An introduction to Gaussian processes

Oliver Stegle and Karsten Borgwardt

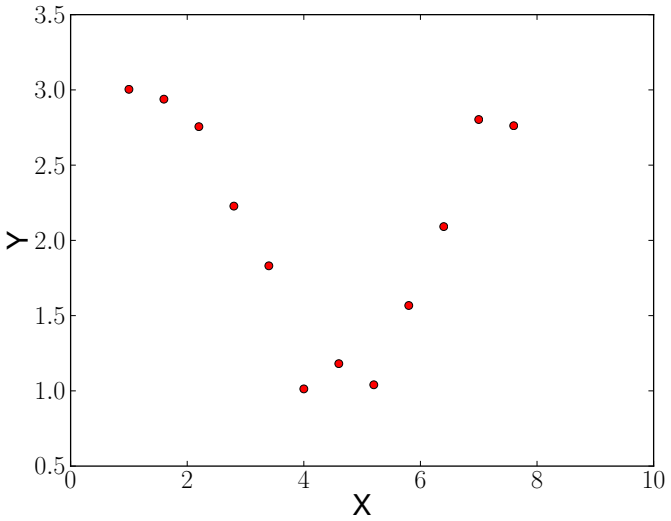
Machine Learning and
Computational Biology Research Group,
Max Planck Institute for Biological Cybernetics and
Max Planck Institute for Developmental Biology, Tübingen

Why Gaussian processes?

- So far: linear models with a finite number of basis functions, e.g. $\boldsymbol{\phi}(x) = (1, x, x^2, \ldots, x^K)$
- Open questions:
  - How to design a suitable basis?
  - How many basis functions to pick?
- Gaussian processes: accurate and flexible regression method yielding predictions alongside with error bars.

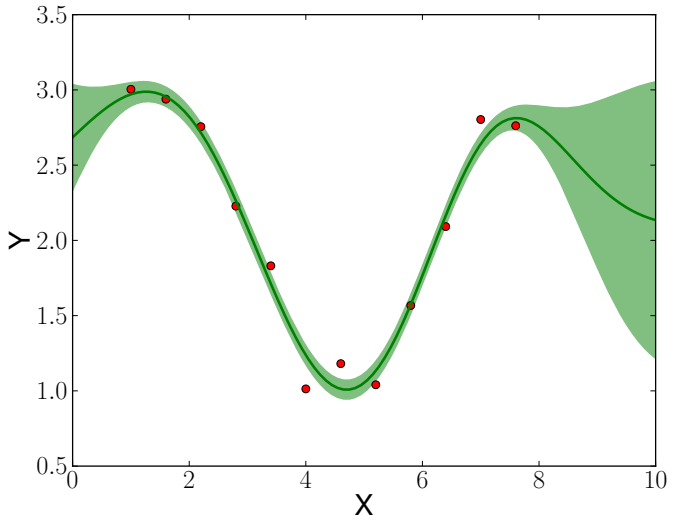
Why Gaussian processes?

- So far: linear models with a finite number of basis functions, e.g. $\boldsymbol{\phi}(x) = (1, x, x^2, \ldots, x^K)$
- Open questions:
  - How to design a suitable basis?
  - How many basis functions to pick?
- Gaussian processes: accurate and flexible regression method yielding predictions alongside with error bars.



Why Gaussian processes?

- So far: linear models with a finite number of basis functions, e.g. $\boldsymbol{\phi}(x) = (1, x, x^2, \ldots, x^K)$
- Open questions:
  - How to design a suitable basis?
  - How many basis functions to pick?
- Gaussian processes: accurate and flexible regression method yielding predictions alongside with error bars.



Further reading

- A comprehensive and very good introduction to Gaussian processes
  C. E. Rasmussen, C. K. Williams
  Gaussian proceesses for machine learning
  - Free download: `http://www.gaussianprocess.org/gpml/`
- A really good introductory movie to watch
  `http://videolectures.net/gpip06_mackay_gpb/`
  - Many ideas used in this course are borrowed from this lecture.

Outline

Outline

The Gaussian distribution

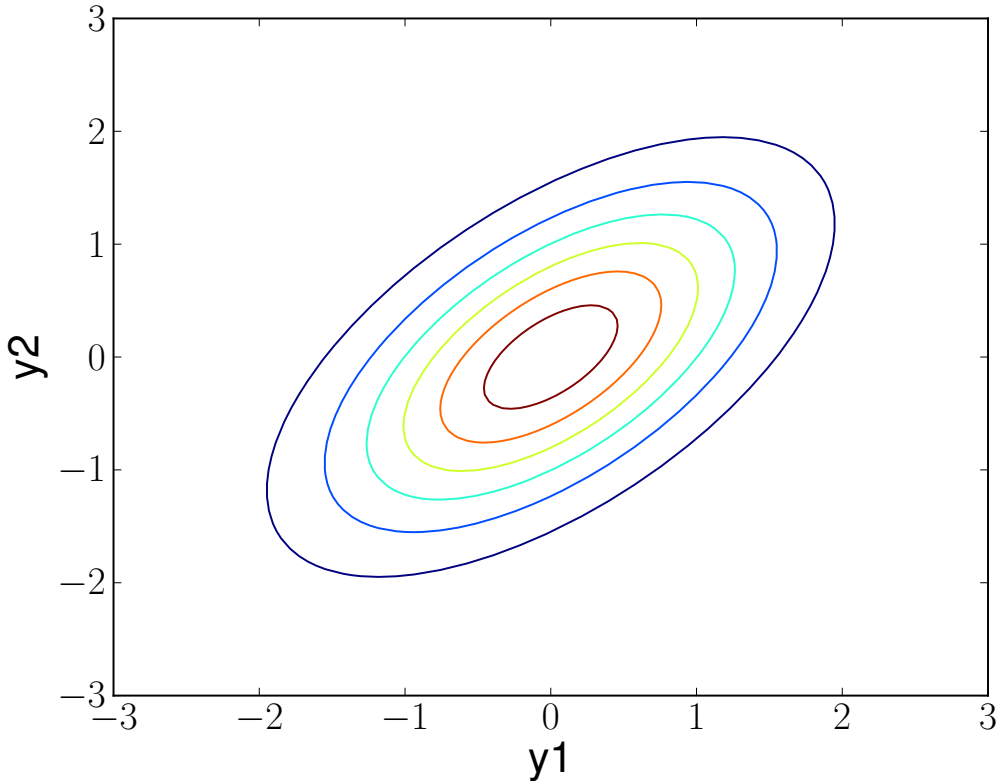
- Gaussian processes are merely based on the good old Gaussian

$$\mathcal{N}\left(\mathbf{x}\,\middle|\,\boldsymbol{\mu}, \mathbf{K}\right) = \frac{1}{\sqrt{|2\pi\,\mathbf{K}|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\,\mathbf{K}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

- Covariance matrix or kernel matrix

A 2D Gaussian

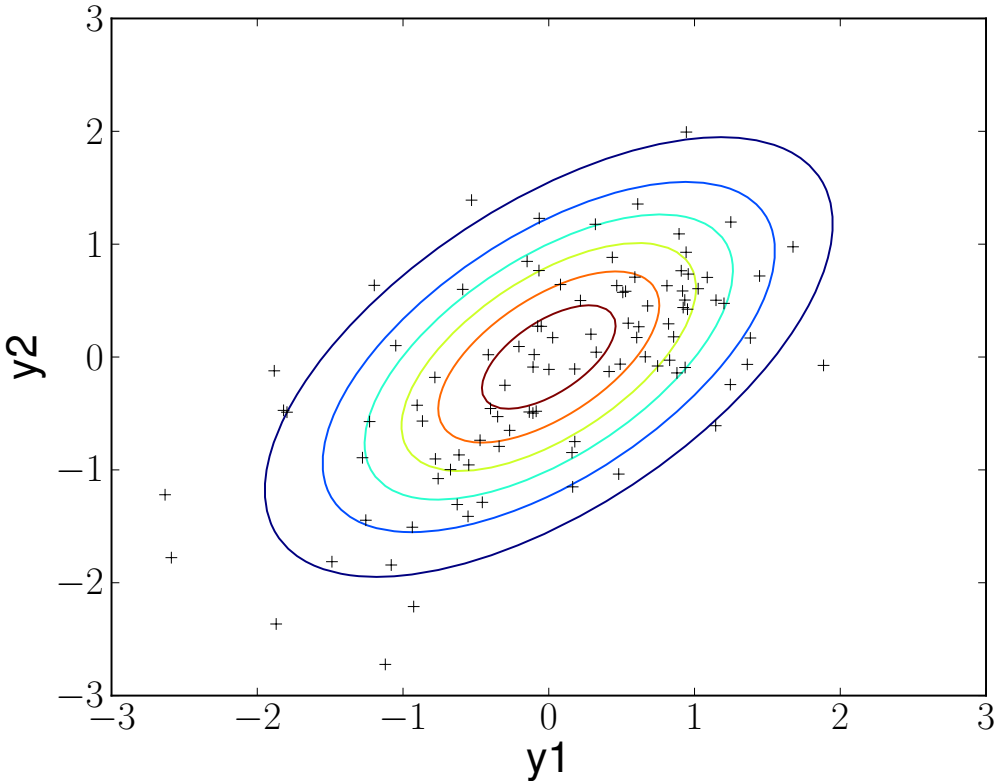
- Probability contour
- Samples



$$\mathbf{K} = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$$

A 2D Gaussian

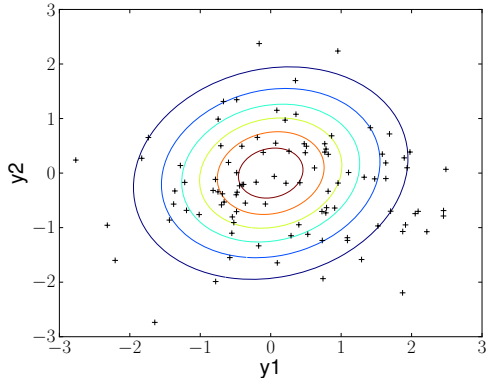
- Probability contour
- Samples



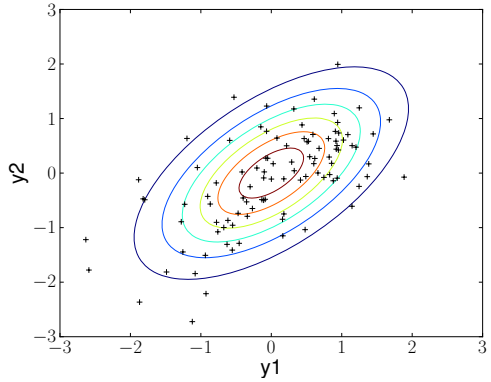
$$\mathbf{K} = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$$

