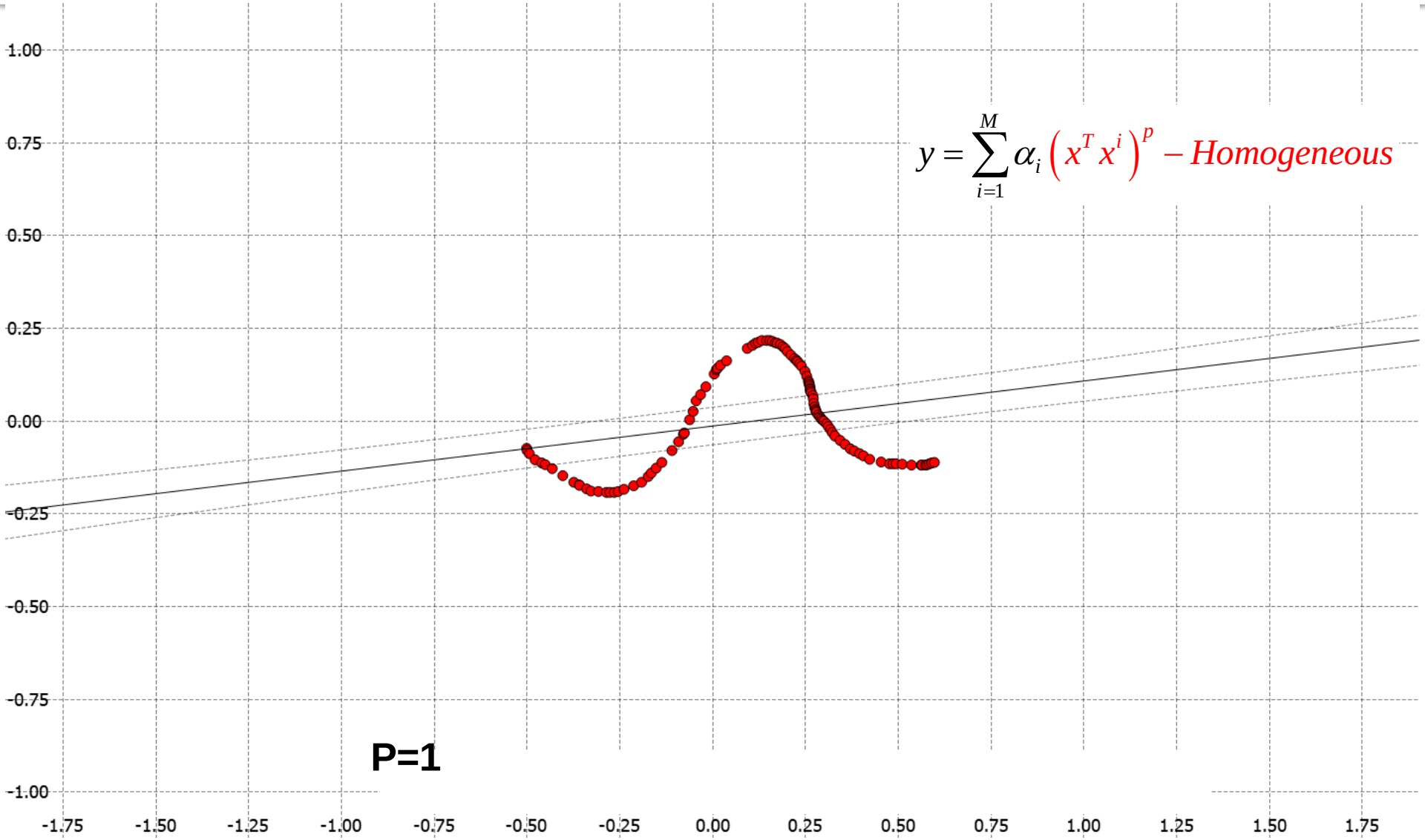
Polynomial: $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^p$ where p is the degree of polynomial



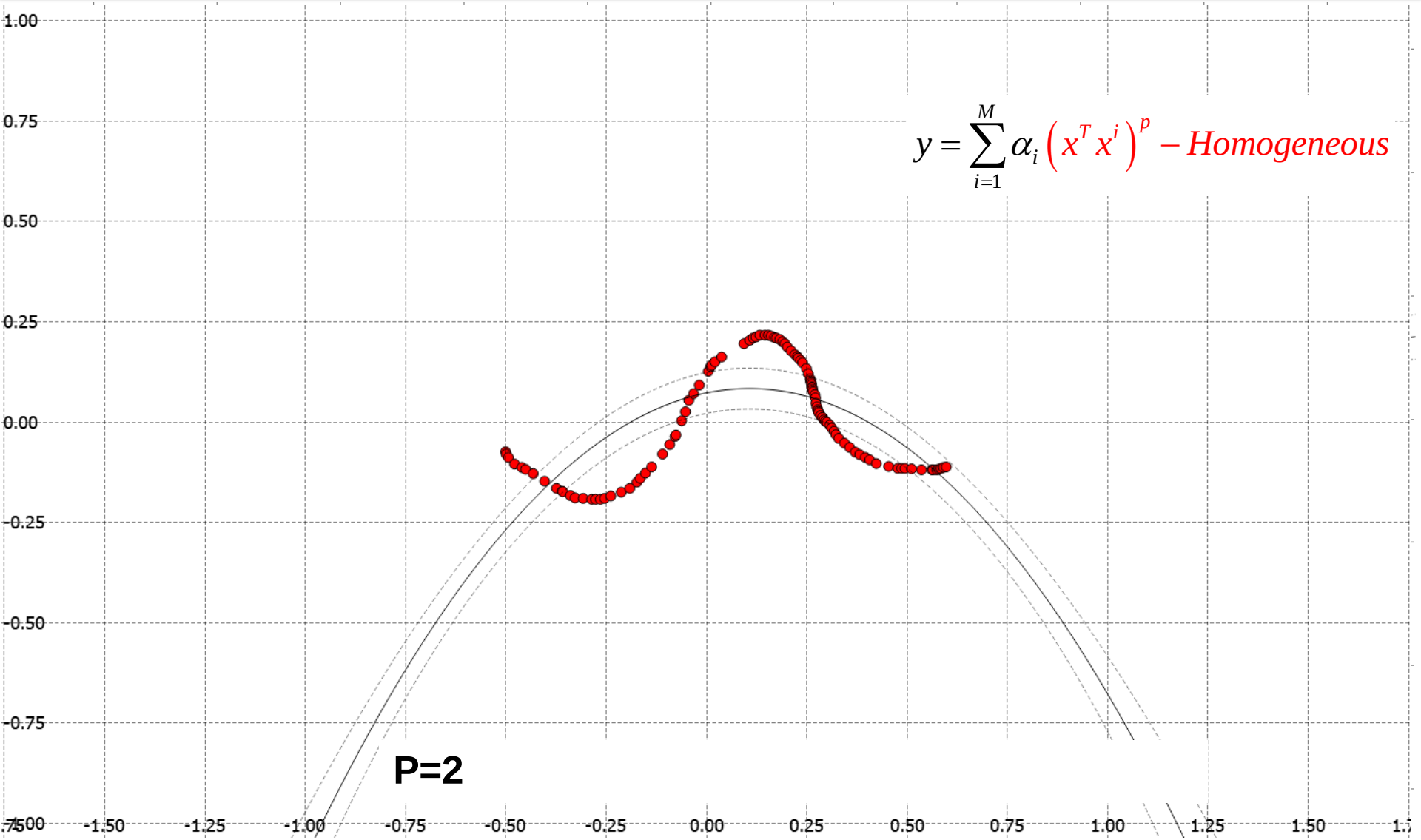
Draw estimate of the effect of the kernel for polynomial kernel

GPR - Kernels

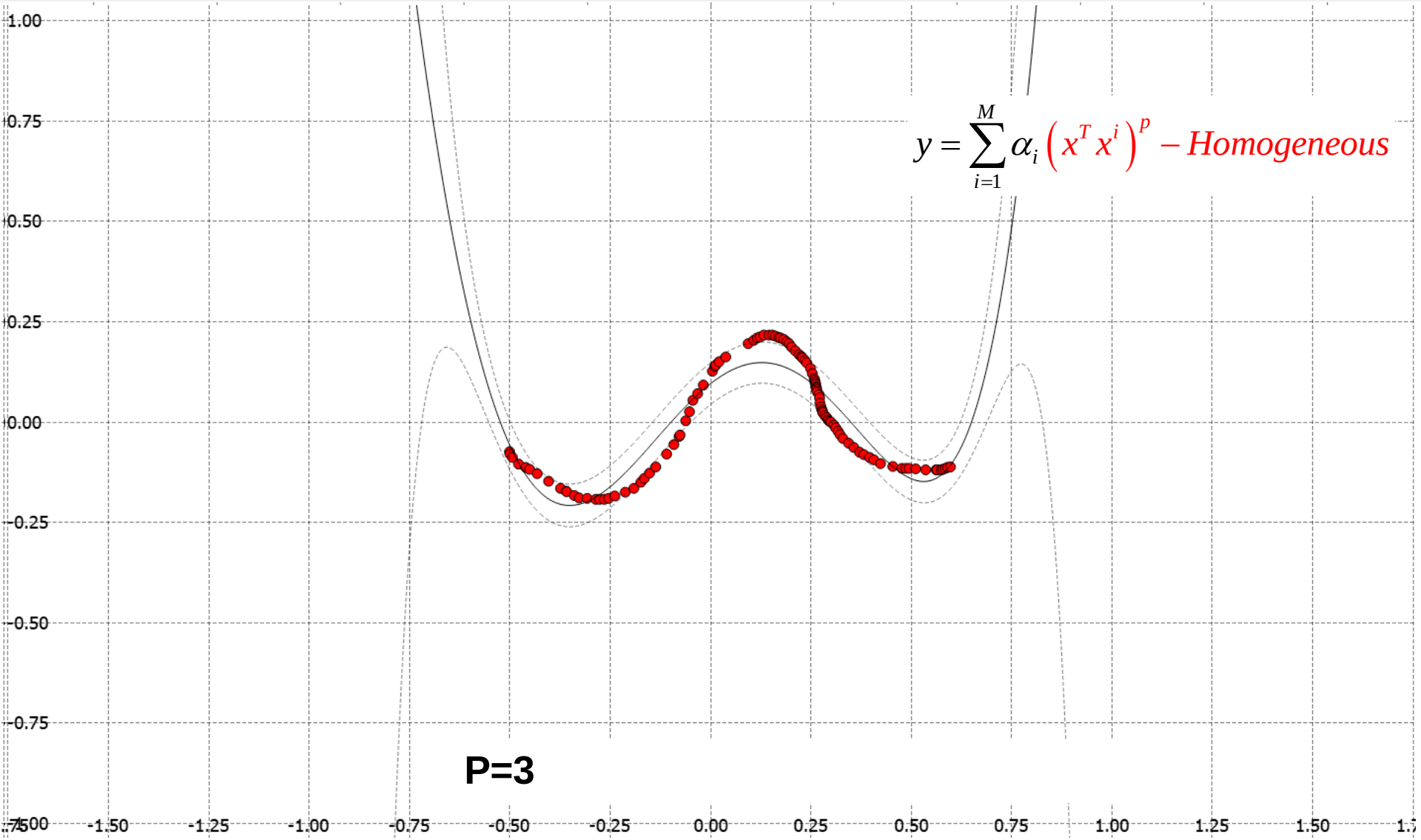
$$y = \sum_{i=1}^{M} \alpha_i \left(x^T x^i \right)^p \; - \textit{Homogeneous}$$



GPR - Kernels

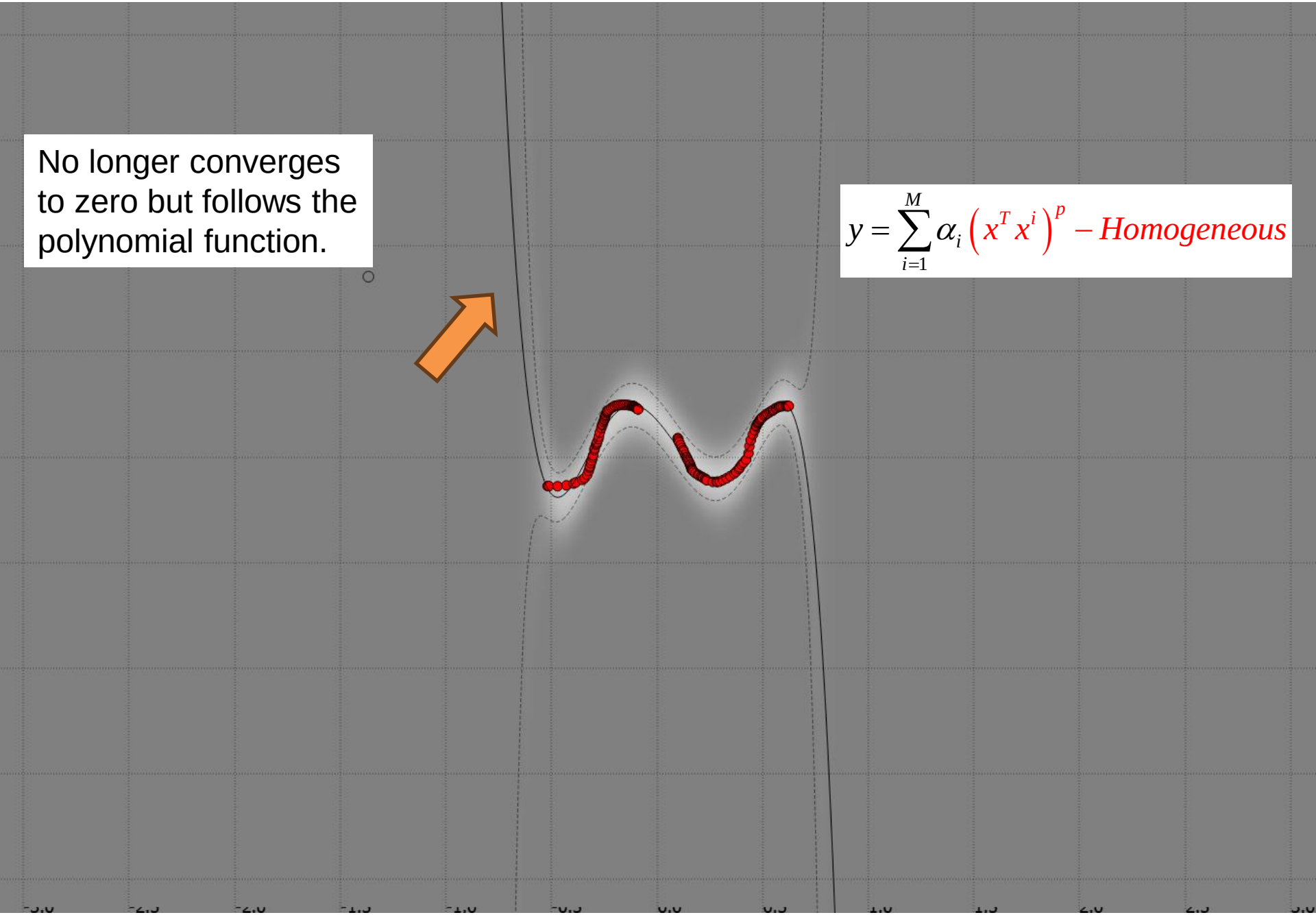

1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.50
-1.25
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
$y = \sum_{i=1}^{M} \alpha_i \left(x^T x^i \right)^p$ – Homogeneous
P=2