A 2D Gaussian

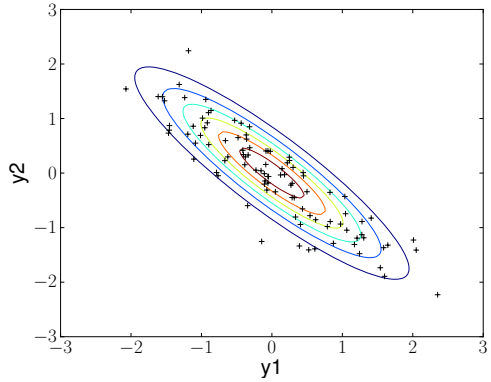
Varying the covariance matrix



$$\mathbf{K} = \begin{bmatrix} 1 & 0.14 \\ 0.14 & 1 \end{bmatrix}$$



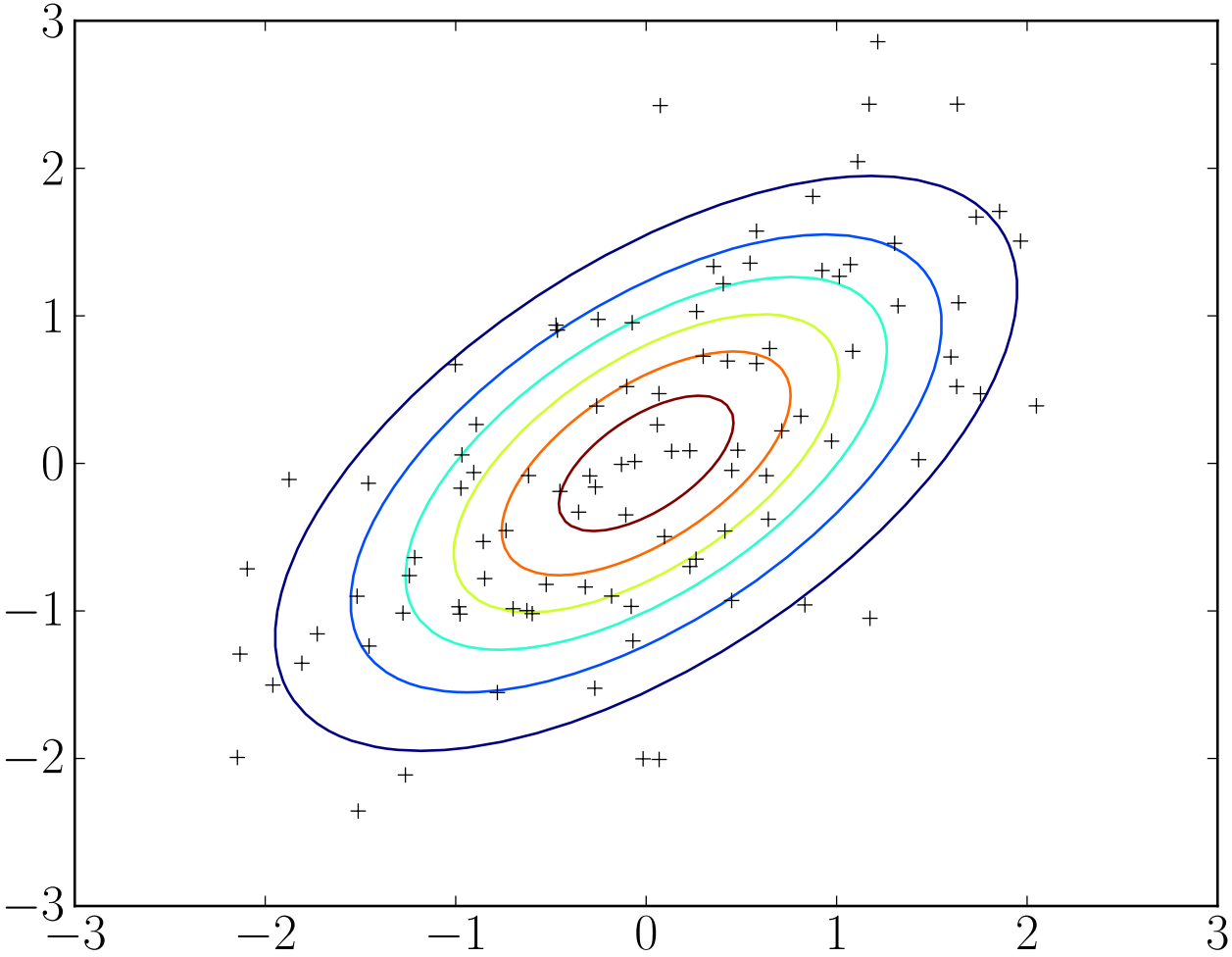
$$\mathbf{K} = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$$



$$\mathbf{K} = \begin{bmatrix} 1 & -0.9 \\ -0.9 & 1 \end{bmatrix}$$

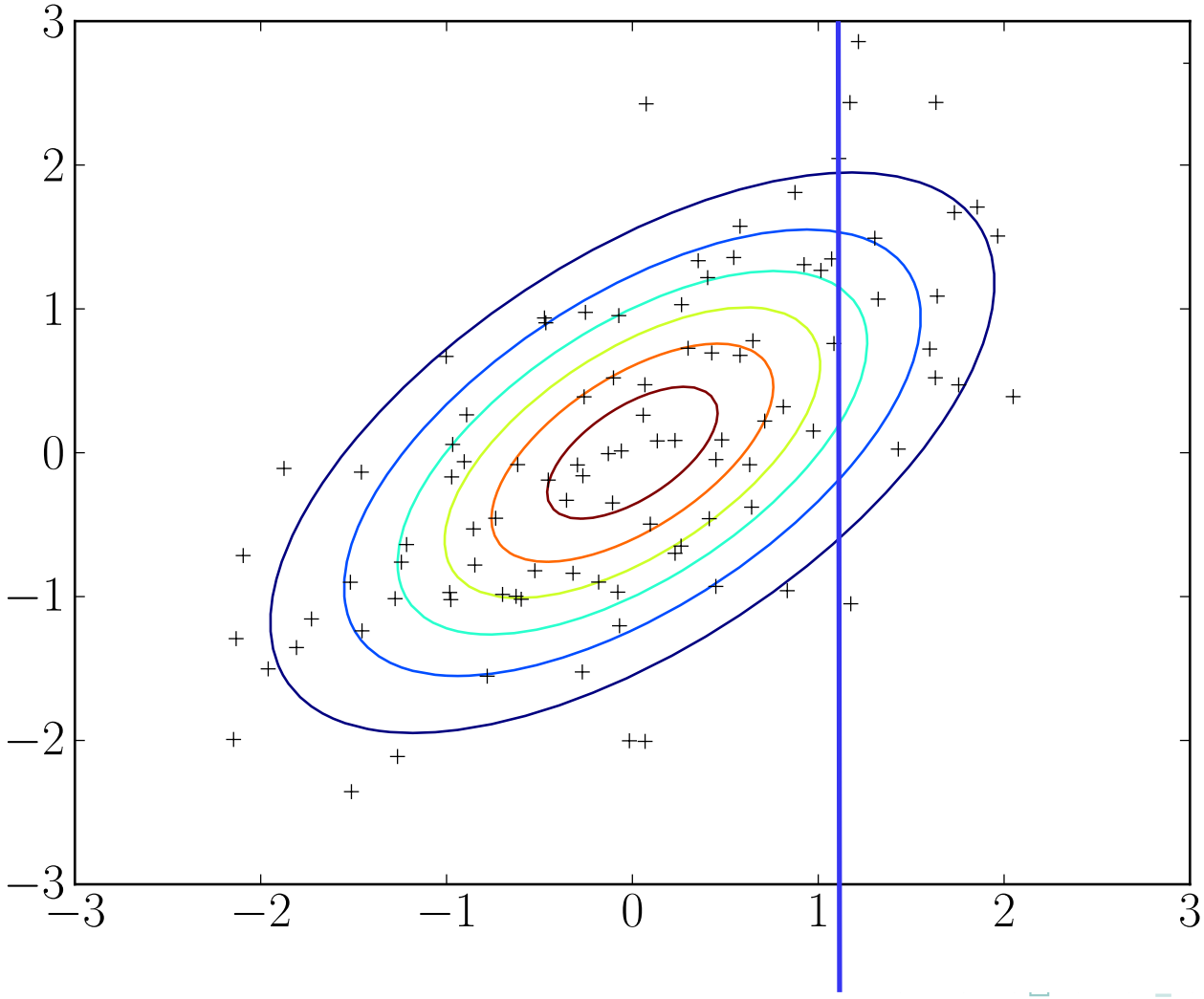
A 2D Gaussian

Inference



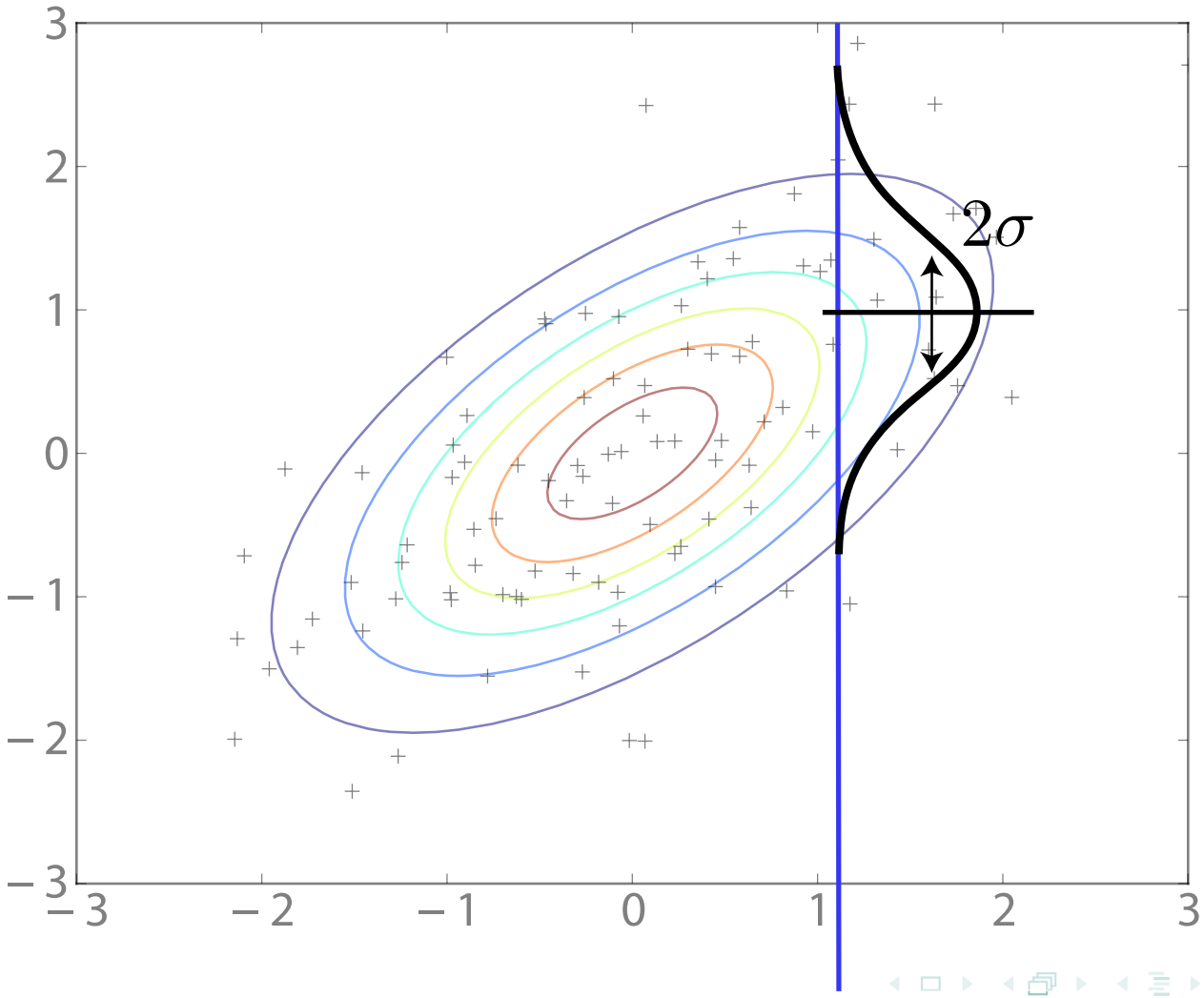
A 2D Gaussian

Inference



A 2D Gaussian

Inference



Inference

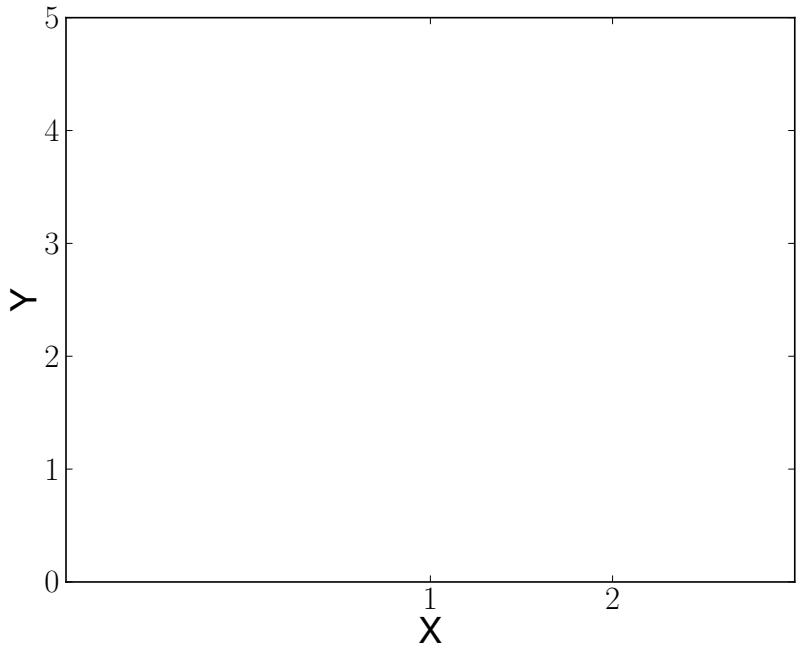
- Joint probability $p(y_1, y_2 \,|\, \mathbf{K}) = \mathcal{N}\left([y_1, y_2] \,|\, \mathbf{0}, \mathbf{K}\right)$
- Conditional probability

$$p(y_2 \,|\, y_1, \mathbf{K}) = \frac{p(y_1, y_2 \,|\, \mathbf{K})}{p(y_1 \,|\, \mathbf{K})}$$
$$\propto \exp\left\{-\frac{1}{2}[y_1, y_2]\,\mathbf{K}^{-1}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right\}$$

- Completing the square yields a Gaussian with non-zero as posterior for y_2.

Extending the idea to higher dimensions

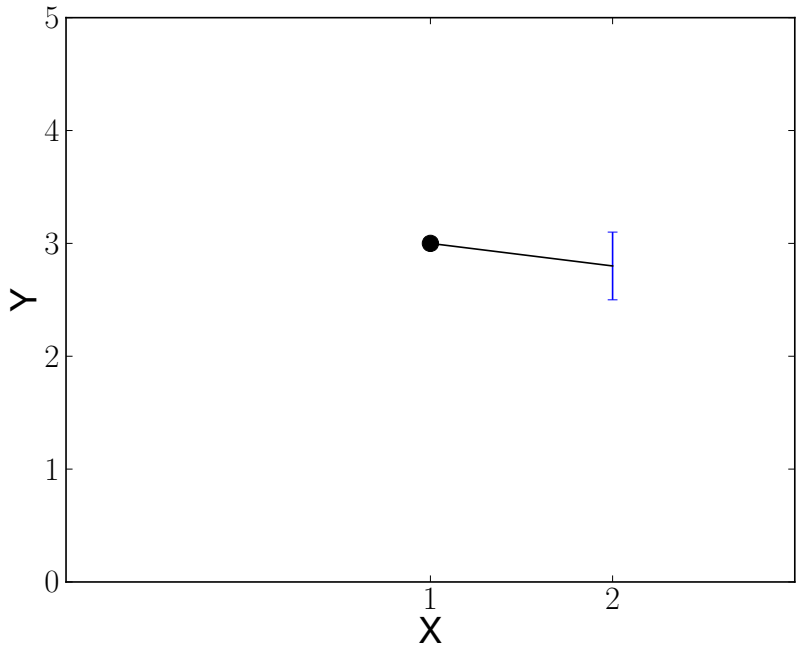
- Let us interpret y_1 and y_2 as outputs in a regression setting.
- We can introduce an additional 3rd point.



- Now $P([y_1, y_2, y_3] \,|\, \mathbf{K}_3) = \mathcal{N}([y_1, y_2, y_3] \,|\, \mathbf{0}, \mathbf{K}_3)$, where $\mathbf{K}_3$ is now a 3 x 3 covariance matrix!

Extending the idea to higher dimensions

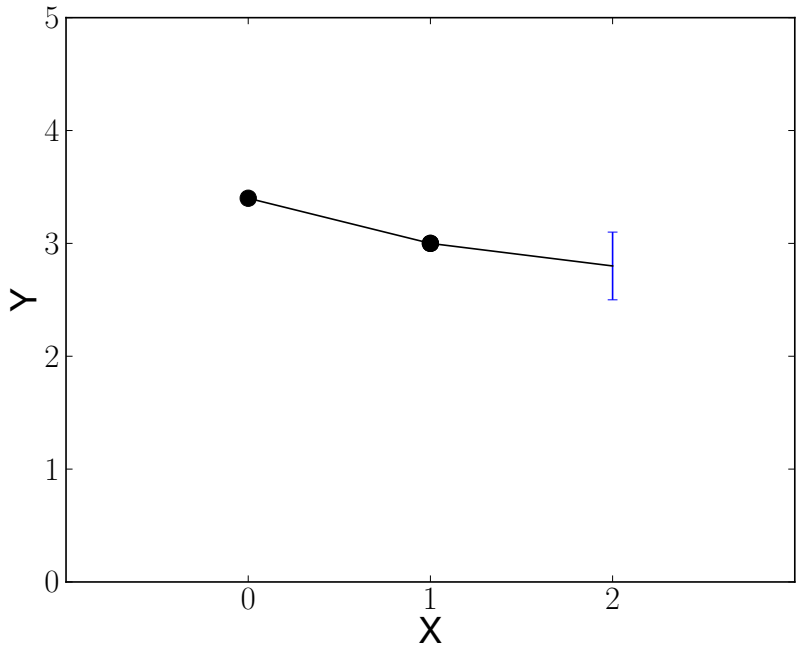
- Let us interpret y_1 and y_2 as outputs in a regression setting.
- We can introduce an additional 3rd point.



- Now $P([y_1, y_2, y_3] \,|\, \mathbf{K}_3) = \mathcal{N}([y_1, y_2, y_3] \,|\, \mathbf{0}, \mathbf{K}_3)$, where $\mathbf{K}_3$ is now a 3 x 3 covariance matrix!

Extending the idea to higher dimensions

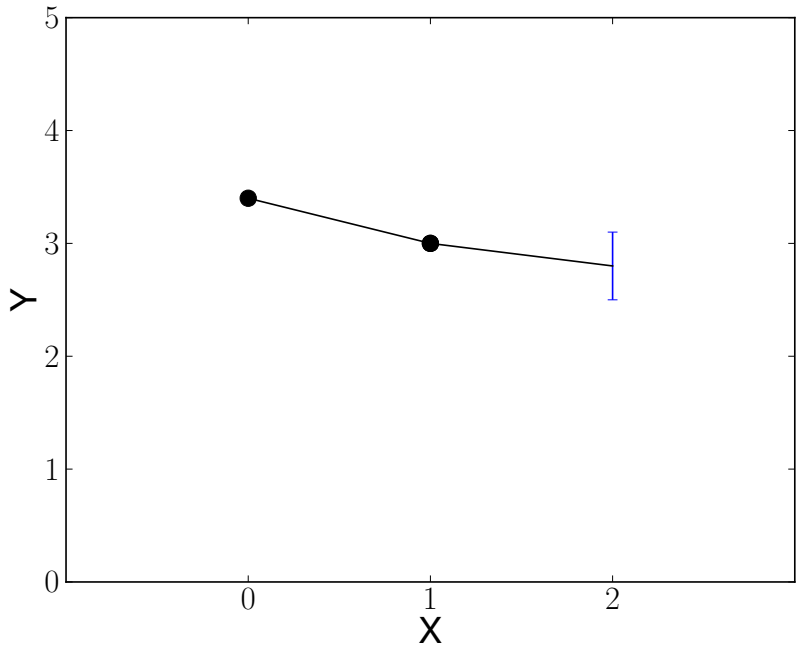
- Let us interpret y_1 and y_2 as outputs in a regression setting.
- We can introduce an additional 3rd point.



- Now $P([y_1, y_2, y_3] \,|\, \mathbf{K}_3) = \mathcal{N}([y_1, y_2, y_3] \,|\, \mathbf{0}, \mathbf{K}_3)$, where $\mathbf{K}_3$ is now a 3 x 3 covariance matrix!

Extending the idea to higher dimensions

- Let us interpret y_1 and y_2 as outputs in a regression setting.
- We can introduce an additional 3rd point.



- Now $P([y_1, y_2, y_3] \,|\, \mathbf{K}_3) = \mathcal{N}([y_1, y_2, y_3] \,|\, \mathbf{0}, \mathbf{K}_3)$, where $\mathbf{K}_3$ is now a 3 x 3 covariance matrix!

Constructing Covariance Matrices

- Analogously we can look at the joint probability for arbitrary many points and obtain predictions.
- Issue: how to construct a good covariance matrix?
- A simple heuristics

$$\mathbf{K}_2 = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$$

$$\mathbf{K}_3 = \begin{bmatrix} 1 & 0.6 & 0 \\ 0.6 & 1 & 0.6 \\ 0 & 0.6 & 1 \end{bmatrix}$$

- Note:
  - The ordering of the points y_1, y_2, y_3 matters.
  - Important to ensure that covariance matrices remain positive definite (matrix inversion).

Constructing Covariance Matrices

- Analogously we can look at the joint probability for arbitrary many points and obtain predictions.
- Issue: how to construct a good covariance matrix?
- A simple heuristics

$$\mathbf{K}_2 = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$$

$$\mathbf{K}_3 = \begin{bmatrix} 1 & 0.6 & 0 \\ 0.6 & 1 & 0.6 \\ 0 & 0.6 & 1 \end{bmatrix}$$

- Note:
  - The ordering of the points y_1, y_2, y_3 matters.
  - Important to ensure that covariance matrices remain positive definite (matrix inversion).