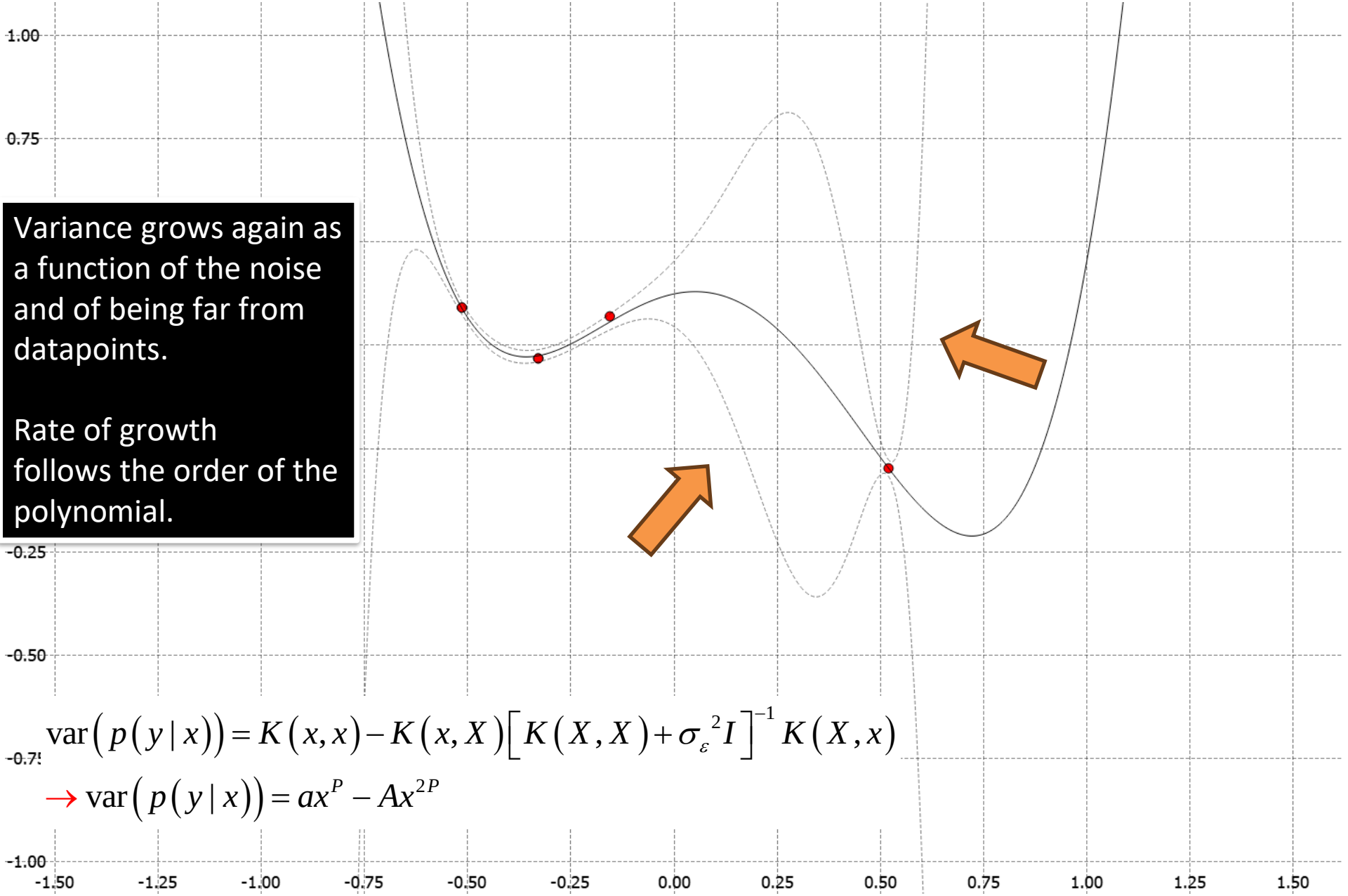
GPR - Kernels


1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.50
-1.25
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50
$y = \sum_{i=1}^{M} \alpha_i \left(x^T x^i \right)^p$ – Homogeneous
P=3

No longer converges to zero but follows the polynomial function.

$$y = \sum_{i=1}^{M} \alpha_i \left(x^T x^i \right)^p - \textit{Homogeneous}$$





Variance grows again as a function of the noise and of being far from datapoints.

Rate of growth follows the order of the polynomial.

$$\mathrm{var}\left(p(y \mid x)\right) = K(x,x) - K(x,X)\left[K(X,X) + \sigma_\varepsilon{}^2 I\right]^{-1} K(X,x)$$

$$\rightarrow \mathrm{var}\left(p(y \mid x)\right) = ax^P - Ax^{2P}$$

GPR with periodic kernel

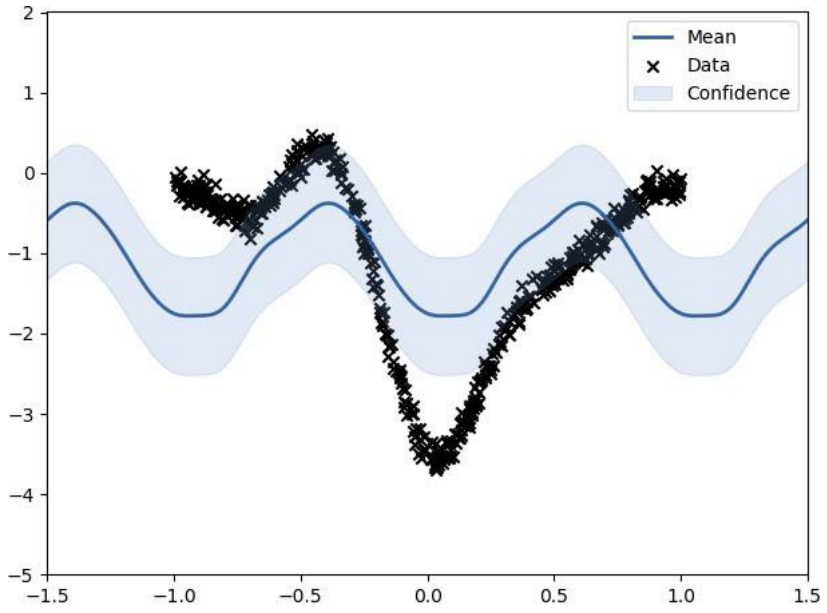
Which has the highest period?

A. 1

B. 2

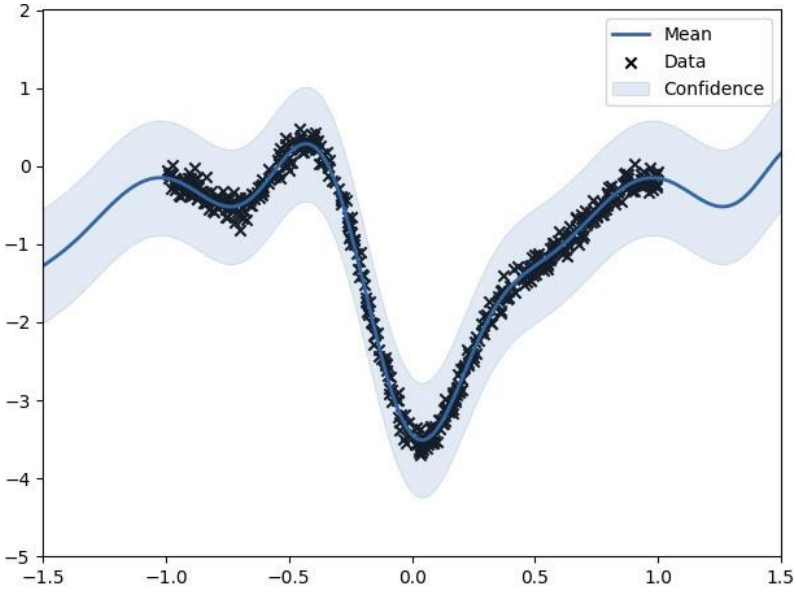
Periodic: $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{2\sin^2(\pi \| \mathbf{x} - \mathbf{x}' \| / T)}{\ell}\right)$ where T is the period.

A



T = 1

B



T = 2

GPR: nonstationary kernel

Kernel is usually Gaussian kernel with *stationary* covariance function
→ Non-Stationary Covariance Functions can encapsulate local variations in the density of the datapoints.

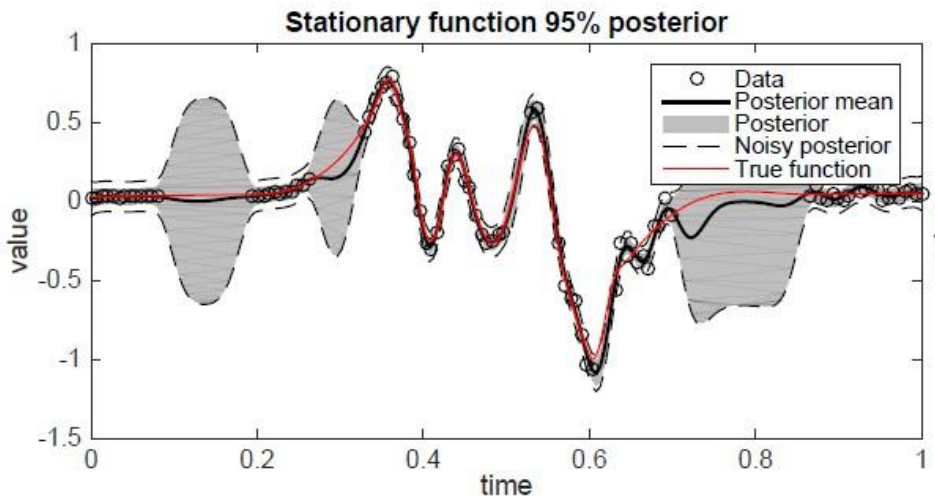
$$k(x,x') = \prod_{i=1}^{M} \left(\frac{2l_i(x)2l_i(x')}{l_i(x)+l_i(x')} \right)^{\frac{1}{2}} \exp\left(-\sum_{i=1}^{N} \frac{\left(x^i - x^i\right)^T \left(x^i - x^i\right)}{l_i^2(x)+l_i^2(x')} \right)$$

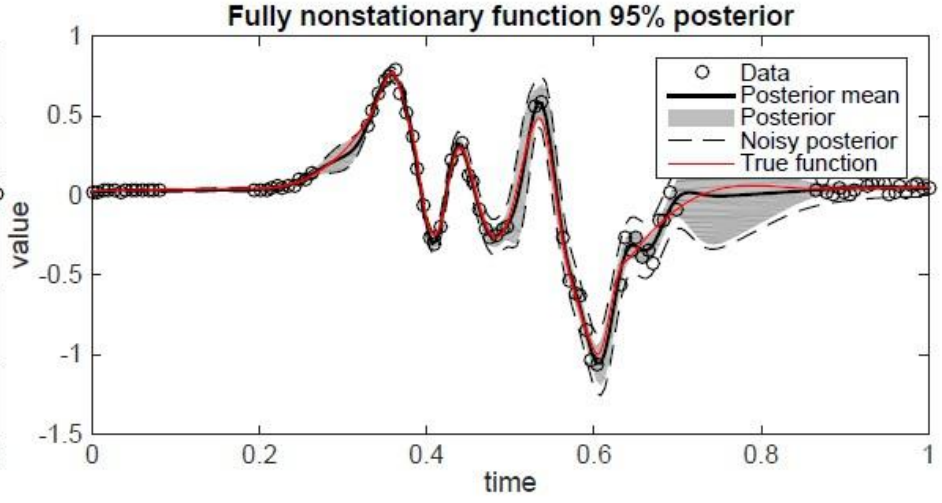
Gibbs' non stationary covariance function (length-scale a function of x)

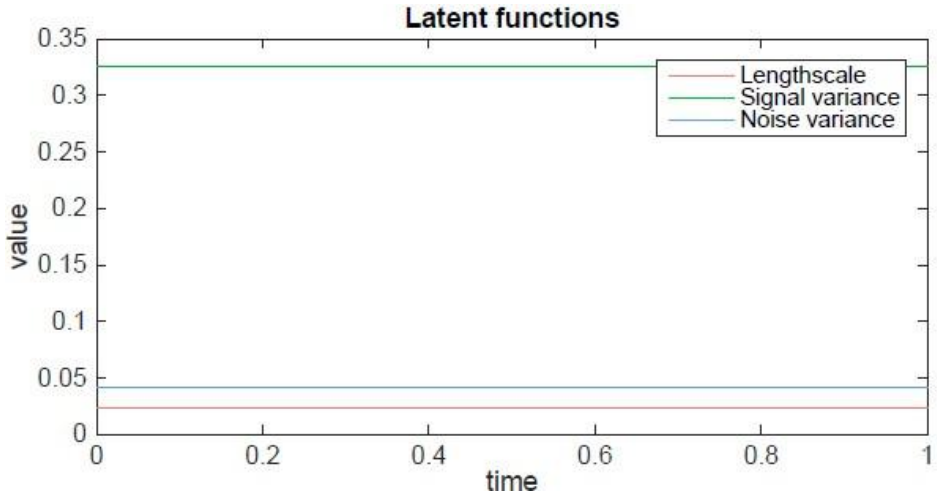
The parameters can be learned automatically (Heinonen et al. 2016).