Constructing Covariance Matrices

- Analogously we can look at the joint probability for arbitrary many points and obtain predictions.
- Issue: how to construct a good covariance matrix?
- A simple heuristics

$$\mathbf{K}_2 = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$$

$$\mathbf{K}_3 = \begin{bmatrix} 1 & 0.6 & 0 \\ 0.6 & 1 & 0.6 \\ 0 & 0.6 & 1 \end{bmatrix}$$

- Note:
  - The ordering of the points y_1, y_2, y_3 matters.
  - Important to ensure that covariance matrices remain positive definite (matrix inversion).

Constructing Covariance Matrices

A general recipe

- Use a covariance function (kernel function) to construct $\mathbf{K}$:

$$\mathbf{K}_{i,j} = k(x_i, x_j; \boldsymbol{\theta}_\mathsf{K})$$

- For example: The squared exponential covariance function embodies the belief that points further apart are less correlated:

$$k_{\mathsf{SE}}(x_i, x_j, ; A, L) = A^2 \exp\left\{-0.5 \cdot \frac{(x_i - x_j)^2}{L^2}\right\}$$

  - Overall correlation, amplitude
  - Scaling parameter, smoothness

- $\theta_\mathsf{K} = \{A, L\}$ are called hyperparameters.
- We denote the covariance matrix for a set of inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ as: $\mathbf{K}_{\mathbf{X},\mathbf{X}}(\theta_\mathsf{K})$

Constructing Covariance Matrices

A general recipe

- Use a covariance function (kernel function) to construct $\mathbf{K}$:

$$\mathbf{K}_{i,j} = k(x_i, x_j; \boldsymbol{\theta}_\mathsf{K})$$

- For example: The squared exponential covariance function embodies the belief that points further apart are less correlated:

$$k_{\mathsf{SE}}(x_i, x_j, ; A, L) = A^2 \exp\left\{-0.5 \cdot \frac{(x_i - x_j)^2}{L^2}\right\}$$

  - Overall correlation, amplitude
  - Scaling parameter, smoothness

- $\boldsymbol{\theta}_\mathsf{K} = \{A, L\}$ are called hyperparameters.
- We denote the covariance matrix for a set of inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ as: $\mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_\mathsf{K})$

Constructing Covariance Matrices

A general recipe

- Use a covariance function (kernel function) to construct $\mathbf{K}$:

$$\mathbf{K}_{i,j} = k(x_i, x_j; \boldsymbol{\theta}_{\mathsf{K}})$$

- For example: The squared exponential covariance function embodies the belief that points further apart are less correlated:

$$k_{\mathsf{SE}}(x_i, x_j, ; A, L) = A^2 \exp\left\{-0.5 \cdot \frac{(x_i - x_j)^2}{L^2}\right\}$$

  - Overall correlation, amplitude
  - Scaling parameter, smoothness

- $\boldsymbol{\theta}_{\mathsf{K}} = \{A, L\}$ are called hyperparameters.
- We denote the covariance matrix for a set of inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ as: $\mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_{\mathsf{K}})$

Constructing Covariance Matrices

A general recipe

- Use a covariance function (kernel function) to construct $\mathbf{K}$:

$$\mathbf{K}_{i,j} = k(x_i, x_j; \boldsymbol{\theta}_{\mathsf{K}})$$

- For example: The squared exponential covariance function embodies the belief that points further apart are less correlated:

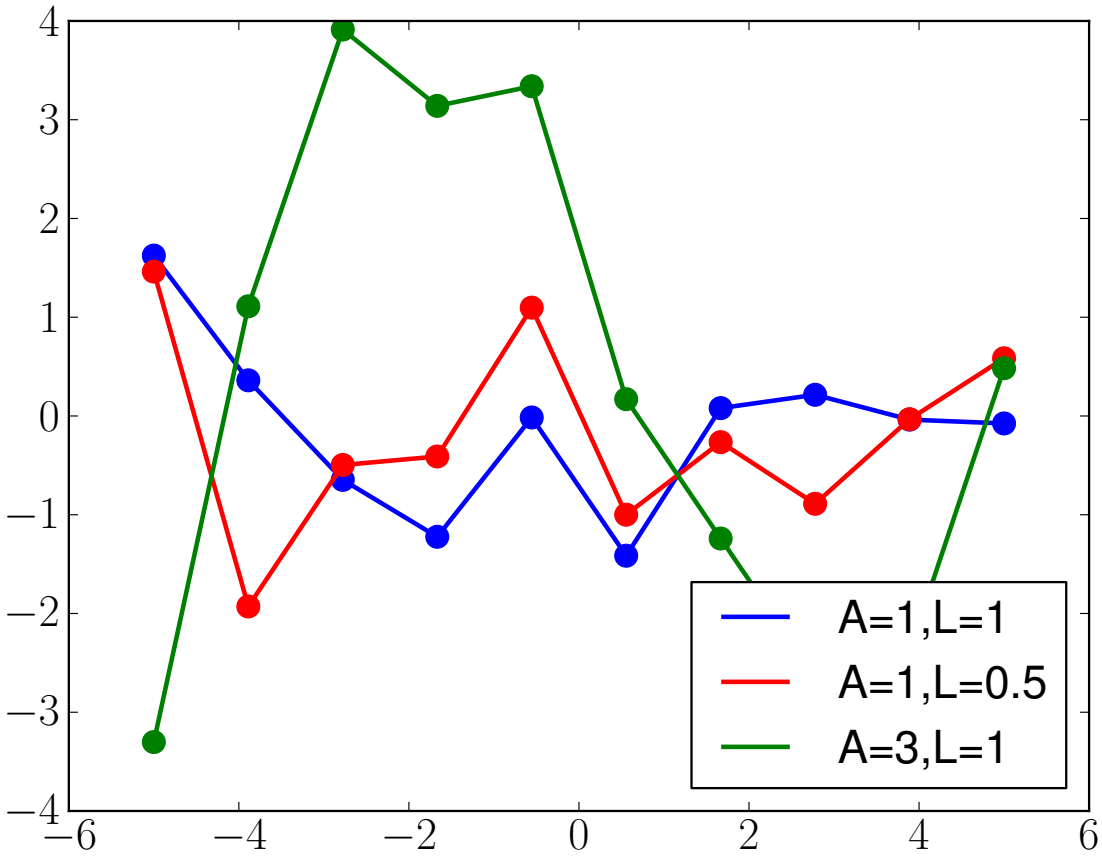
$$k_{\mathsf{SE}}(x_i, x_j, ; A, L) = A^2 \exp\left\{-0.5 \cdot \frac{(x_i - x_j)^2}{L^2}\right\}$$

  - Overall correlation, amplitude
  - Scaling parameter, smoothness

- $\boldsymbol{\theta}_{\mathsf{K}} = \{A, L\}$ are called hyperparameters.
- We denote the covariance matrix for a set of inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ as: $\mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_{\mathsf{K}})$

Constructing Covariance Matrices

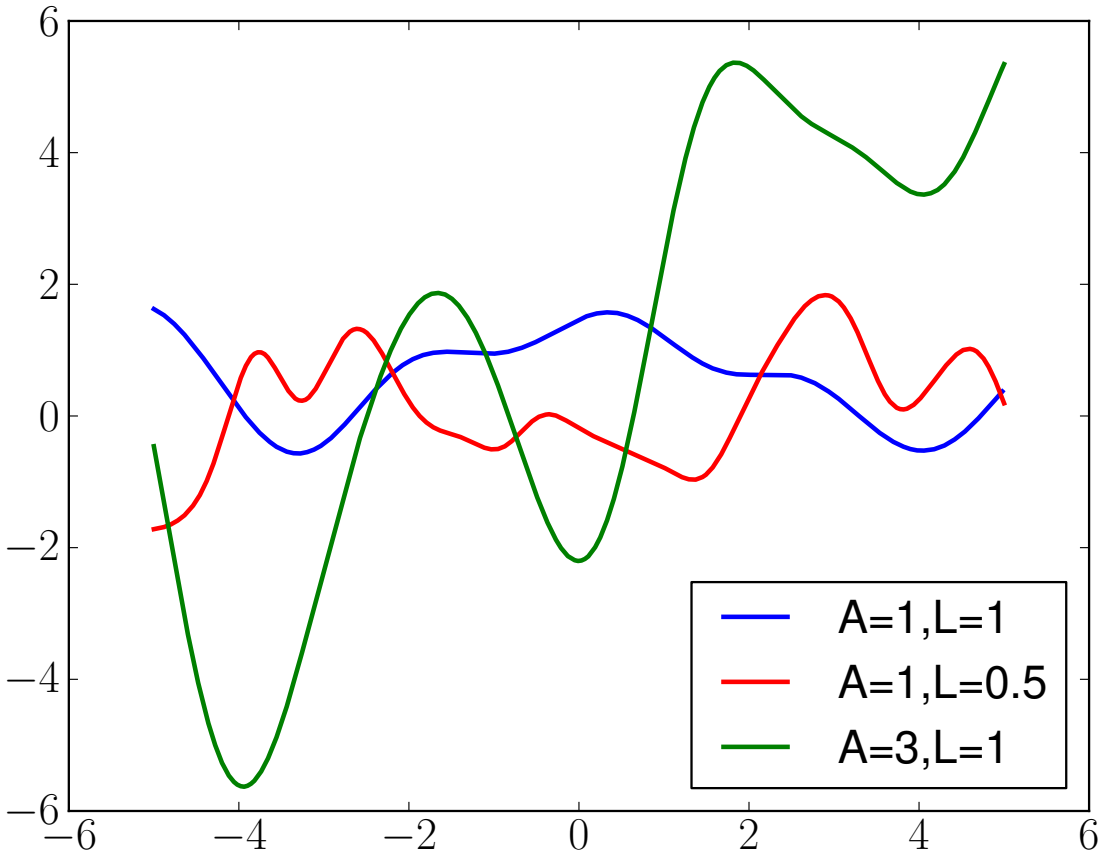
GP samples using the squared exponential covariance function