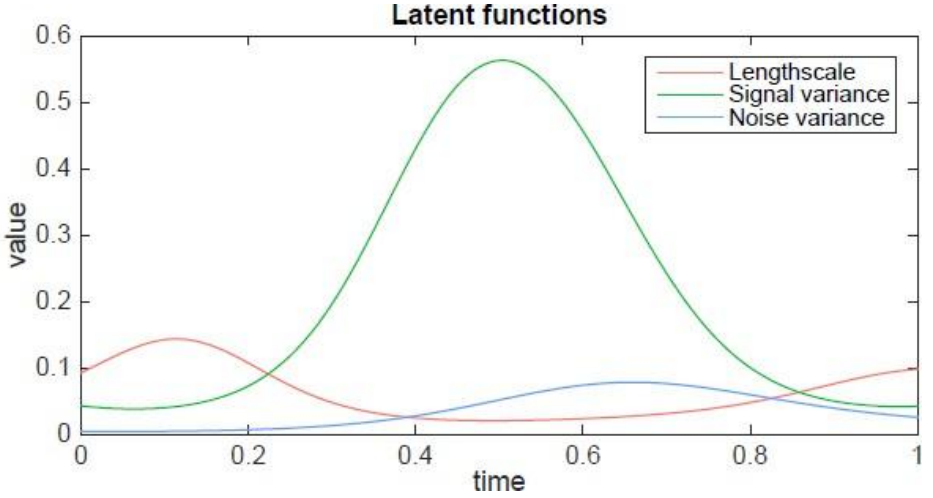
Heinonen, Markus, et al. "Non-stationary gaussian process regression with hamiltonian monte carlo." *Artificial Intelligence and Statistics*. PMLR, 2016.

GPR: nonstationary kernel



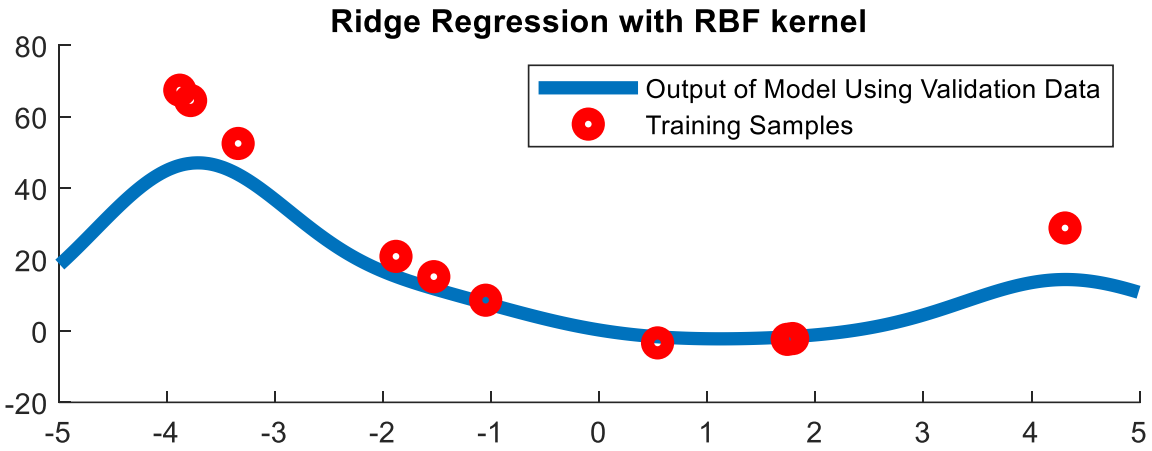




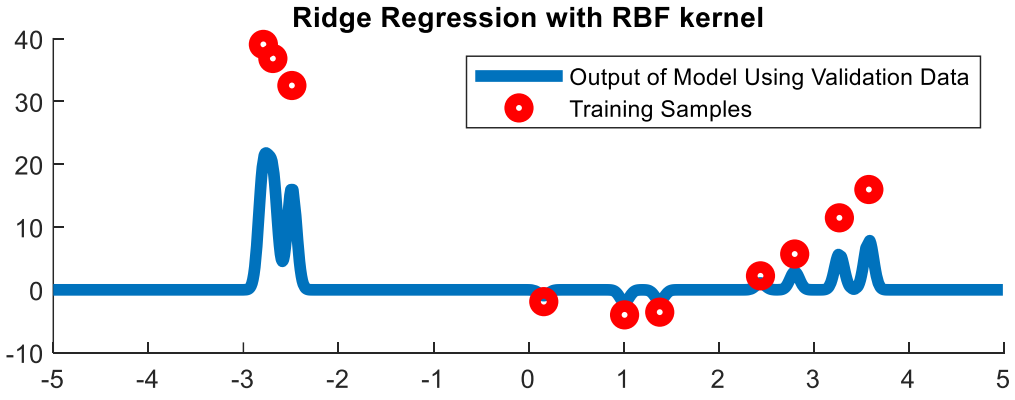


Heinonen, Markus, et al. "Non-stationary gaussian process regression with hamiltonian monte carlo." *Artificial Intelligence and Statistics*. PMLR, 2016.

Ridge Regression: Kernel

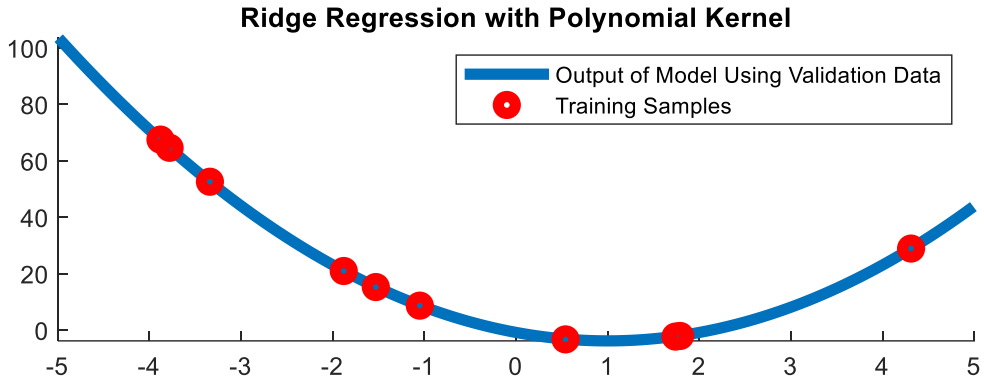


Large kernel width



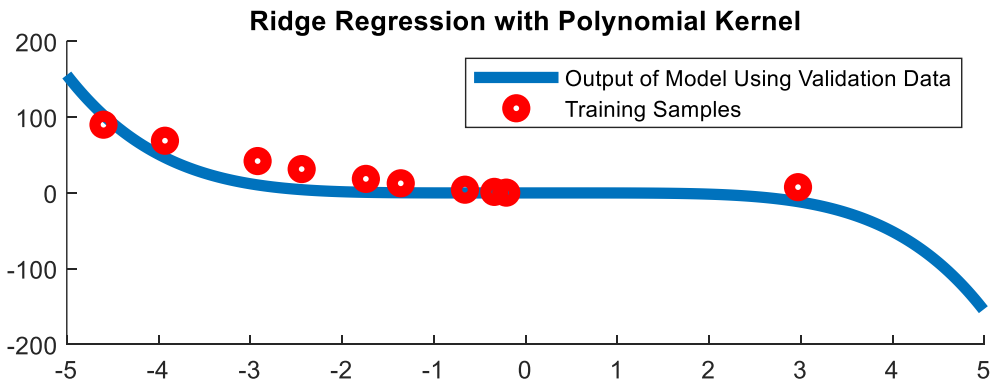
Small kernel width

Ridge Regression: kernel

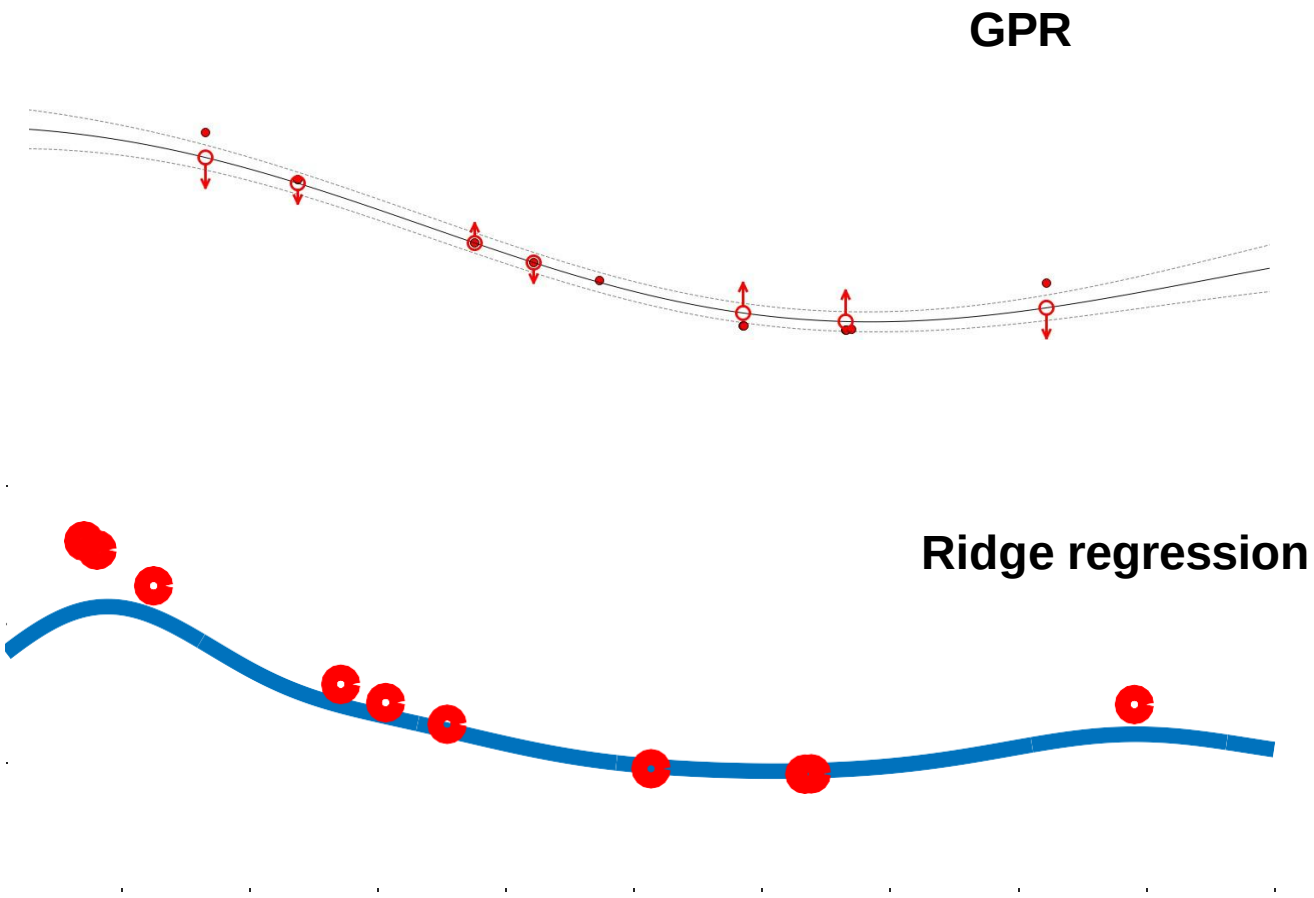


P=2

P=5

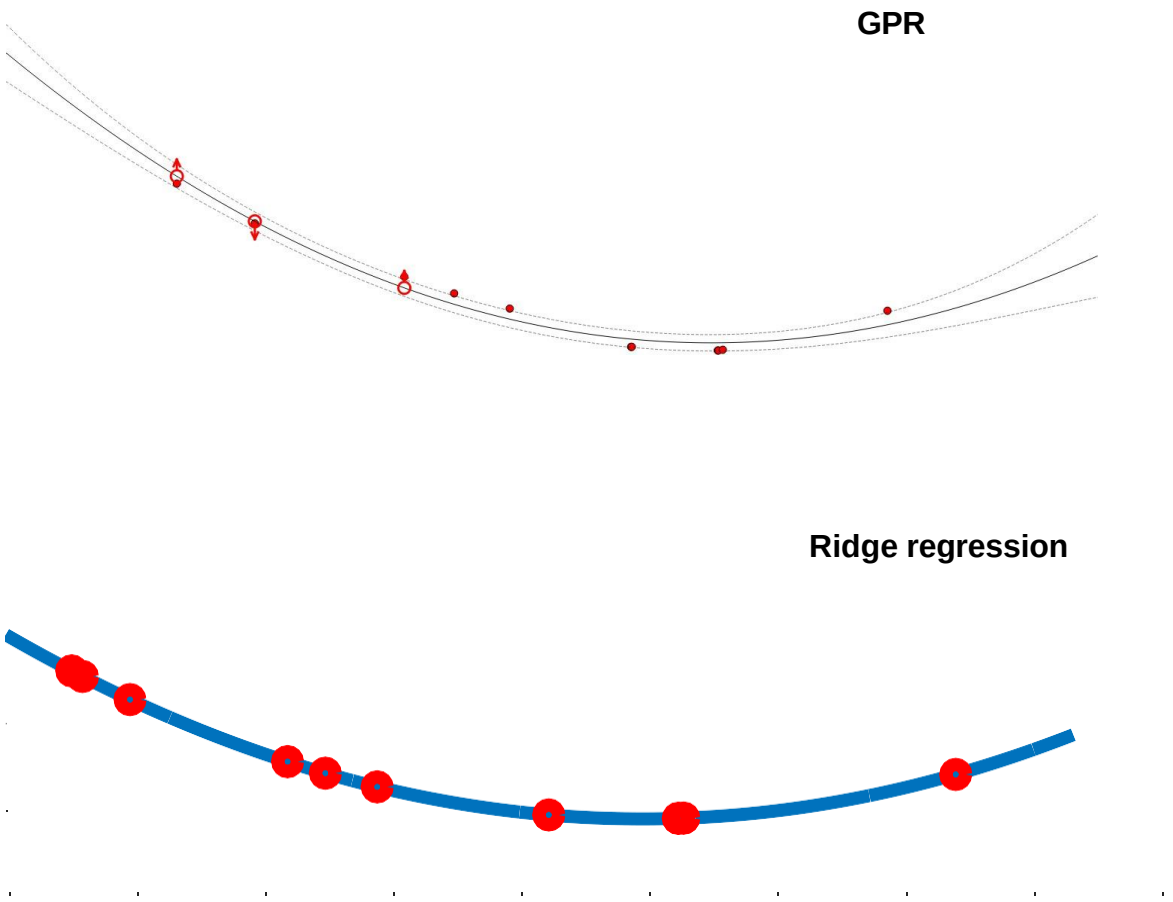


Ridge Regression versus GPR with RBF kernel



The regressive line of ridge regression has similar properties to that of Gaussian Process Regression. GPR adds a notion of variance.

Ridge Regression versus GPR with polynomial kernel



The regressive line of ridge regression has similar properties to that of Gaussian Process Regression. GPR adds a notion of variance.

Gaussian Process Extension for Classification and Manifold Learning

Gaussian Process for Classification

Gaussian Process for Classification

GPs from the Bayesian perspective

Recall that GPR takes a bayesian approach to estimating $f()$, where:

$y = f(\mathbf{x}) + \epsilon$,where: $\epsilon \sim \mathcal{N}(0, \sigma_y^2)$

An alternative approch to interpret derivation of GP is to say that

it models a collection of variables $\{ f_i := f(x^i) \}_{i=1}^{M}$ that jointly follow a Gauss distribution (same approach as that seen for RVM/RVR).

This generates a distribution of functions $p(f)$:

$$p(f) \sim \mathcal{GP}(m(x), k(x, x'))$$

$$m(x) = E\{ f(x) \}$$

$$k(x, x') = E\{ (f(x) - m(x))^T (f(x) - m(x)) \}$$

Gaussian Process for Classification

GPs from the Bayesian perspective

Under a GP, each estimate y is an instance function from the distribution $p(f)$

$$y_i \sim \mathcal{N}\left(\left[m\left(x^i\right)\right], K\right), \qquad y_i = E\left\{f_i(x)\right\}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \mathcal{N}\left(\begin{bmatrix} m\left(x^1\right) \\ m\left(x^2\right) \\ \vdots \\ m\left(x^M\right) \end{bmatrix}, \underbrace{\begin{bmatrix} k\left(x^1,x^1\right)+\sigma^2 & k\left(x^1,x^2\right) & \dots & k\left(x^1,x^M\right) \\ k\left(x^2,x^1\right) & k\left(x^2,x^2\right)+\sigma^2 & \dots & k\left(x^2,x^M\right) \\ \vdots & \vdots & \ddots & \vdots \\ k\left(x^M,x^1\right) & k\left(x^M,x^2\right) & \dots & k\left(x^M,x^M\right)+\sigma^2 \end{bmatrix}}_{K} \right)$$

The prediction $y_* = f\left(x^*\right)$ for a novel input x^* can be obtained by computing the conditional $p\left(y_* \mid x^*, X, Y\right)$,

Set of training pairs of datapoints: $X = [x^1 \dots x^M], Y = [y_1 \dots y_M]^T$

Gaussian Process for Classification

GPs from the Bayesian perspective

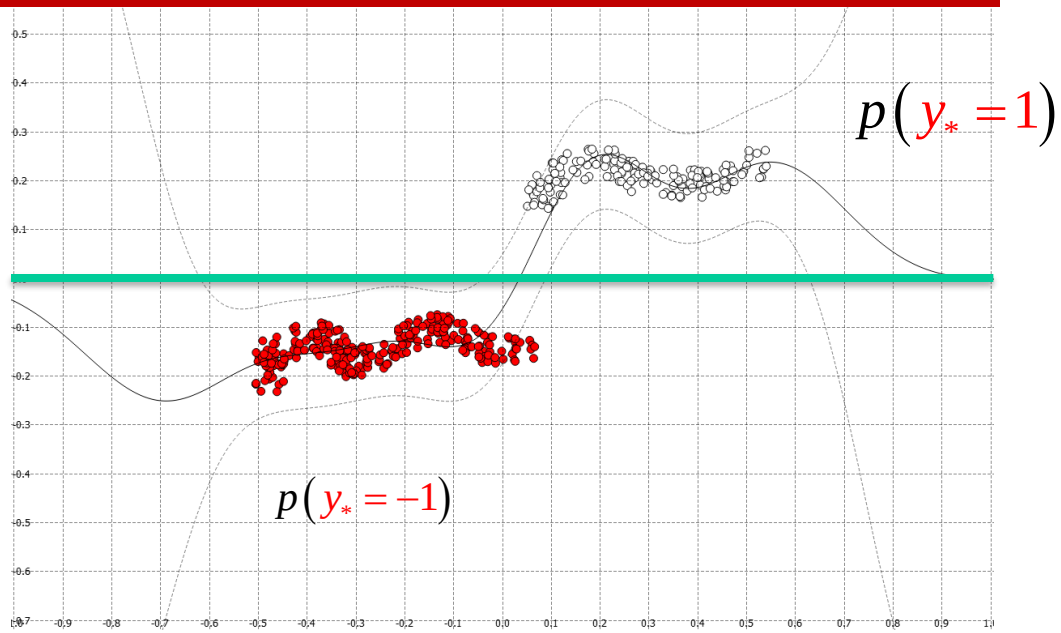
Class predictions for class label 1:

$$p\left(y_* = 1 \mid \mathbf{x}^*, \mathbf{X}_*, \mathbf{Y}\right)$$

Naive approch. Applies a threshold on output of GPR, e.g.

$y_* = +1$ if y > threshold

How can one compute the posterior estimate, $p\left(y_* = 1 \mid \mathbf{x}_*, \mathbf{X}_*, \mathbf{Y}\right)$?



Gaussian Process for Classification

GPs from the Bayesian perspective

Class predictions for class label 1:

$$p\left(y_* = 1 \mid \mathbf{x}^*, \mathbf{X}_*, \mathbf{Y}\right) = \int p\left(y_* = 1 \mid \mathbf{f}_*\right) p(\mathbf{f}_* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y}) d\mathbf{f}_*, \quad f_* = f\left(x^*\right)$$

Bayes rule for GPs:

$$p(\mathbf{f} \mid \mathbf{X}, \mathbf{y}) = \frac{p\left(\mathbf{y} \mid \mathbf{f}, \mathbf{X}\right) p\left(\mathbf{f}\right)}{p\left(\mathbf{y} \mid \mathbf{X}\right)}$$

Prior:

$$p\left(\mathbf{f}\right) \sim \mathcal{GP}\left(0, k\left(x, x'\right)\right)$$

Likelihood:

$$p\left(\mathbf{y} \mid \mathbf{f}, \mathbf{X}\right)$$

Posterior:

$$p\left(\mathbf{f} \mid \mathbf{X}, \mathbf{y}\right) \sim \mathcal{GP}\left(m_{\text{post}}, k_{\text{post}}\left(x, x'\right)\right)$$

Posterior predictive distribution of class label:

$$p(\mathbf{f}_* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p\left(\mathbf{f}_* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y}, \mathbf{f}\right) p\left(\mathbf{f} \mid \mathbf{X}, \mathbf{Y}\right) d\mathbf{f}$$

No closed form predictions. It must be approximated.

Gaussian Process for Classification

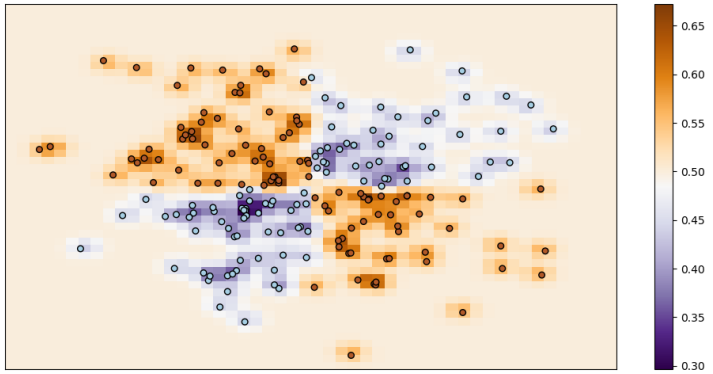
Approximation methods

Many methods exist to approximate the posterior:

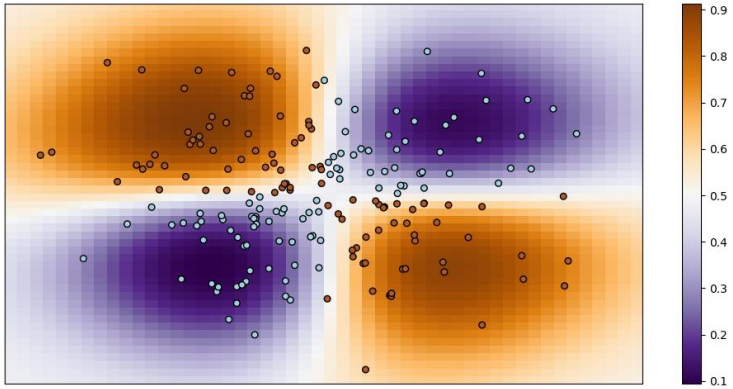
- Laplace Approximation
- Expectation propagation
- Markov Chain Monte Carlo

Gaussian Process for Classification

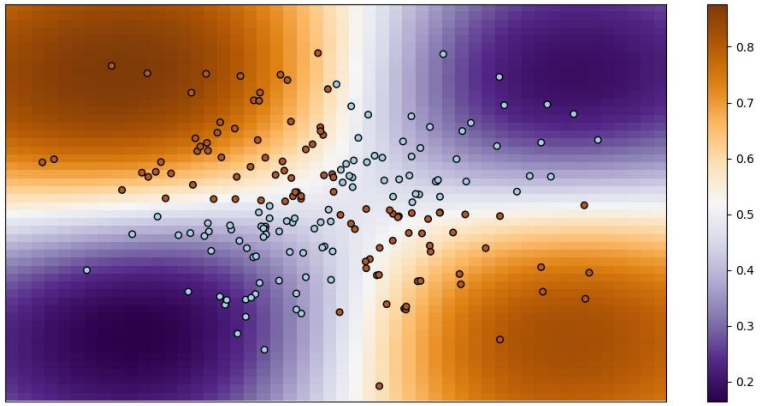
Impact of RBF kernel width



Width =0.1



Width = 1

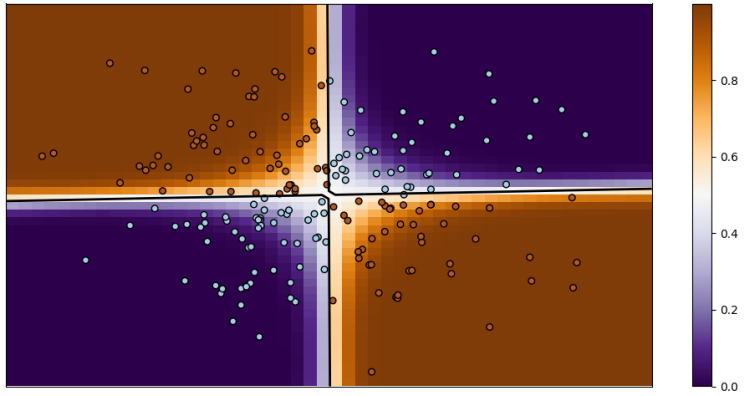


Width = 2

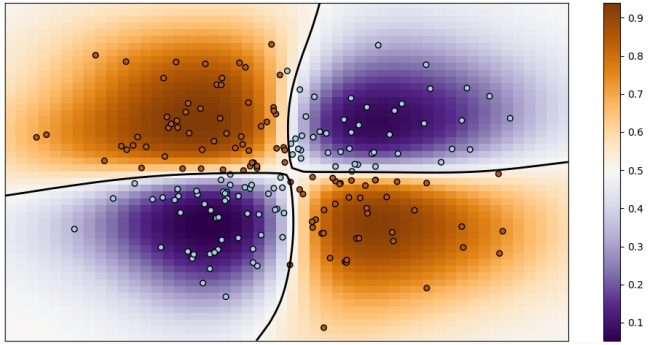
GP classification of a XOR dataset with various RBF widths.

Gaussian Process for Classification

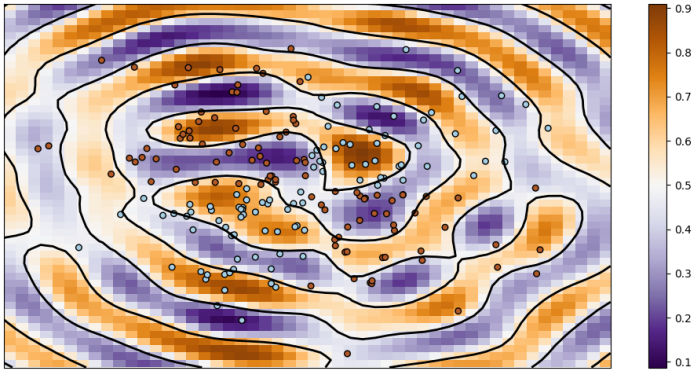
Toy examples



2nd degree polynomial



RBF



Periodic

GP classification of a XOR dataset with different kernels

Gaussian Process for Manifold Learning

Gaussian Process Latent Variable Models - GPLVM

GPLVM:

- extends the concept of Gaussian Process to determine latent manifold,
- is a manifold learning technique,
- is an unsupervised variant on Gaussian Process.
- can be seen as a probabilistic dimensionality reduction technique.

- Derivation
  - PCA to Probabilistic PCA
  - Dual formulation of Probabilistic PCA
  - GPLVM through kernel trick

  (omitted from these slides, see supplement)

PCA: principle

PRINCIPLE:

- Define a low dimensional manifold in the original space.
- Represent each data point *X* by its projection *Y* onto this manifold.

FORMALISM:

Consider a data set of M N-dimensional data points

$$\mathrm{X}=\left\{x_j^i\right\}_{j=1,\ldots N}^{i=1\ldots M} \quad \text{and} \quad \mathrm{x}^i \in \mathbb{R}^N, i=1,\ldots,M:$$

PCA aims at finding a linear map A, s.t

$$A\colon \mathbb{R}^N \xrightarrow{A} \mathbb{R}^q, \quad \mathrm{q} \leq N$$

$$Y = AX, \qquad Y=\left\{y^1,\ldots,y^M\right\} \quad \text{and each} \quad y^i \in \mathbb{R}^q$$

Standard PCA: Variance Maximization through Eigenvalue Decomposition

Algorithm:

1) Zero mean: $X \rightarrow X' = X - E\{X\}$

2) Compute Correlation matrix: $C = X'(X')^T$

3) Compute eigenvalues using $|C - \lambda_i I| = 0, \; i = 1...N$

4) Compute eigenvectors using $Ce_i = \lambda_i e_i$

5) Choose first $q < N$ eigenvectors: $e_1, ...e_q$ with $\lambda_1 \geq \lambda_2 \geq ...\lambda_q$

6) Project data onto new basis: $X' \rightarrow X'' = W_q X', \; W_q = \begin{pmatrix} e_1^1 .....e_N^1 \\ .. \\ e_q .....e_q^1 \end{pmatrix}$

LIMITATION OF STANDARD and MSQ PCA:

The variance-covariance matrix needs to be calculated:

- ❑ Can be very computation-intensive for large datasets with a high # of dimensions
- ❑ Does not deal properly with missing data
- ❑ Incomplete data must either be discarded or imputed using ad-hoc methods
- ❑ Outlying data observations can unduly affect the analysis

→ Probabilistic PCA addresses the above limitation

PCA with Latent variable models

Latent variables correspond to unobserved variables. They offer a *lower dimensional* representation of the data and their *dependencies.*

In PCA, the latent variable model consists then of:
– *X:* observed variables (*Dimension N*)
– *z:* latent variables *(Dimension q*)
with $q<N$

Less dimensions results in more parsimonious models

Probabilistic PCA

Idea: Assume that the data *X* were generated by a Gaussian latent variable model, Probabilistic PCA (PPCA) consists then in estimating the density of the latent variable through maximum likelihood.