10D Gaussian

Constructing Covariance Matrices

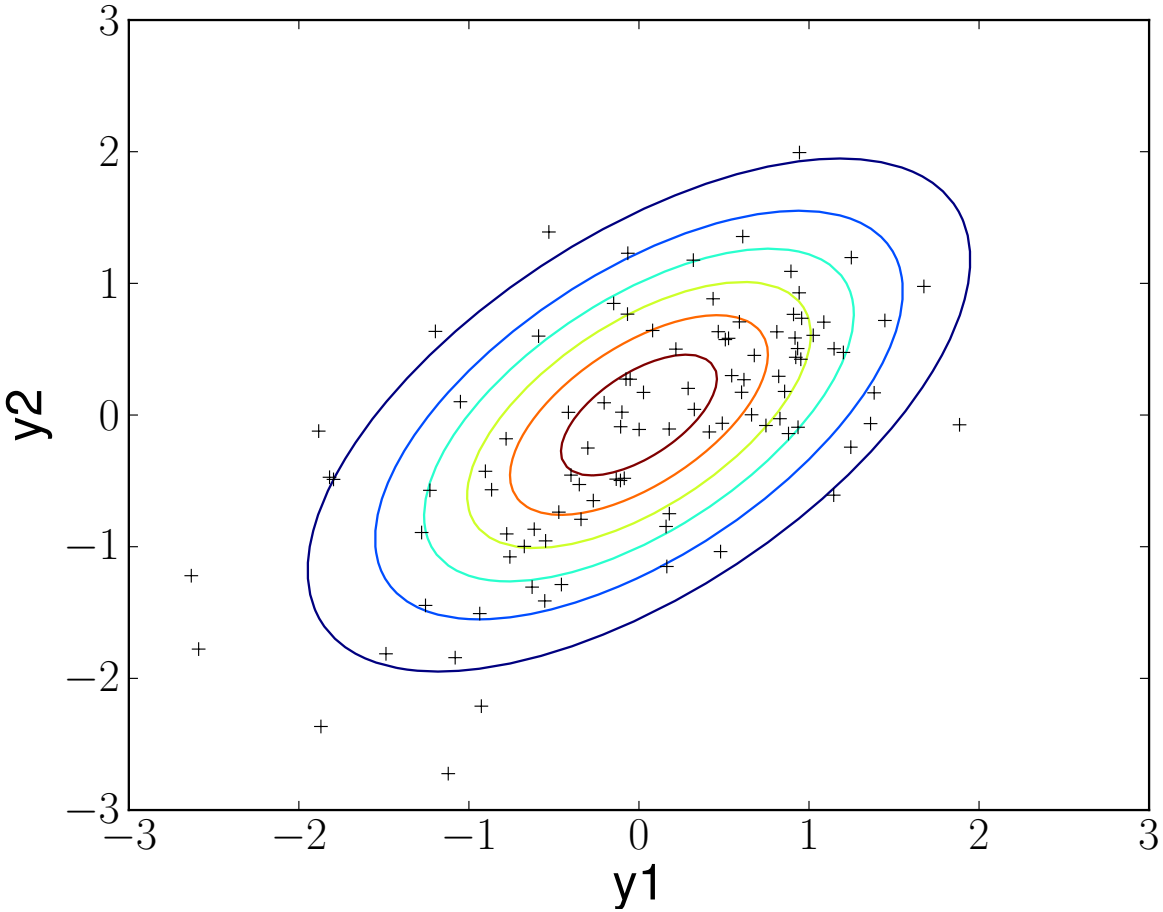
GP samples using the squared exponential covariance function



500D Gaussian

Constructing Covariance Matrices

GP samples using the squared exponential covariance function



Reminder: Every function line corresponds to a sample drawn from this 2D Gaussian!

Why this all works

- Consistency of the 10D and 500D Gaussian.
- A small quizz:
  - Let y_1, y_2, y_3 have covariance matrix

$$\mathbf{K}_3 = \begin{bmatrix} 1 & 5 & 0 \\ 5 & 1 & 5 \\ 0 & 5 & 1 \end{bmatrix} \text{ and inverse } \mathbf{K}_3^{-1} = \begin{bmatrix} 1.5 & -1 & 5 \\ -1 & 2 & -1 \\ 5 & -1 & 1.5 \end{bmatrix}$$

  - Now focus on the variables y_1, y_2, integrating out y_3. Which of the following statements is true

$$\text{a) } \mathbf{K}_2 = \begin{bmatrix} 1 & 5 \\ 5 & 1 \end{bmatrix} \qquad \text{b) } \mathbf{K}_2^{-1} = \begin{bmatrix} 1.5 & -1 \\ -1 & 2 \end{bmatrix}$$

Why this all works

- Consistency of the 10D and 500D Gaussian.
- A small quizz:
  - Let y_1, y_2, y_3 have covariance matrix

$$\mathbf{K}_3 = \begin{bmatrix} 1 & 5 & 0 \\ 5 & 1 & 5 \\ 0 & 5 & 1 \end{bmatrix} \text{ and inverse } \mathbf{K}_3^{-1} = \begin{bmatrix} 1.5 & -1 & 5 \\ -1 & 2 & -1 \\ 5 & -1 & 1.5 \end{bmatrix}$$

  - Now focus on the variables y_1, y_2, integrating out y_3. Which of the following statements is true

$$\text{a) } \mathbf{K}_2 = \begin{bmatrix} 1 & 5 \\ 5 & 1 \end{bmatrix} \qquad \text{b) } \mathbf{K}_2^{-1} = \begin{bmatrix} 1.5 & -1 \\ -1 & 2 \end{bmatrix}$$

Outline

Function space view

So far

1. Joint Gaussian distribution over the set of all outputs $\mathbf{y}$.
2. Covariance function as a recipe to construct a suitable covariance matrices from the corresponding inputs $\mathbf{X}$.

Function space view

The Gaussian process as a prior on functions

- Covariance function and hyperparameters reflect the prior belief on function smoothness, lengthscales etc.
- The general recipe allows a joint Gaussian to be constructed for an arbitrary selection of input locations $\mathbf{X}$.

Prior on infinite function $f(x)$

$$p(f(x)) = \mathsf{GP}(f(x) \,|\, k)$$

Prior on function values $\mathbf{f} = (f_1, \ldots, f_N)$

$$p(\mathbf{f} \,|\, \mathbf{X}, \boldsymbol{\theta}_\mathsf{K}) = \mathcal{N}\left(\mathbf{f} \,|\, \mathbf{0}, \mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_\mathsf{K})\right)$$

Noise-free observations

- Given noise-free training data $\mathcal{D} = \{\mathbf{x}_n, f_n\}_{n=1}^N$
- Want to make predictions $\mathbf{f}^\star$ at test points $\mathbf{X}^\star$
- Joint distribution of $\mathbf{f}$ and $\mathbf{f}^\star$ is

$$p([\mathbf{f}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \boldsymbol{\theta}_\mathsf{K}) = \mathcal{N}\left([\mathbf{f}, \mathbf{f}^\star] \,\middle|\, \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} & \mathbf{K}_{\mathbf{X},\mathbf{X}^\star} \\ \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} \end{bmatrix}\right)$$

(All kernel matrices $\mathbf{K}$ depend on hyperparameters $\boldsymbol{\theta}_\mathsf{K}$ which are dropped for brevity.)

- Real data is rarely noise-free.

Noise-free observations

- Given noise-free training data $\mathcal{D} = \{\mathbf{x}_n, f_n\}_{n=1}^N$
- Want to make predictions $\mathbf{f}^\star$ at test points $\mathbf{X}^\star$
- Joint distribution of $\mathbf{f}$ and $\mathbf{f}^\star$ is

$$p([\mathbf{f}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \boldsymbol{\theta}_\mathsf{K}) = \mathcal{N}\left([\mathbf{f}, \mathbf{f}^\star] \,\middle|\, \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} & \mathbf{K}_{\mathbf{X},\mathbf{X}^\star} \\ \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} \end{bmatrix}\right)$$

  (All kernel matrices $\mathbf{K}$ depend on hyperparameters $\boldsymbol{\theta}_\mathsf{K}$ which are dropped for brevity.)
- Real data is rarely noise-free.

Inference

- Given observed noisy data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, the joint probability over latent function values $\mathbf{f}$ and $\mathbf{f}^\star$ given $\mathbf{y}$ is

$$p([\mathbf{f}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \mathbf{y}, \boldsymbol{\theta}_\mathrm{K}, \sigma^2) \propto \overbrace{\mathcal{N}\left([\mathbf{f}, \mathbf{f}^\star] \,|\, \mathbf{0}, \mathbf{K}\right)}^{\text{Prior}} \times \underbrace{\prod_{n=1}^{N} \mathcal{N}\left(y_n \,|\, f_n, \sigma^2\right)}_{\text{Likelihood}},$$

Inference

- Given observed noisy data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, the joint probability over latent function values $\mathbf{f}$ and $\mathbf{f}^\star$ given $\mathbf{y}$ is

$$p([\mathbf{f}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \mathbf{y}, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) \propto \overbrace{\mathcal{N}\left([\mathbf{f}, \mathbf{f}^\star] \,\middle|\, \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} & \mathbf{K}_{\mathbf{X},\mathbf{X}^\star} \\ \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} \end{bmatrix}\right)}^{\text{Prior}} \times \underbrace{\prod_{n=1}^{N} \mathcal{N}\left(y_n \,|\, f_n, \sigma^2\right)}_{\text{Likelihood}},$$

Inference

- Applying "Gaussian calculus", integrating out $\mathbf{f}$ yields

$$p([\mathbf{y}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \mathbf{y}, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) \propto \mathcal{N}\left([\mathbf{y}, \mathbf{f}^\star] \,\middle|\, \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I} & \mathbf{K}_{\mathbf{X},\mathbf{X}^\star} \\ \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} \end{bmatrix}\right)$$

- Note: Assuming noisy instead of perfect observation noise merely corresponds to adding a diagonal component to the self-covariance $\mathbf{K}_{\mathbf{X},\mathbf{X}}$.

Inference

- Applying "Gaussian calculus", integrating out $\mathbf{f}$ yields

$$p([\mathbf{y}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \mathbf{y}, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) \propto \mathcal{N}\left([\mathbf{y}, \mathbf{f}^\star] \,\middle|\, \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I} & \mathbf{K}_{\mathbf{X},\mathbf{X}^\star} \\ \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} \end{bmatrix}\right)$$

- Note: Assuming noisy instead of perfect observation noise merely corresponds to adding a diagonal component to the self-covariance $\mathbf{K}_{\mathbf{X},\mathbf{X}}$.

Making predictions

- The predictive distribution follows from the joint distribution by completing the square (conditioning)

$$p([\mathbf{y}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \mathbf{y}, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) \propto \mathcal{N}\left([\mathbf{y}, \mathbf{f}^\star] \,\middle|\, \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2\mathbf{I} & \mathbf{K}_{\mathbf{X},\mathbf{X}^\star} \\ \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} \end{bmatrix}\right)$$

- Gaussian predictive distribution for $\mathbf{f}^\star$

$$p(\mathbf{f}^\star \,|\, \mathbf{X}, \mathbf{y}, \mathbf{X}^\star, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) = \mathcal{N}(\mathbf{f}^\star \,|\, \boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star) \text{ with}$$

$$\boldsymbol{\mu}^\star = \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} \left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2\mathbf{I}\right]^{-1} \mathbf{y}$$

$$\boldsymbol{\Sigma}^\star = \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} - \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} \left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2\mathbf{I}\right]^{-1} \mathbf{K}_{\mathbf{X},\mathbf{X}^\star}$$

Making predictions

- The predictive distribution follows from the joint distribution by completing the square (conditioning)

$$p([\mathbf{y}, \mathbf{f}^\star] \,|\, \mathbf{X}, \mathbf{X}^\star, \mathbf{y}, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) \propto \mathcal{N}\left([\mathbf{y}, \mathbf{f}^\star] \middle| \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2\mathbf{I} & \mathbf{K}_{\mathbf{X},\mathbf{X}^\star} \\ \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} & \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} \end{bmatrix}\right)$$

- Gaussian predictive distribution for $\mathbf{f}^\star$

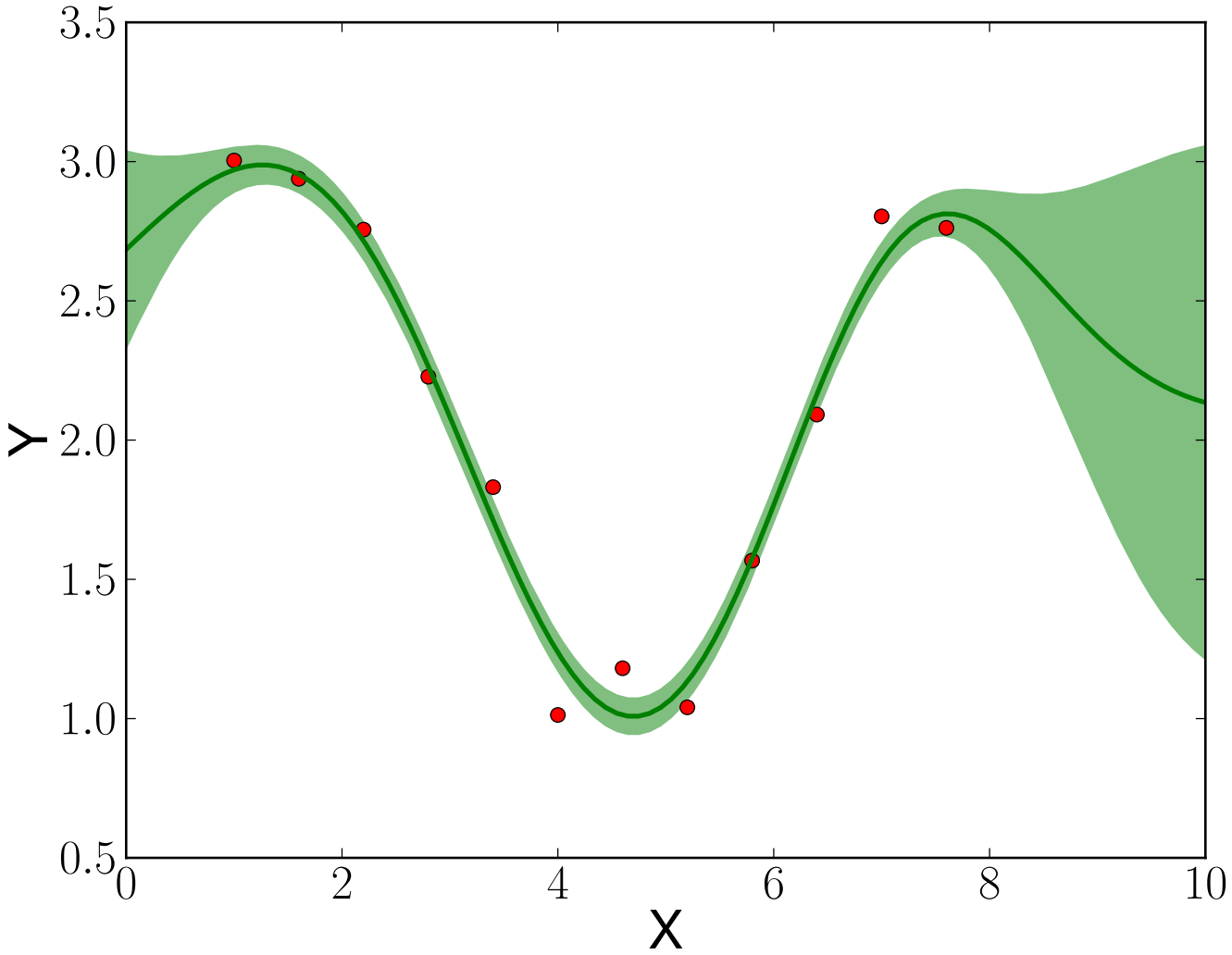
$$p(\mathbf{f}^\star \,|\, \mathbf{X}, \mathbf{y}, \mathbf{X}^\star, \boldsymbol{\theta}_\mathsf{K}, \sigma^2) = \mathcal{N}(\mathbf{f}^\star \,|\, \boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star) \text{ with}$$

$$\boldsymbol{\mu}^\star = \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} \left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2\mathbf{I}\right]^{-1} \mathbf{y}$$

$$\boldsymbol{\Sigma}^\star = \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}^\star} - \mathbf{K}_{\mathbf{X}^\star,\mathbf{X}} \left[\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2\mathbf{I}\right]^{-1} \mathbf{K}_{\mathbf{X},\mathbf{X}^\star}$$

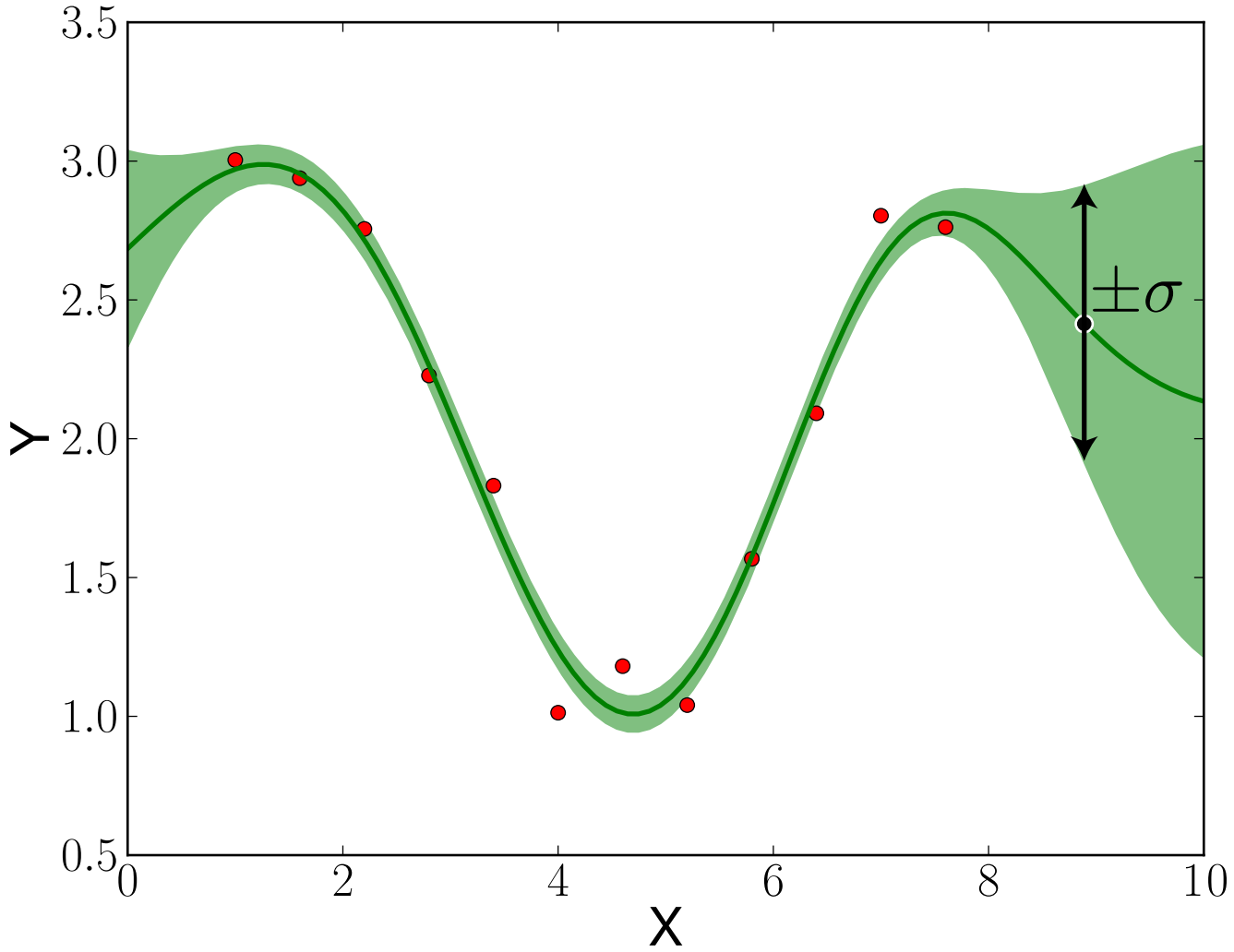
Making predictions

Example



Making predictions

Example



Learning hyperparameters

1. Fixed covariance matrix: $p(\mathbf{y} \mid \mathbf{K})$
2. Constructed covariance matrix: $\{\mathbf{K}\}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta}_\mathsf{K})$
3. Can we learn the hyperparameters $\boldsymbol{\theta}_\mathsf{K}$?

Learning hyperparameters

- Formally we are interested in the posterior

$$p(\boldsymbol{\theta}_K \mid \mathcal{D}) \propto p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}_K\right) p(\boldsymbol{\theta}_K)$$

- Inference is analytically intractable!
- MAP estimate instead of a full posterior. Set θ_K to the most probable hyperparameter settings:

$$\begin{aligned}
\hat{\boldsymbol{\theta}_K} &= \operatorname*{argmax}_{\boldsymbol{\theta}_K} \ln \left[p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}_K\right) p(\boldsymbol{\theta}_K)\right] \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}_K} \ln \mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_K) + \sigma^2 \mathbf{I}\right) + \ln p(\boldsymbol{\theta}_K) \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}_K} \Bigg[-\frac{1}{2} \log \det[\mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_K) + \sigma^2 \mathbf{I}] \\
&\quad - \frac{1}{2} \mathbf{y}^{\mathrm{T}} [\mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_K) + \sigma^2 \mathbf{I}]^{-1} \mathbf{y} - \frac{N}{2} \log 2\pi + \ln p(\boldsymbol{\theta}_K) \Bigg]
\end{aligned}$$

- Optimization can be carried out using standard optimization techniques.

Learning hyperparameters

- Formally we are interested in the posterior

$$p(\boldsymbol{\theta}_{\mathsf{K}} \mid \mathcal{D}) \propto p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}_{\mathsf{K}}\right) p(\boldsymbol{\theta}_{\mathsf{K}})$$

- Inference is analytically intractable!
- MAP estimate instead of a full posterior. Set $\boldsymbol{\theta}_{\mathsf{K}}$ to the most probable hyperparameter settings:

$$\begin{aligned}
\hat{\boldsymbol{\theta}_{\mathsf{K}}} &= \operatorname*{argmax}_{\boldsymbol{\theta}_{\mathsf{K}}} \ln \left[p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}_{\mathsf{K}}\right) p(\boldsymbol{\theta}_{\mathsf{K}})\right] \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}_{\mathsf{K}}} \ln \mathcal{N}\left(\mathbf{y} \,\middle|\, \mathbf{0}, \mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_{\mathsf{K}}) + \sigma^2 \mathbf{I}\right) + \ln p(\boldsymbol{\theta}_{\mathsf{K}}) \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}_{\mathsf{K}}} \Bigg[-\frac{1}{2} \log \det[\mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_{\mathsf{K}}) + \sigma^2 \mathbf{I}] \\
&\quad - \frac{1}{2} \mathbf{y}^{\mathrm{T}} [\mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_{\mathsf{K}}) + \sigma^2 \mathbf{I}]^{-1} \mathbf{y} - \frac{N}{2} \log 2\pi + \ln p(\boldsymbol{\theta}_{\mathsf{K}}) \Bigg]
\end{aligned}$$

- Optimization can be carried out using standard optimization techniques.