Probabilistic PCA is then PCA through projection on a latent space.

Advantages of expressing PCA in probabilistic form:

- It can easily be extended to estimation from mixtures of PCA models.
- The estimated density can easily be used for classification and other Bayesian computation afterwards.

Probabilistic PCA

The latent variables z generate the observables x following:

$x = W^T z + \mu + \varepsilon$

- The latent variable z are centered and white, i.e. $z = \mathrm{N}(0, I)$
- μ is a parameter (usually the mean of the data x)
- ε the noise follows a zero mean Gaussian distribution $\varepsilon = \mathrm{N}(0, \sum_{\varepsilon}^{2})$
- W^T is a $N \times q$ matrix.

If the variance of the noise is diagonal
→ conditional independence on the observables *given* the latent variables
→ z encapsulate the correlations.

Probabilistic PCA

Assuming further an isotropic Gaussian noise model $\mathrm{N}(0, \sigma_\varepsilon^2 \mathrm{I})$ implies that the conditional probability distribution of the observables given the latent variables $p(x \mid z)$ is given by:

$$p(x \mid z) = \mathrm{N}\left(W^T z + \mu, \sigma_\varepsilon^2 I\right)$$

The marginal distribution can be computed by integrating out the latent variable and is then:

$$p_z(x) = \mathrm{N}\left(\mu, W^T W + \sigma_\varepsilon^2 I\right)$$

Probabilistic PCA

If we set $B = W^T W + \sigma_\varepsilon^2 I$, one can then compute the log-likelihood:

$$\mathrm{L}\left(B, \sigma_\varepsilon, \mu\right) = -\frac{M}{2}\left\{N \ln\left(2\pi\right) + \ln\left|B\right| + tr\left(B^{-1}C\right)\right\}$$

where C is the sample covariance matrix of the complete set of M datapoints $\mathrm{X} = \left\{x^1, ..., x^M\right\}$.

The maximum likelihood estimate of μ is the mean of the dataset X.

The parameters B and σ_ε are estimated through E-M.

Probabilistic PCA

The maximum-likelihood estimate of B and σ_{ϵ}^2 is:

$$B^* = W_q^T \left(\Lambda_q - \sigma_{\varepsilon}^2 I\right)^{\frac{1}{2}} R$$

$$\left(\sigma_{\epsilon}^*\right)^2 = \frac{1}{N-q} \sum_{j=q+1}^{N} \lambda_j$$

Residuals

where W_q^T is the matrix of eigenvectors of C and the λ_j are the associated eigenvalues.

Probabilistic PCA: Dimensionality Reduction

Reduction of the dimensionality is obtained by looking at the latent variable and estimating their distribution.

The conditional distribution of the latent variable over the data is given by:

$$p(z \mid x) = \mathrm{N}\left(B^{-1}W(x-\mu), B^{-1}\sigma_{\varepsilon}^{2}\right)$$

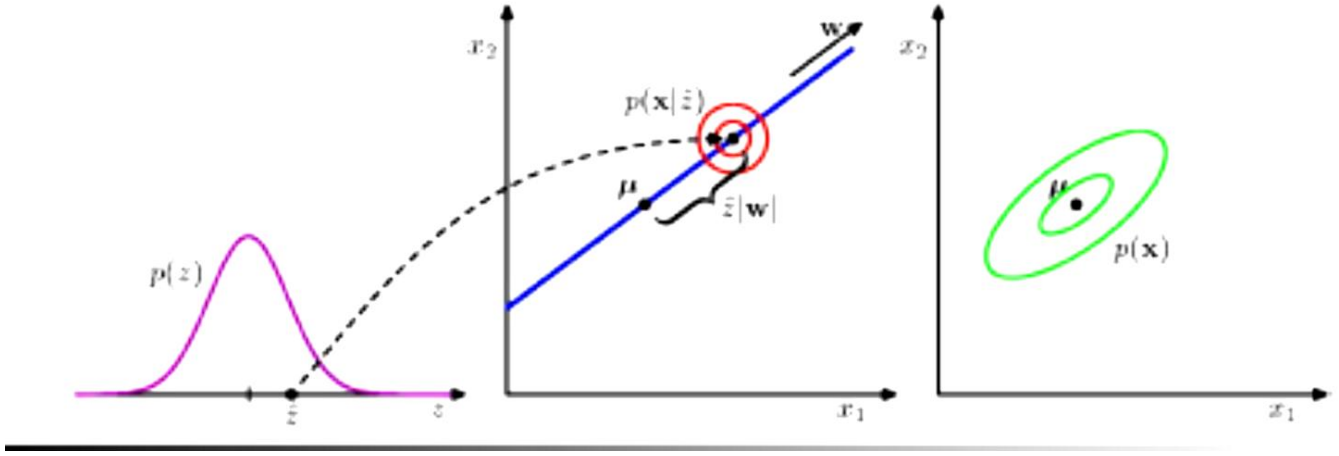
Is again Gaussian!

$$B = W_q^T W_q + \sigma_{\varepsilon}^{2} I$$

In the absence of noise, one recovers standard PCA, as $\left((W)^T W\right)^{-1} W(x-\mu)$ is an orthogonal projection of x onto the latent space.

Probabilistic PCA

Assumptions:
– underlying latent variable has a Gaussian distribution
– linear relationship between latent and observed variables
– isotropic Gaussian noise in observed dimensions



Probabilistic PCA: reconstruction

Note that projection of the data through $\hat{x} = W_q^T \bar{z}$ is no longer optimal in a mean-square sense because of the noise parameter, i.e. when $\sigma_\varepsilon^2 > 0,$ the projection through $\left((W)^T W + \sigma_\varepsilon^2 I \right)^{-1} W$ is no longer orthogonal.

Nevertheless, optimal reconstruction of the observed data from the conditional latent mean $\bar{z}$ may still be obtained through $\hat{x} = B^* \left(B^* B^{*T} \right) \left(BB^T + \sigma_\varepsilon^2 I \right) z + \mu.$

From Linear Probabilistic PCA
To Probabilistic Kernel PCA

→ GPLVM

Probabilistic PCA: Advantages

Probabilistic PCA can be applied as a simple covariance model, using W and σ.

Reduces computation of full covariance matrix through projection to dimension *q.*

PPCA offers a natural approach to the estimation of the principal axes in cases where some, or indeed all, of the data vectors X exhibit one or more *missing* (at random) values.

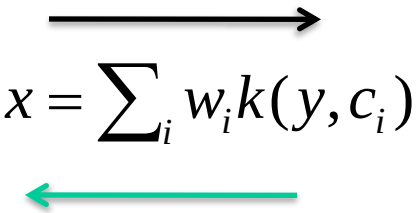
→ Exploit E-M approach to estimating the latent variables

kPCA: From Feature to Original Space

Two-side projection: what if we need to manipulate data in the projected space and then reconstruct them back to the original?

kPCA -> does not define the projection sending back from feature space to the original data-space.

Hard



Easy

Dual Probabilistic PCA II

Latent Variables Optimization:

$$p(\mathbf{Y} \mid \mathbf{X}, \mathbf{W}) = \prod_{i=1}^{M} N(\mathbf{y}_{i,:} \mid \mathbf{W}\mathbf{x}_{i,:}, \sigma^2 \mathbf{I})$$

Define Gaussian prior over the parameter, $\mathbf{W}$:

$$p(\mathbf{W}) = \prod_{i=1}^{N} N(\mathbf{w}_{i,:} \mid \mathbf{0}, \mathbf{I})$$

Integrate out the parameters:

$$p(\mathbf{Y} \mid \mathbf{X}) = \prod_{i=1}^{M} N(\mathbf{y}_{i,:} \mid 0, \mathbf{X}\mathbf{X}^T + \sigma^2 \mathbf{I})$$

Dual Probabilistic PCA II

The marginalized likelihood takes the form:

$$p(\mathbf{Y} \mid \mathbf{X}, \sigma) = \prod_{i=1}^{M} \mathrm{N}(\mathbf{y}_i \mid 0, \mathbf{X}\mathbf{X}^T + \sigma^2 \mathbf{I})$$

To find the latent variables, optimize the log-likelihood:

$$L = -\frac{\mathrm{N}M}{2} \ln 2\pi - \frac{\mathrm{N}}{2} \ln |\mathbf{K}| - \frac{1}{2} tr(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T) \qquad \mathbf{K} = \mathbf{X}\mathbf{X}^T + \sigma^2 \mathbf{I}$$

The gradient of L w.r.t. the latent variables:

$$\frac{\partial L}{\partial \mathbf{X}} = \mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\mathbf{K}^{-1}\mathbf{X} - N\mathbf{K}^{-1}\mathbf{X}$$

Dual Probabilistic PCA II

Introducing the substitution: $\mathbf{S} = \mathrm{N}^{-1}\mathbf{Y}\mathbf{Y}^T$

We rewrite the gradient of the likelihood as:

$$\frac{\partial L}{\partial \mathbf{X}} = \mathbf{K}^{-1}\mathbf{S}\mathbf{K}^{-1}\mathbf{X} - \mathbf{K}^{-1}\mathbf{X} = \mathbf{0} \quad \text{or} \quad \mathbf{S}(\underbrace{\sigma^2\mathbf{I} + \mathbf{X}\mathbf{X}^T}_{\mathbf{K}})^{-1}\mathbf{X} = \mathbf{X}$$

Dual Probabilistic PCA II

Introducing the substitution: $\mathbf{S} = \mathrm{N}^{-1}\mathbf{Y}\mathbf{Y}^T$

We rewrite the gradient of the likelihood as:

$$\frac{\partial L}{\partial \mathbf{X}} = \mathbf{K}^{-1}\mathbf{S}\mathbf{K}^{-1}\mathbf{X} - \mathbf{K}^{-1}\mathbf{X} = \mathbf{0} \quad \text{or} \quad \mathbf{S}(\sigma^2\mathbf{I} + \mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X} = \mathbf{X}$$

Let us consider the single value decomposition of $\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^T$, therefore, we can rewrite the equation for the latent $\mathbf{X}$:

$$\mathbf{S}\mathbf{U}[\mathbf{L} + \sigma^2\mathbf{L}^{-1}]^{-1}\mathbf{V}^T = \mathbf{U}\mathbf{L}\mathbf{V}^T$$

The solution is invariant to $\mathbf{V}$:

$$\mathbf{S}\mathbf{U} = \mathbf{U}(\sigma^2\mathbf{I} + \mathbf{L}^2)$$

Dual Probabilistic PCA II

Need to find matrices in the decomposition of $\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^T$

$\mathbf{\Lambda}$ is the matrix of eigenvalues of $\mathbf{S}$

The covariance matrix of the observable data $\rightarrow$ $\mathbf{S}\mathbf{U} = \mathbf{U}(\sigma^2\mathbf{I} + \mathbf{L}^2)$

Eigenvectors of $\mathbf{S}$

$\Rightarrow$ This implies that the elements from the diagonal of $\mathbf{L}$ are given by

$$l_i = (\lambda_i - \sigma^2)^{\frac{1}{2}}$$

Dual Probabilistic PCA II

- Solution for PPCA:

$$\mathbf{Y}^T\mathbf{Y}\mathbf{U}_W = \mathbf{U}_W\mathbf{\Lambda} \qquad \mathbf{W} = \mathbf{U}_W\mathbf{L}\mathbf{V}^T$$

- Solution for Dual PPCA:

$$\mathbf{Y}\mathbf{Y}^T\mathbf{U} = \mathbf{U}\mathbf{\Lambda} \qquad \mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^T$$

- Equivalence is of the form:

$$\mathbf{U}_W = \mathbf{Y}^T\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}$$

If one knows a solution of PPCA then the solution of Dual PPCA can be computed directly through the equivalence form.

Marginalization over the latent variables and parameters is equivalent. But marginalization over the parameters allows for interesting extensions; see next.

Gaussian Processes I

To recall, the marginal likelihood in Dual PPCA is given by:

$$p(\mathbf{Y} \mid \mathbf{X}) = \prod_{i=1}^{M} N(\mathbf{y}_i \mid 0, \mathbf{X}\mathbf{X}^T + \sigma^2 \mathbf{I})$$

The following distribution is known as the *Gaussian Process*:

$$p(\mathbf{y} \mid \mathbf{X}) = N(\mathbf{y} \mid 0, \mathbf{K}(\mathbf{X}))$$

Hence the marginal likelihood in dual PPCA is the combination of M independent Gaussian Processes.

Gaussian Processes I

To recall, the marginal likelihood in Dual PPCA is given by:

$$p(\mathbf{Y} \mid \mathbf{X}) = \prod_{i=1}^{M} N(\mathbf{y}_i \mid 0, \mathbf{X}\mathbf{X}^T + \sigma^2 \mathbf{I})$$

The following distribution is known as the *Gaussian Process*:

$$p(\mathbf{y} \mid \mathbf{X}) = N(\mathbf{y} \mid 0, \mathbf{K}(\mathbf{X}))$$

Gaussian Processes have many useful properties (e.g. for modeling nonLinear functional dependencies). For now, we just need to pay attention to the covariance function

n dual PPCA is the combination
ocesses.

Non-linear Latent Variables Model

In the marginal likelihood of Dual PPCA:

$$p(\mathbf{Y} \mid \mathbf{X}) = \prod_{i=1}^{M} N(\mathbf{y}_{i,:} \mid 0, \mathbf{X}\mathbf{X}^T + \sigma^2 \mathbf{I})$$

Linear Covariance function with the noise term

What if we use a non-linear covariance function (e.g. RBF)?

Non-linear Latent Variables Model

We get a non-linear mapping into a low dimension manifold:

Gaussian Process Latent Variables Model

- **Consider a Gaussian kernel:**

$$k(\mathbf{x}, \mathbf{x}') = \alpha_1 \exp\left[-\frac{\alpha_2}{2} \|\mathbf{x} - \mathbf{x}'\| \right]$$

- **No longer possible to find a closed-form solution when optimizing for *X*. No longer possible to simple proceed to a single value decomposition**

- **Instead find gradient w.r.t. $\mathbf{X}, \alpha_1, \alpha_2, \sigma^2$ and optimize using conjugate gradients**

Non-linear Latent Variables Model

We get a non-linear mapping into a low dimension manifold:

Gaussian Process Latent Variables Model

Optimization of the non-linear model is done by gradient descent on:

$$\frac{\partial L}{\partial K} = K^{-1}YY^{T}K^{-1} - \mathrm{N}K^{-1},$$

with each element of K given by $k(\mathbf{x},\mathbf{x}') = \alpha_1 \exp\left[-\frac{\alpha_2}{2}\|\mathbf{x}-\mathbf{x}'\|\right]$

Computationally heavy → sparse technique: pick a subset of datapoints according to how much they reduce the posterior process entropy.

GPLVM: Optimization

- In contrast to linear models such as PCA and PPCA, which allow for a closed-form solution to determine the latent variables (or projections in feature space), GP-LVM requires to proceed iteratively through gradient-based optimization.

- Therefore, a smart initialization is important.

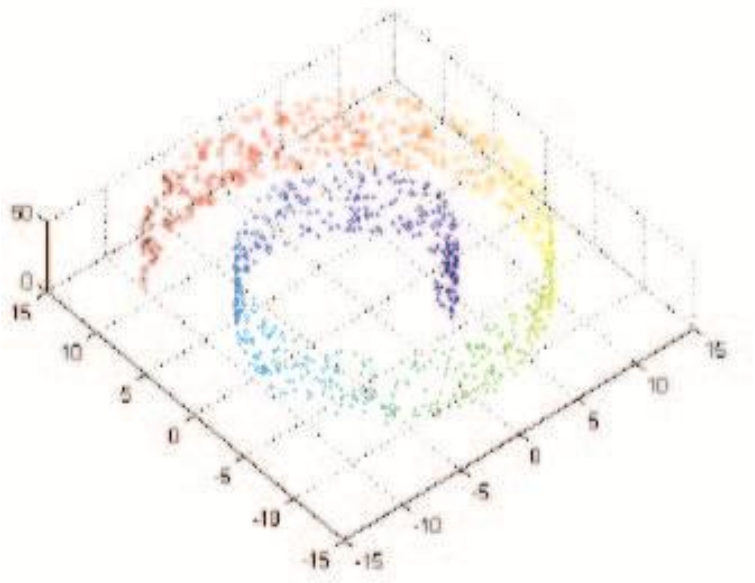
- Usually initialization with PCA works fine…

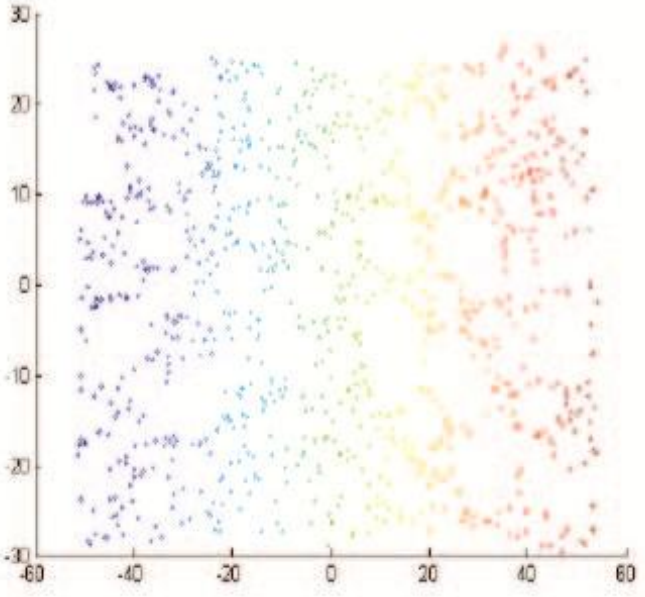
GPLVM: Initialization

But not always…

We have already seen the ‘swiss roll’ dataset, when discussed kPCA.

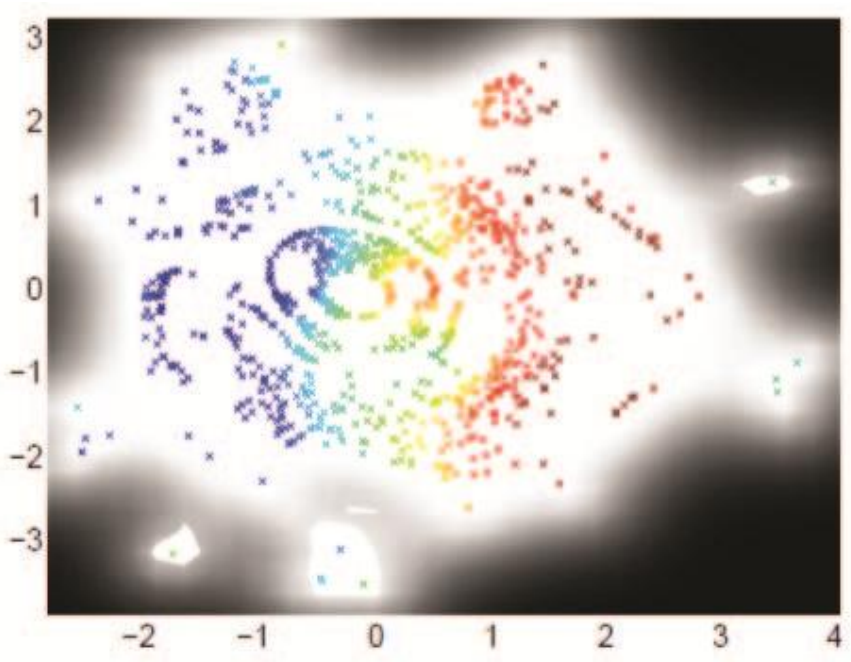
For this data the true structure is known: the manifold is a two-dimensional square twisted into a spiral along one of its dimensions and living in a three-dimensional space.

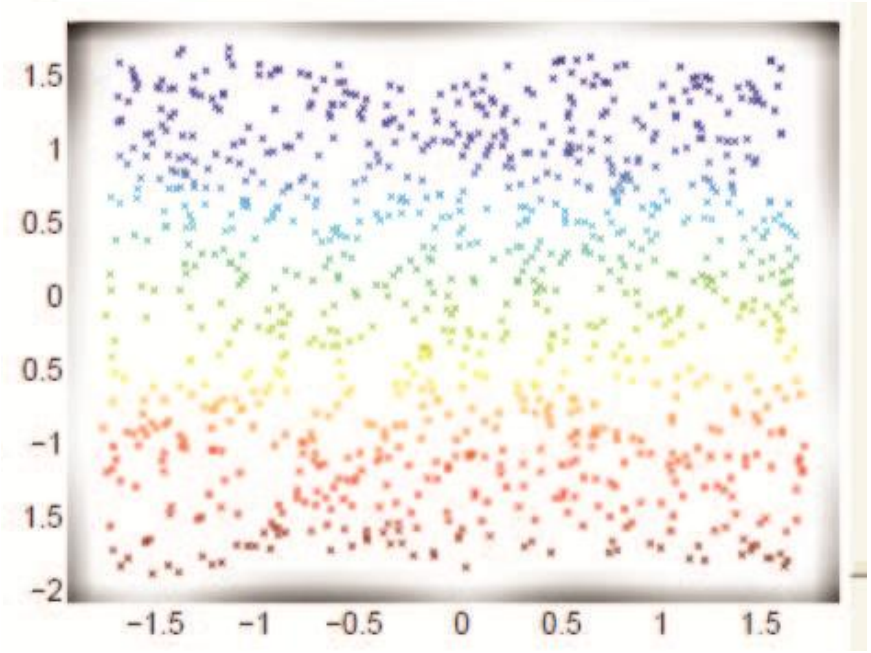




GPLVM: Initialization

- GP-LVM with PCA initialization gives poor results.
- Initialization with Isomap allows to restore the original structure of the data.





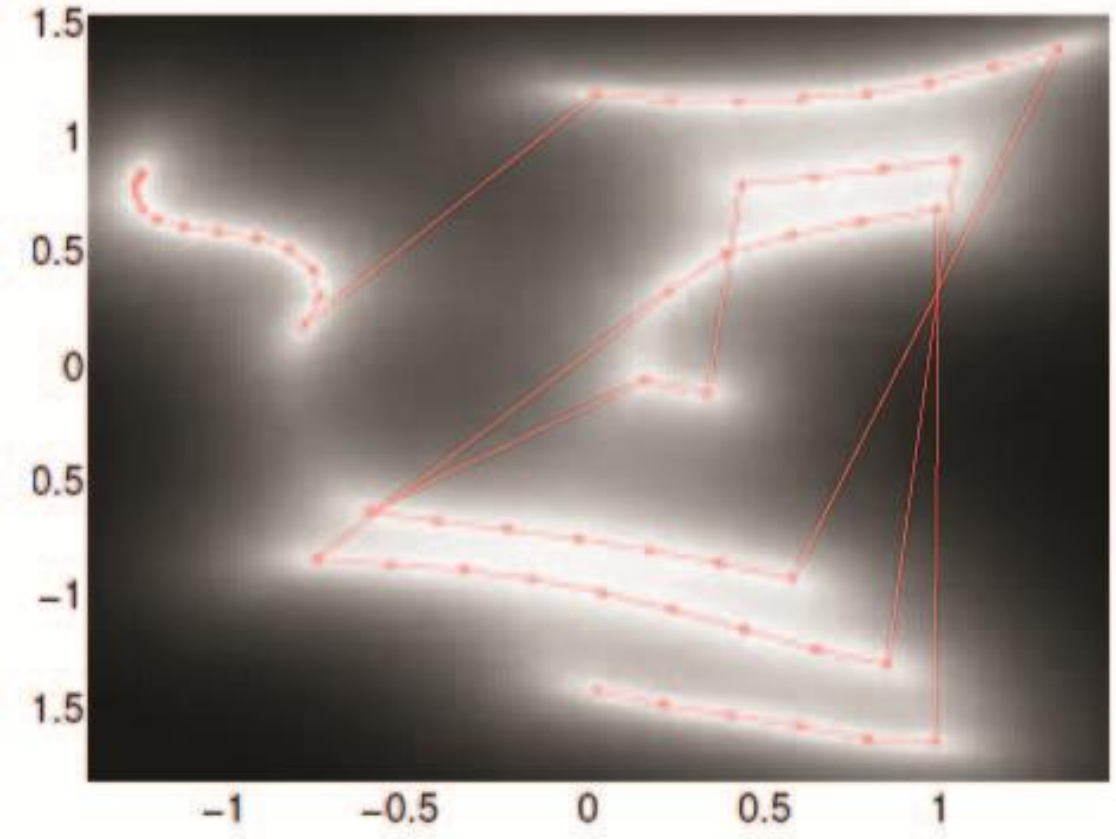
Locality Preservation

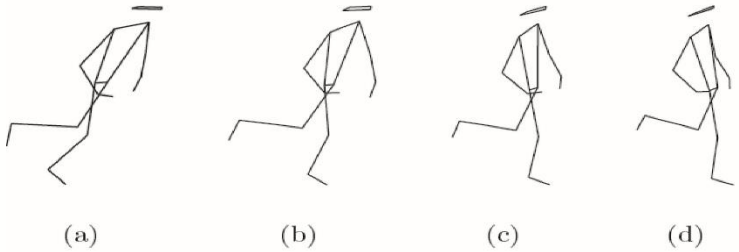
Most dimensionality reduction techniques preserve local distances

- kPCA maps smoothly from data in original space to latent space and points close in the data space are close in the latent space but points close in the latent space may be not close in the data space

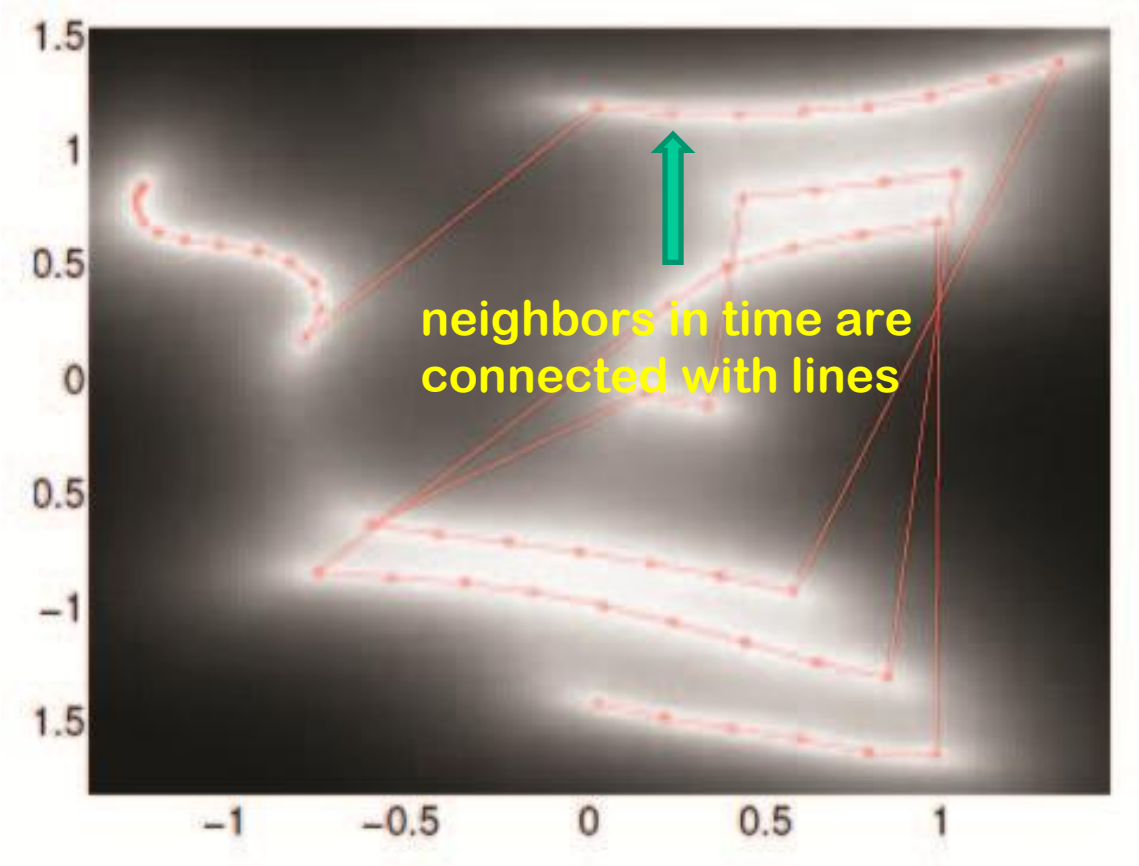
- GP LVM does not preserve local distances but points close in the latent space are close in the data space

Motion capture data of human walking. The paths of the sequences in the latent space are shown as solid lines. The dimension of the original data is 102 (34 markers x 3 coordinates).