Learning hyperparameters

- Formally we are interested in the posterior

$$p(\boldsymbol{\theta}_{\mathsf{K}} \mid \mathcal{D}) \propto p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}_{\mathsf{K}}\right) p(\boldsymbol{\theta}_{\mathsf{K}})$$

- Inference is analytically intractable!
- MAP estimate instead of a full posterior. Set $\boldsymbol{\theta}_{\mathsf{K}}$ to the most probable hyperparameter settings:

$$\begin{aligned}
\hat{\boldsymbol{\theta}_{\mathsf{K}}} &= \operatorname*{argmax}_{\boldsymbol{\theta}_{\mathsf{K}}} \ln \left[p\left(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}_{\mathsf{K}}\right) p(\boldsymbol{\theta}_{\mathsf{K}})\right] \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}_{\mathsf{K}}} \ln \mathcal{N}\left(\mathbf{y} \,\middle|\, \mathbf{0}, \mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_{\mathsf{K}}) + \sigma^2 \mathbf{I}\right) + \ln p(\boldsymbol{\theta}_{\mathsf{K}}) \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}_{\mathsf{K}}} \Bigg[-\frac{1}{2} \log \det[\mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_{\mathsf{K}}) + \sigma^2 \mathbf{I}] \\
&\quad - \frac{1}{2} \mathbf{y}^{\mathrm{T}} [\mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_{\mathsf{K}}) + \sigma^2 \mathbf{I}]^{-1} \mathbf{y} - \frac{N}{2} \log 2\pi + \ln p(\boldsymbol{\theta}_{\mathsf{K}}) \Bigg]
\end{aligned}$$

- Optimization can be carried out using standard optimization techniques.

Choosing covariance functions

- The covariance function embodies the prior belief about functions.
- Example: linear regression

$$y_n = wx_n + c + \psi_n$$

- Covariance function denote covariation

$$\begin{aligned} k(x_n, x'_n) &= \langle y_n y'_n \rangle \\ &= \langle (wx_n + c + \psi_n)(wx'_n + c + \psi'_n) \rangle \\ &= \underbrace{w^2 \cdot x_n x'_n + c^2}_{\text{kernel: } k(x_n, x'_n)} + \delta_{n,n'} \psi_n^2 \end{aligned}$$

Choosing covariance functions

Multidimensional input space

- Generalise squared exponential covariance function to multiple dimensions
  - 1 Dimension $k_{\mathsf{SE}}(x_i, x_j, ; A, L) = A^2 \exp\left\{-0.5 \cdot \frac{(x_i - x_j)^2}{L^2}\right\}$
  - D Dimensions dD

$$k_{\mathsf{SE}}(\mathbf{x}_i, \mathbf{x}_j, ; A, \mathbf{L}) = A^2 \exp\left\{-0.5 \sum_{d=1}^{D} \frac{(x_i^d - x_j^d)^2}{L_d^2}\right\}$$

- Lengthscale parameters L_d denote "relevance" of a particular data dimension.
  - Large L_d correspond to irrelevant dimensions.

Choosing covariance functions

Multidimensional input space

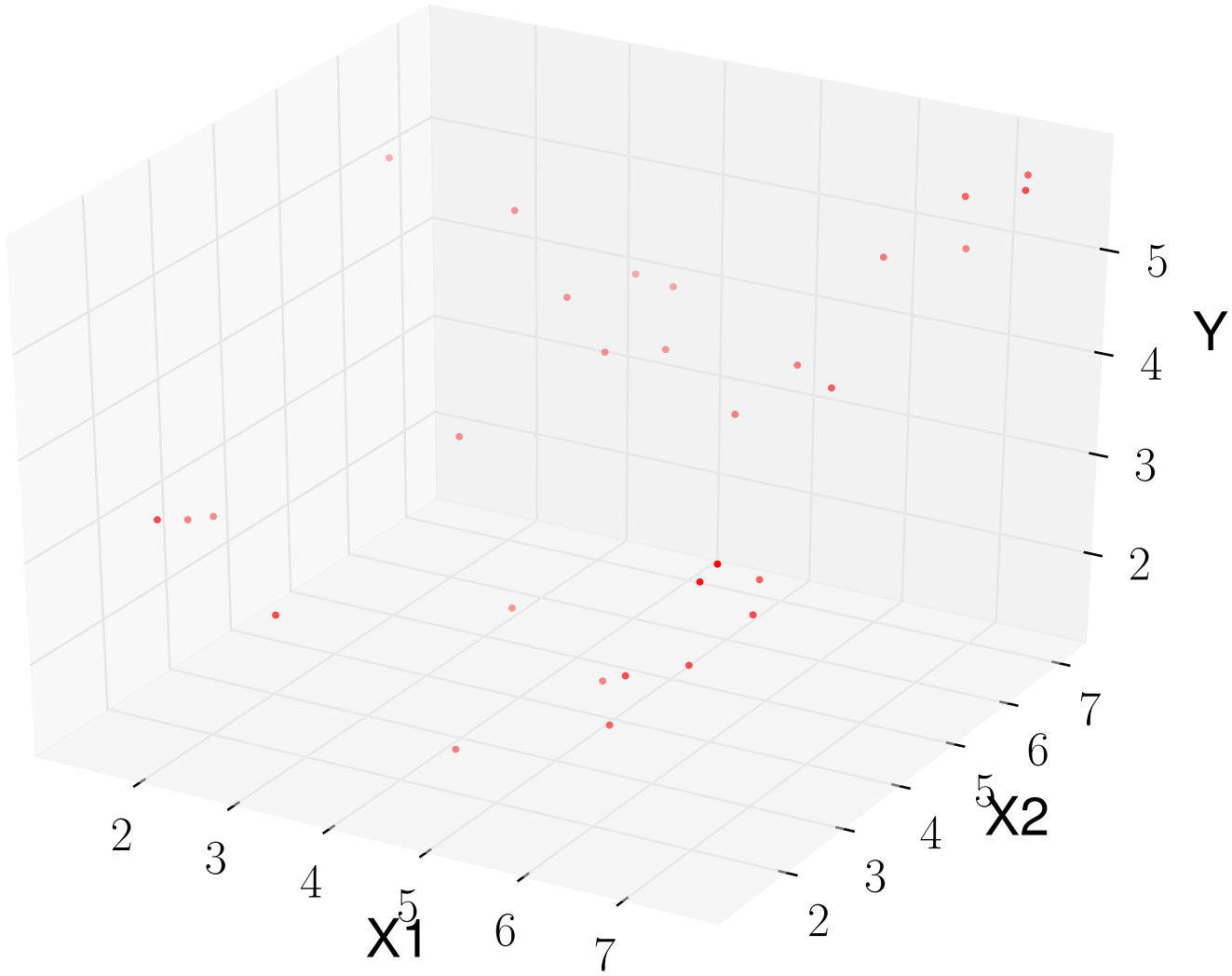
- Generalise squared exponential covariance function to multiple dimensions
  - 1 Dimension $k_{\mathsf{SE}}(x_i, x_j, ; A, L) = A^2 \exp\left\{-0.5 \cdot \frac{(x_i - x_j)^2}{L^2}\right\}$
  - D Dimensions dD

$$k_{\mathsf{SE}}(\mathbf{x}_i, \mathbf{x}_j, ; A, \mathbf{L}) = A^2 \exp\left\{-0.5 \sum_{d=1}^{D} \frac{(x_i^d - x_j^d)^2}{L_d^2}\right\}$$

- Lengthscale parameters L_d denote "relevance" of a particular data dimension.
  - Large L_d correspond to irrelevant dimensions.

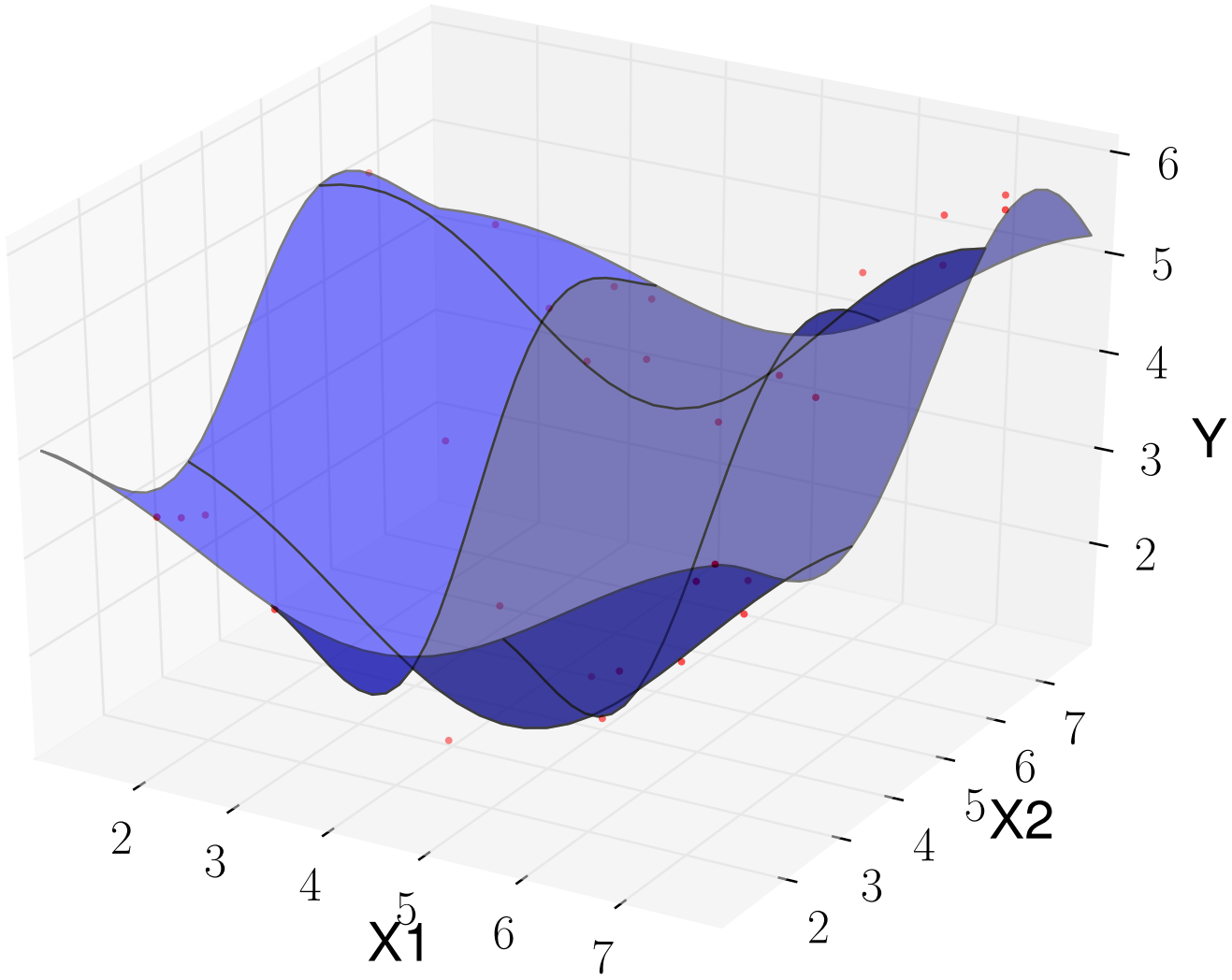
Choosing covariance functions

2D regression



Choosing covariance functions

2D regression



Choosing covariance functions

Any kernel will do

- Established kernels are all valid covariance functions, allowing for a wide range of possible input domains $\mathbf{X}$:
  - Graph kernels (molecules)
  - Kernels defined on strings (DNA sequences)