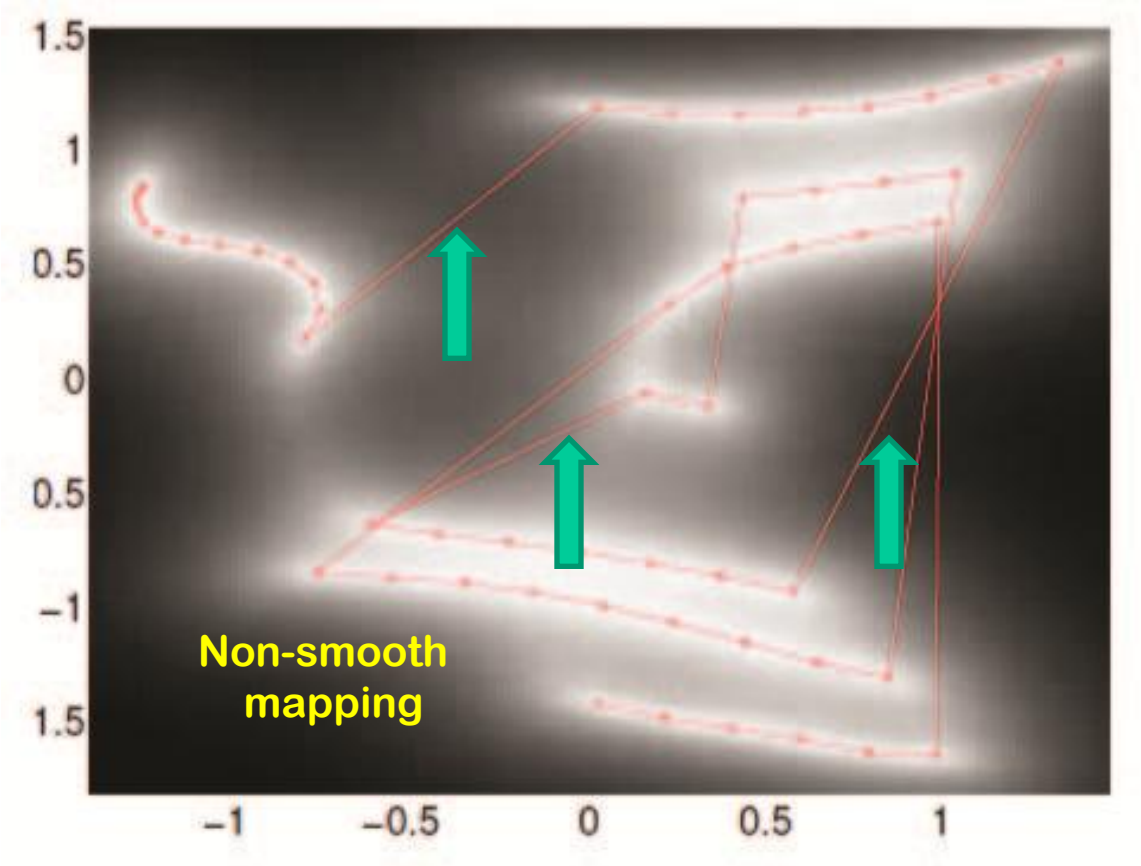




The data obtained from a subject breaking into a run from standing – cyclic motion.



We are supposed to see a smooth periodic pattern in the latent space. But, instead …



- Lowe and Tipping [1997] made **latent** positions a **function** of the data.

$$x_{ij} = f_j(\mathbf{y}_i, \mathbf{w})$$

- Function was either a multi-layer perceptron or a radial basis function network.

- The same idea can be used to force the GP-LVM to respect local distances [Lawrence and Candela, 2006].

By constraining each $\mathbf{x}_i$ to be a smooth mapping from $\mathbf{y}_i$ local distances can be respected

- This works because in GP-LVM, one maximizes w.r.t. the latent variables and does not integrate these out.

- GP-LVM normally proceeds by optimizing

$$L(\mathbf{X}) = \log p(\mathbf{Y} \mid \mathbf{X})$$

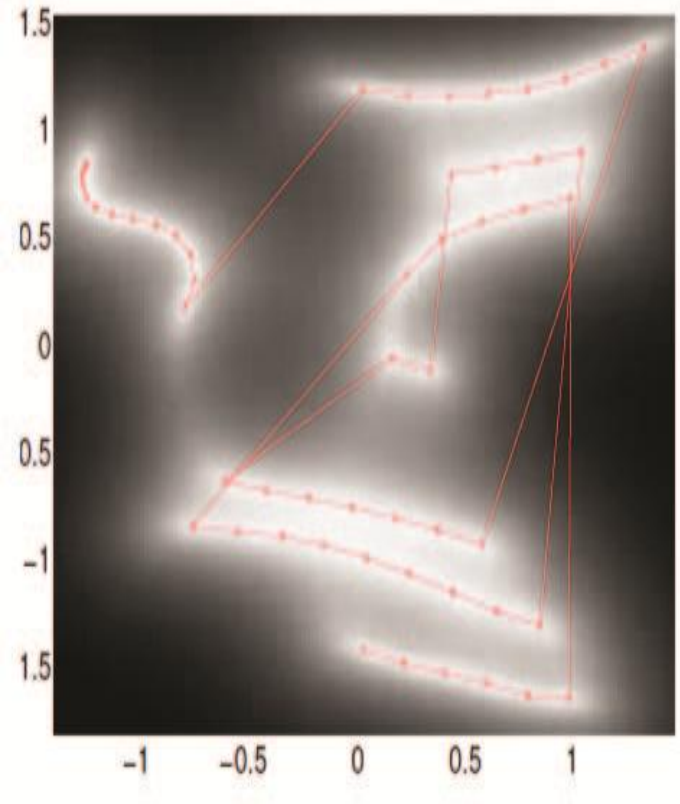
with respect to $\mathbf{X}$ using $\frac{\partial L}{\partial \mathbf{X}}$

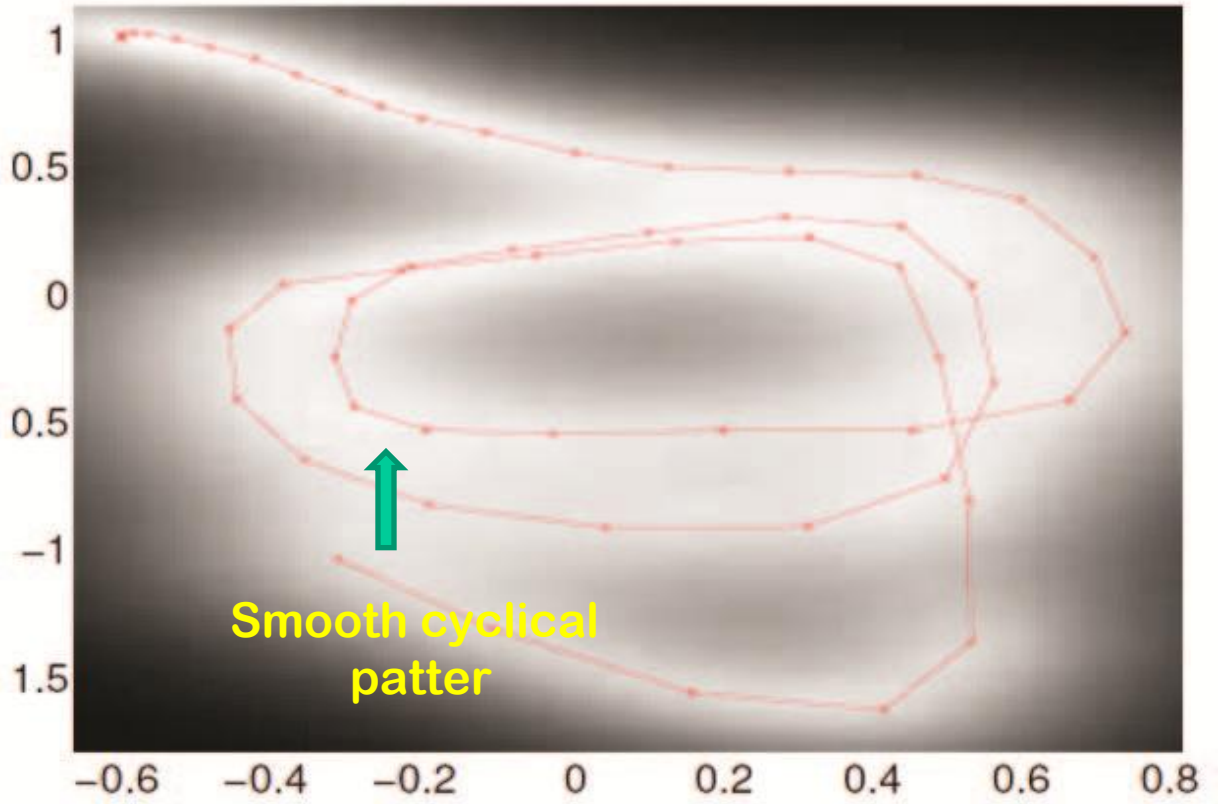
- The back constraints are of the form

$x_{ij} = f_j(\mathbf{y}_{i,:}, Q)$ where Q are a set of unknown parameters

- We can compute $\frac{\partial L}{\partial Q}$ via the chain rule and optimize

GP-LVM with the back constraints applied to the runner dataset , that we have seen before.

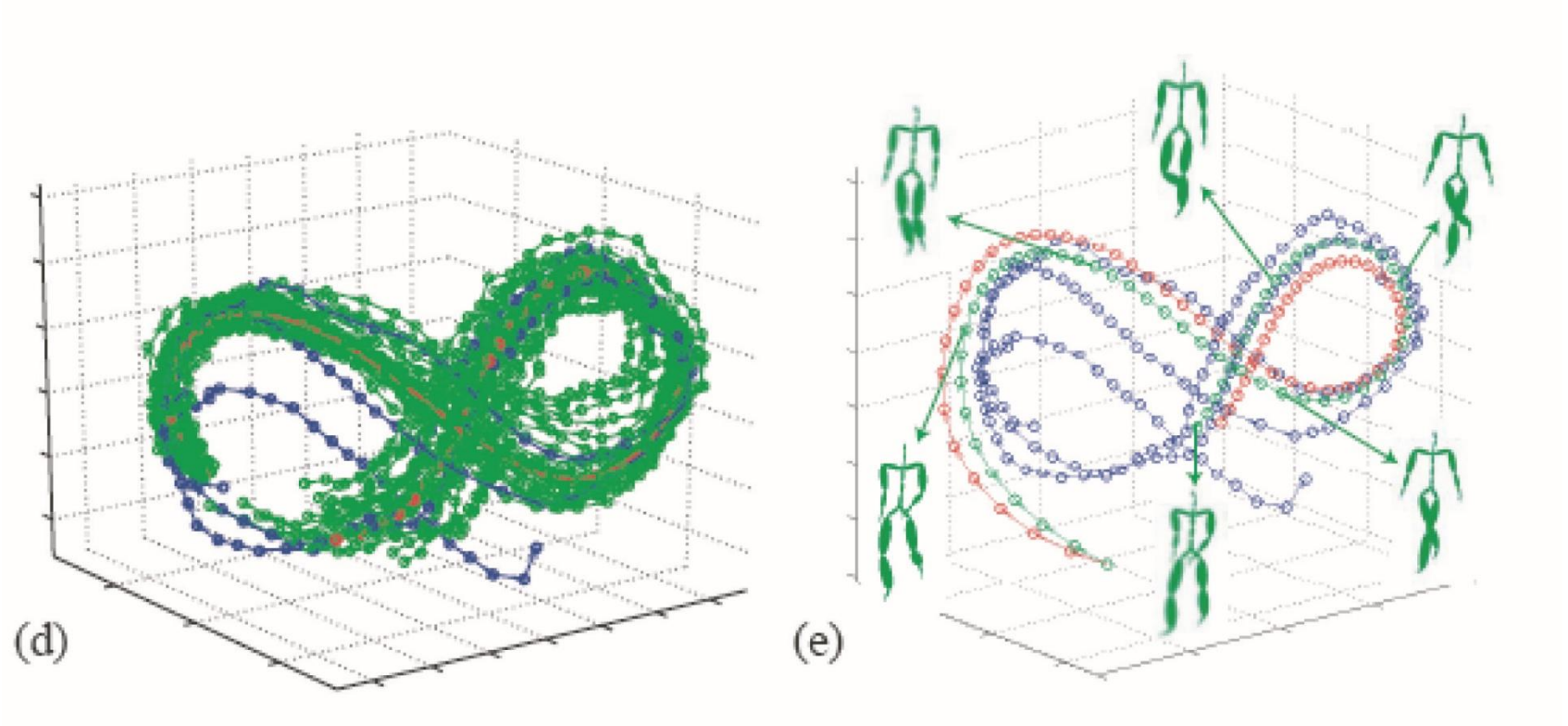




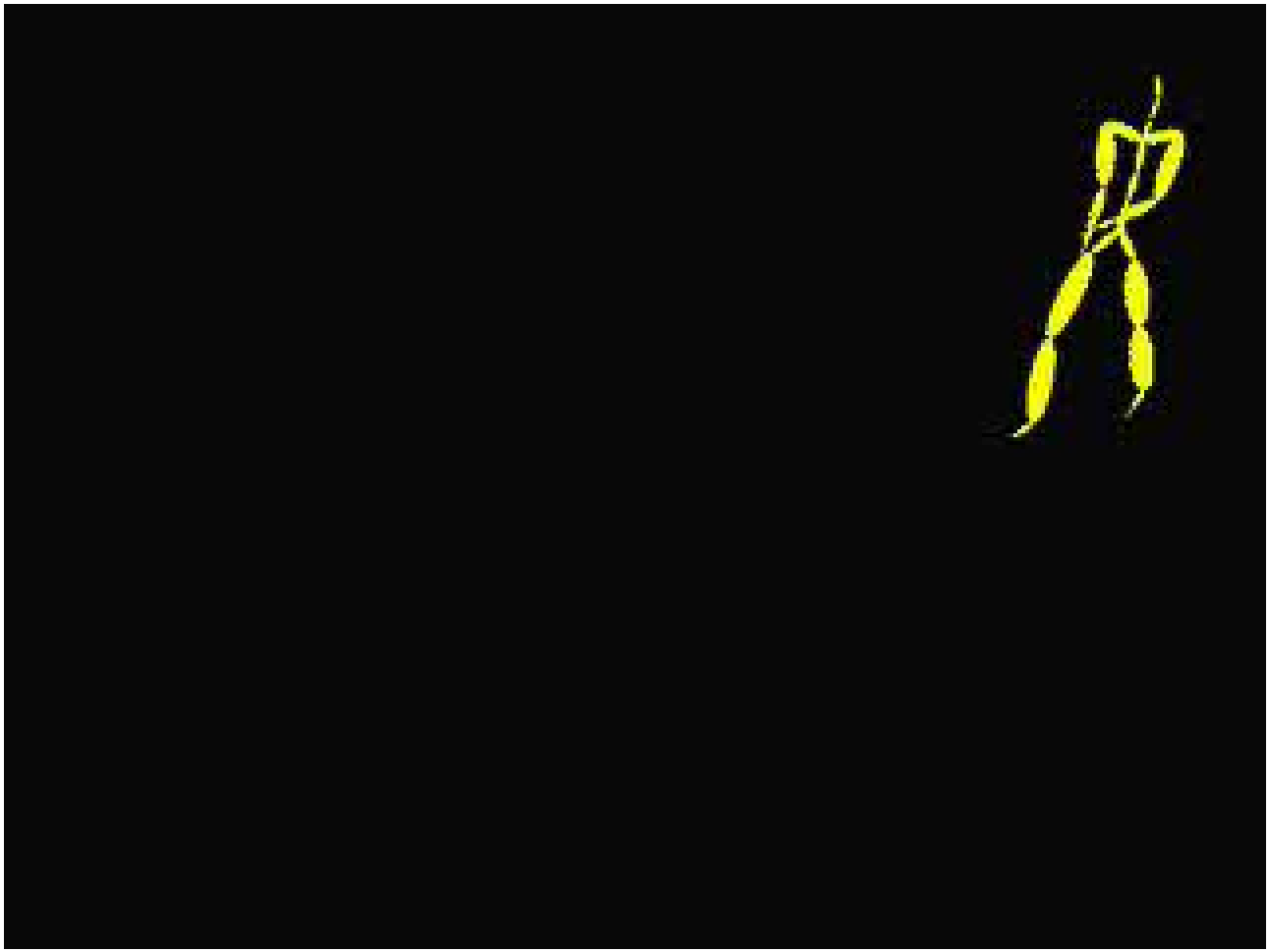
Gaussian Process Latent Variable Models

From PCA to Probabilistic PCA

M datapoints. Notation: **Y** data in original space and **X** projections in latent space



Missing Data



Recovered

Gaussian Process

Discussion

Advantages:

- ❑ Generalizations
- ❑ Accuracy
- ❑ Estimation of predictions' uncertainty
- ❑ Auto-tuning of hyper-parameters

Disadvantages:

- ❑ Computational complexity

Implementations in Python:

- Gpy
- Scikit-learn

Comparison Across Ridge Regression, GPR and RVR

Closing words on Regression

Quick Recap of Relevance Vector Regression

- Considers each $y_i := y(x_i)$ and $y_i \sim \mathcal{N}(\phi_i(x_i)\alpha_i, \sigma^2)$ and collectively,

$$p(Y|\boldsymbol{\alpha}, \sigma) = \mathcal{N}(\Phi\boldsymbol{\alpha}, \sigma^2)$$

- The function $\Phi = [\phi_1, \ldots, \phi_M] \in \mathbb{R}^{M\times(M+1)}$, where each $\phi_i = [1\ k(x_i, x_1), \ldots,\ k(x_i, x_M)] \in \mathbb{R}^{M+1}$, $\boldsymbol{\alpha} = [\alpha_0, \ldots, \alpha_M]^\top$

- Considers $\alpha_i \sim \mathcal{N}(0, s_i^{-1})$, k is the kernel function.

- Prediction likelihood of output is also Gaussian:

$$p(y|Y) = \int p(y|\boldsymbol{\alpha}, \boldsymbol{s}_{MAP}, \boldsymbol{\sigma}_{MAP}) p(\boldsymbol{\alpha}|Y, \boldsymbol{s}_{MAP}, \boldsymbol{\sigma}_{MAP}) \mathrm{d}\boldsymbol{\alpha}$$

- $p(y|Y) \sim \mathcal{N}(\mu_*, \sigma_*^2)$, with

$$\mu_* = \sigma_{MAP}^{-2}\phi(x)\Sigma_{MAP}\Phi\phi^\top(x)Y, \quad \sigma_*^2 = \sigma_{MAP}^2 + \phi(x)\Sigma_{MAP}\phi^\top(x)$$

$$\Sigma = (\sigma^{-2}\Phi^\top\Phi + S)^{-1}$$

- Marginal likelihood of output in RVR (also a convolution integral):

$$p(Y|\boldsymbol{s}, \sigma^2) = \int p(Y|\boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{\alpha}|\boldsymbol{s}, \sigma^2) \mathrm{d}\boldsymbol{\alpha} = \mathcal{N}(0, \sigma^2 I + \Phi S^{-1}\Phi^\top)$$

Marginal likelihood derivation

Recall that convolution of two Gaussian distributions is also a Gaussian distribution:

$$\text{Given } p_1(x) = \mathcal{N}(a, A); \quad p_2(x) = \mathcal{N}(b, B),$$

$$\text{We have, } p_1 \star p_2(z) = \int p(z-x)p(x)\mathrm{d}x = \mathcal{N}(a+b, A+B) \qquad (1)$$

Simplifications:

$$p(Y|\boldsymbol{\alpha}, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{\left\|Y - \Phi^\top \boldsymbol{\alpha}\right\|^2}{2\sigma^2}\right) \implies \boxed{p(Y - \Phi^\top \alpha) = \mathcal{N}(0, \sigma^2 I)}$$

and since $p(\alpha_i) = \mathcal{N}(0, s_i^{-1})$,

$$p(\boldsymbol{\alpha}) = \mathcal{N}(0, S^{-1}) \implies \boxed{p(\Phi^\top \boldsymbol{\alpha}) = \mathcal{N}(0, \Phi S^{-1} \Phi^\top)}$$

By definition,

$$\begin{aligned} p(Y|\boldsymbol{s}, \sigma^2) &= \int p(Y|\boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{\alpha}|\boldsymbol{s}, \sigma^2) \mathrm{d}\boldsymbol{\alpha} \\ &= p(Y - \phi^\top \alpha) \star p(\Phi^\top \boldsymbol{\alpha}) \\ &= \mathcal{N}(0, \sigma^2 I + \Phi S^{-1} \Phi^\top) \qquad \text{Follows from (1)} \end{aligned}$$

Quick Recap of Gaussian Process Regression

- Considers each $y_i := y(x_i)$ and $y_i \sim \mathcal{N}(0, k(x_i, x_i))$
- $k(x_i, x_j)$ is the symmetric positive definite kernel (or covariance) function.
- All y_i are jointly Gaussian:

$$p(Y) = \mathcal{N}(0, K(X, X)), \quad K(X, X) = \begin{pmatrix} k(x_1, x_1) \dots k(x_1, x_M) \\ \ddots \\ k(x_1, x_M) \dots k(x_M, x_M) \end{pmatrix} \quad (1)$$

  - $K(X, X) \in \mathbb{R}^{M \times M}$ is mathematically similar to Φ in RVR (which is in $\mathbb{R}^{M \times (M+1)}$)
  - Covariance function specifies a distribution over functions, each sample of distribution in (1) is a function!
  - Prediction by maximising marginal likelihood:

$$p(y(x)|Y) = \mathcal{N}(\underbrace{\boldsymbol{k}(x, X)K(X, X)^{-1}Y}_{\text{Mean}}, \underbrace{k(x, x) - \boldsymbol{k}(x, X)K(X, X)^{-1}\boldsymbol{k}^\top(x, X)}_{\text{Variance}}) \quad (2)$$

where $\boldsymbol{k}(x, X)$ is a row vector $[k(x, x_1) \dots k(x, x_M)]$.

Discussion on GPR

- If noise is assumed on output $y_i = y(x_i) + \xi, \xi_i \sim \mathcal{N}(0, \sigma^2)$, change $K(X, X)$ to $K(X, X) + \sigma^2 I$ in Eqn (2)
- Covariance function must be symmetric and lead to a positive semi definite covariance matrix across all points on the range (satisfy Mercer's theorem)
- Covariance function may be decomposed to continuous an orthonormal basis of Eigen functions e_i with eigenvalues λ_i

$$K(X, X) = \sum_{i=1}^{\infty} \lambda_i e_i(X) e_i^\top(X), \quad \lambda_i \geq 0$$

$$\implies Ke_i = \lambda_i e_i \implies e_i^\top K^{-1} e_i = \lambda_i^{-1}$$

- Eigen functions of square exponential kernel are periodic curves: λ_i increases with frequency
- Prior likelihood on Y prefers Eigen functions with high Eigenvalues (smooth functions):

$$p(Y) = (2\pi)^{-M/2} (Det(K))^{-1/2} exp(- \underbrace{Y(K(X, X))^{-1} Y}_{\text{Estimator is biased to its lower values}})$$

- GP can be approximated by retaining Eigen functions corresponding to high Eigenvalues.

RVR converted to GP

- RVR can be thought of as a GP with a data dependent prior
- Marginal likelihood of output in RVR:

$$p(Y|\boldsymbol{s},\sigma^2) = \mathcal{N}(0,\sigma^2 I + \Phi S^{-1}\Phi^\top), \quad S = diag(s_0,\ldots,s_M)$$
$$= \mathcal{N}(0,\sigma^2 I + Q(X,X))$$

- $Q(X,X) = \Phi S^{-1}\Phi^\top$
- Therefore kernel of RVR converted to GP is symmetric, positive definite for any chosen kernel function $k(x,y)$.
- Predicted variance is similar to GP: confident close to given data points, high values of variance away from the data set.

Comparison of RVR and GPR

RVR	GPR
Based on Bayesian inference	Based on Bayesian inference
Prediction is a linear function of Y. has the form: $f(x, X)Y$, $f(x, X) = \sigma_{MAP}^{-2}\phi(x)\Sigma_{MAP}\Phi(X)\phi^{\top}(x)$	Prediction has the form $y(x) = \sum_{I=1}^{M} g(x, X)Y$, where $g(x, X) = \boldsymbol{k}(x, X)K(X, X)^{-1}Y$
Data dependent Prior (arising from SVM)	Prior not necessarily data dependent
Hyper parameters are $\boldsymbol{s}$ and σ^2	Hyper parameter is kernel width of output probability distribution
Kernel need not satisfy Mercer's condition	Kernel must satisfy Mercer's condition
Automatically reduces number of basis functions as some of them are rejected: (low computation cost)	Not so useful kernel basis functions are not automatically rejected

Recap of GP-LVM

- Given dataset: (X, Y), objective is to find the embedding of X in a lower dimensional space.
- Consider a prior on weights: $P(W) = \mathcal{N}(0, I)$
  - Assume X is a latent variable (hidden)
  - For starters, assume $Y = W^\top X$
  - Assume a prior distribution on Y

$$P(Y|W, X) = \mathcal{N}(W^\top X, \beta I)$$

- Marginalization of parameter W leads to:

$$P(Y|X) = \int P(Y|W, X)P(W)\mathrm{d}W = \mathcal{N}(0, \underbrace{XX^\top + \beta^{-1}I}_{\text{Same as RVR considered as a GP!}})$$

- Maximize the log likelihood $L(X)$ to obtain X as:

$$X = ULV^\top, \quad U \in \mathbb{R}^{M \times q}, \quad L = diag(l_j) \in \mathbb{R}^{q \times q}, \quad l_j = (\lambda_j - \beta^{-1})^{-1}$$

λ_j is the eigenvalue associated with the jth eigenvector of $D^{-1}YY^\top$, V is an arbitrary $q \times q$ rotation matrix.

Kernels in GP-LVM

- GP LVM Reduces the dimension of input data by treating them as latent variables
- Projection to feature space through nonlinear map possible:

$$XX^\top + \beta^{-1}I \text{ is replaced by } K(X,X) + \beta^{-1}I$$

- $L(K(X))$ is the modified log likelihood which cannot be optimised to get a closed form solution
- Gradients w.r.t kernel parameters must be computed and used to jointly optimise them along with X In RVR only kernel parameters need to be optimized
- Probabilistic PCA $\rightarrow$ Assuming a linear kernel $K(X,X) = XX^\top$

Comparison across non-linear regression techniques seen in class

SVR, RVR and GPR are based on the same regressive model.

The regressive solution has the same form:

$$y(x) = f(x) = \sum_{i=1}^{M} \alpha_i k(x, x^i)$$

All techniques estimate the α.

Optimization for GPR can estimate the kernel parameters.

While GPR uses all datapoints, SVR/RVR select a subset of datapoints with non-zero α.

Different optimization functions $\Rightarrow$ Different results

Comparison across non-linear regression techniques seen in class

- The techniques differ not only in their algorithm, but also in the number of hyperparameters and how easy it is to define them.
  - Some have bounded hyperparameters, others not.
  - Some hyperparameters have a geometrical interpretation, others not.
- Some techniques (GPR, RVR) provide a metric of uncertainty of the model, which can be used to determine when inference is trustable.
- Some techniques (ϖ-SVR, RVR) are designed to be computationally cheap at retrieval (very few support vectors, few models).
- Other techniques (GPR) are meant to provide very accurate estimate of the data, at the cost of retaining all datapoints for retrieval.

Comparison

No easy way to determine which regression technique fits best your problem. Here are only a few metrics. Other aspects matters, e.g. precision of estimate, ease to determine the hyperparameters, availability of code.

Computational Costs at Training

Computational steps

SVR / ϖ-SVR

Convex optimization
(SMO solver)
Parameters grow O(M*N)

RVR

Iterative optimization
Memory: $O(NM^2)$
Computation: $O(NM^3)$

GMR

EM, iterative technique,
needs several runs
Parameters grow
$O(K*N^2)$

Boosting/Bagging

Iterative technique, speed at
training depends on complexity
of basic models, usuall $O(NM^2)$

GPR

Inversion of huge matrix $O(NM^3)$
Iterative technique for learning
parameters, but done once only

Growth of costs with size of data

M: number of datapoints; N: Dimension of data; K: Number of basis functions

Comparison

Computational Costs at Testing

Computational steps

Boosting/Bagging

Fast if model simple

SVR

Growths with
O (number of SV) ~ O(M)
SV - Small fraction of original data, very small with RVR)

GMR

O(N^2K), K small

RVR / ϖ-SVR

Growths with
***O (M) but* very small number of SV-s)**

GPR

O(M^3)

Growth of costs with size of data

M: number of datapoints; N: Dimension of data; K: Number of Gauss Functions in GMM model / of RBF in LWPR