Choosing covariance functions

Combining existing covariance functions

- The sum of two covariances functions is itself a valid covariance function

$$k_S(x, x') = k_1(x, x') + k_2(x, x')$$

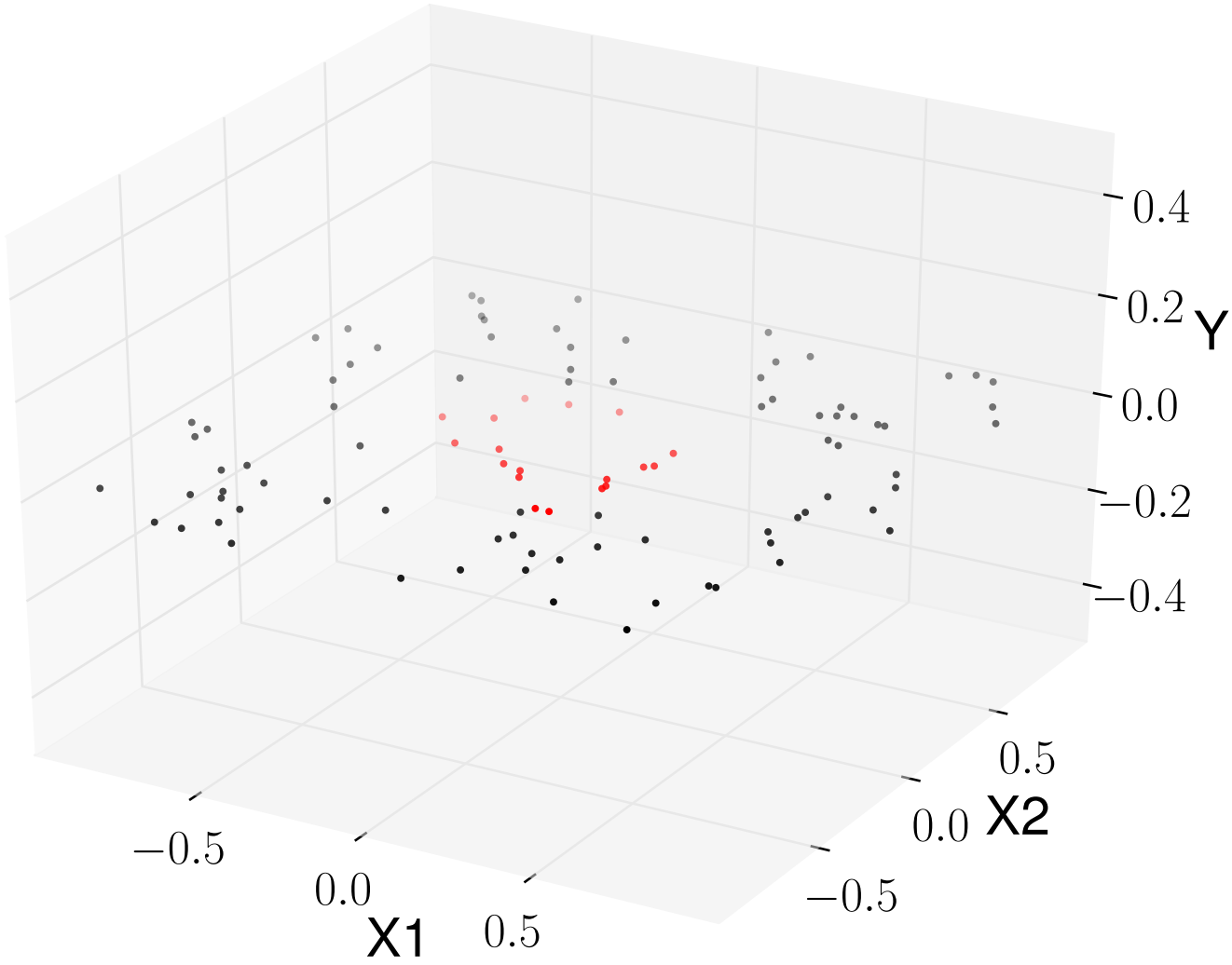
- The product of two covariance functions is itself a valid covariance function

$$k_P(x, x') = k_1(x, x') \cdot k_2(x, x')$$

Outline

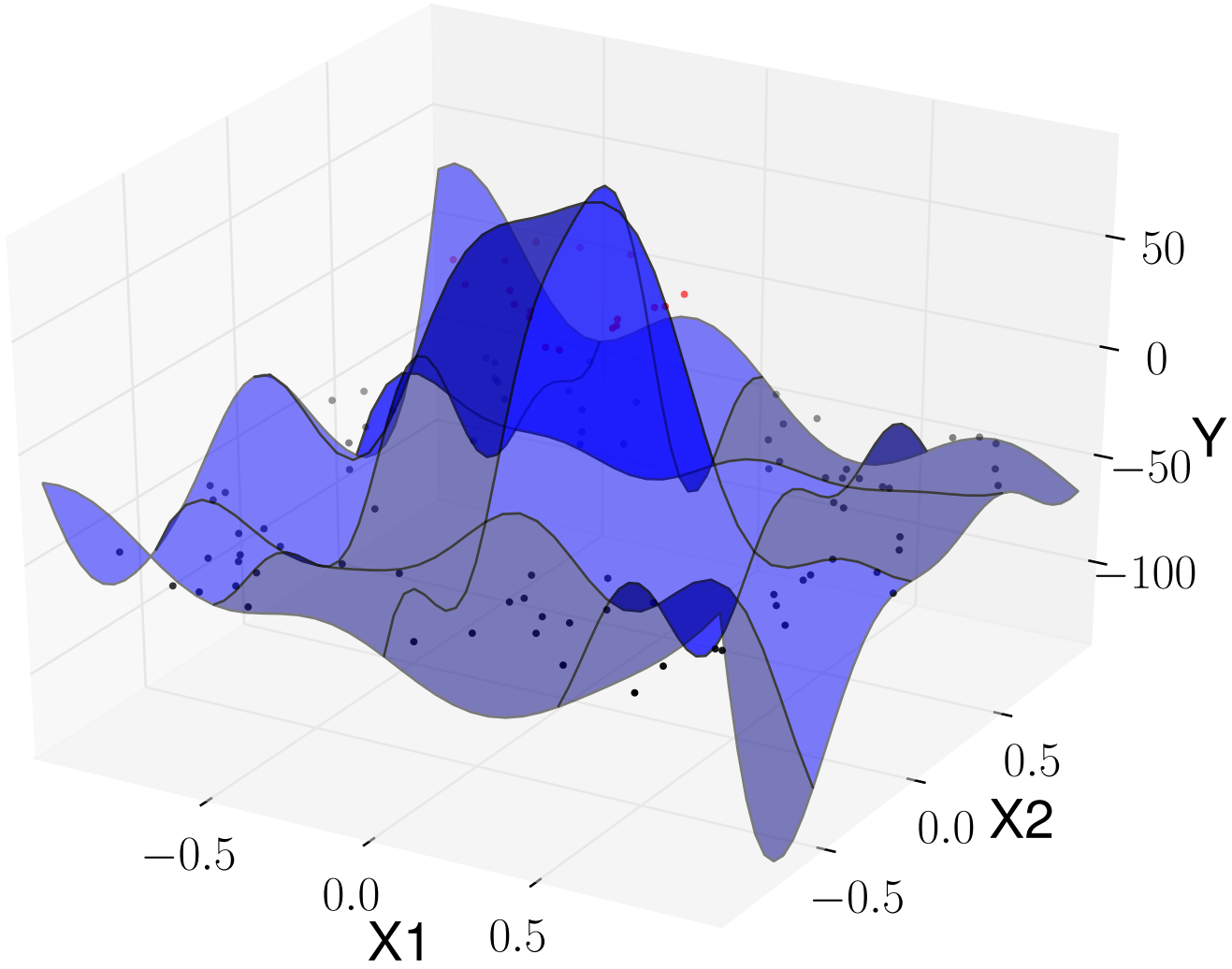
GPs for classification

- How to deal with binary observations?



GPs for classification

- How to deal with binary observations?



GPs for classification

Probit likelihood model

- Posterior with a general likelihood model

$$p(\mathbf{f} \mid \mathbf{X}, \mathbf{y}, \boldsymbol{\theta}_{\mathsf{K}}, \sigma^2) \propto \overbrace{\mathcal{N}\left(\mathbf{f} \mid \mathbf{0}, \mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_{\mathsf{K}})\right)}^{\text{Prior}} \times \overbrace{\prod_{n=1}^{N} p(y_n \mid f_n)}^{\text{Likelihood}}$$

- Classification: probit link model

$$p(y_n = 1 \mid f_n) = \frac{1}{1 + exp(-f_n)}$$

GPs for classification

Inference

- Inference with non-Gaussian likelihood is analytically intractable.
- Idea: approximate the true likelihood terms each with a Gaussian

$$KL\Bigg[\overbrace{\mathcal{N}\left(\mathbf{f} \,|\, \mathbf{0}, \mathbf{K}_{\mathbf{X},\mathbf{K}}(\boldsymbol{\theta}_\mathsf{K})\right)}^{\text{Prior}} \times \overbrace{\prod_{n=1}^{N} p(y_n \,|\, f_n)}^{\text{exact likelihood}} \,\Big|\Big|$$

$$\underbrace{\mathcal{N}\left(\mathbf{f} \,|\, \mathbf{0}, \mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_\mathsf{K})\right)}_{\text{Prior}} \times \underbrace{\prod_{n=1}^{N} \mathcal{N}\left(f_n \,|\, \tilde{\mu}_n, \tilde{\sigma}_n\right)}_{\text{approximation}} \Bigg]$$

- The KL divergence is a common measure of approximation accuracy

$$KL[P||Q] = \int_{\boldsymbol{\theta}} P(\boldsymbol{\theta}) \frac{P(\boldsymbol{\theta} \,|\, \mathcal{D})}{Q(\boldsymbol{\theta})}$$

GPs for classification

Inference

- Inference with non-Gaussian likelihood is analytically intractable.
- Idea: approximate the true likelihood terms each with a Gaussian

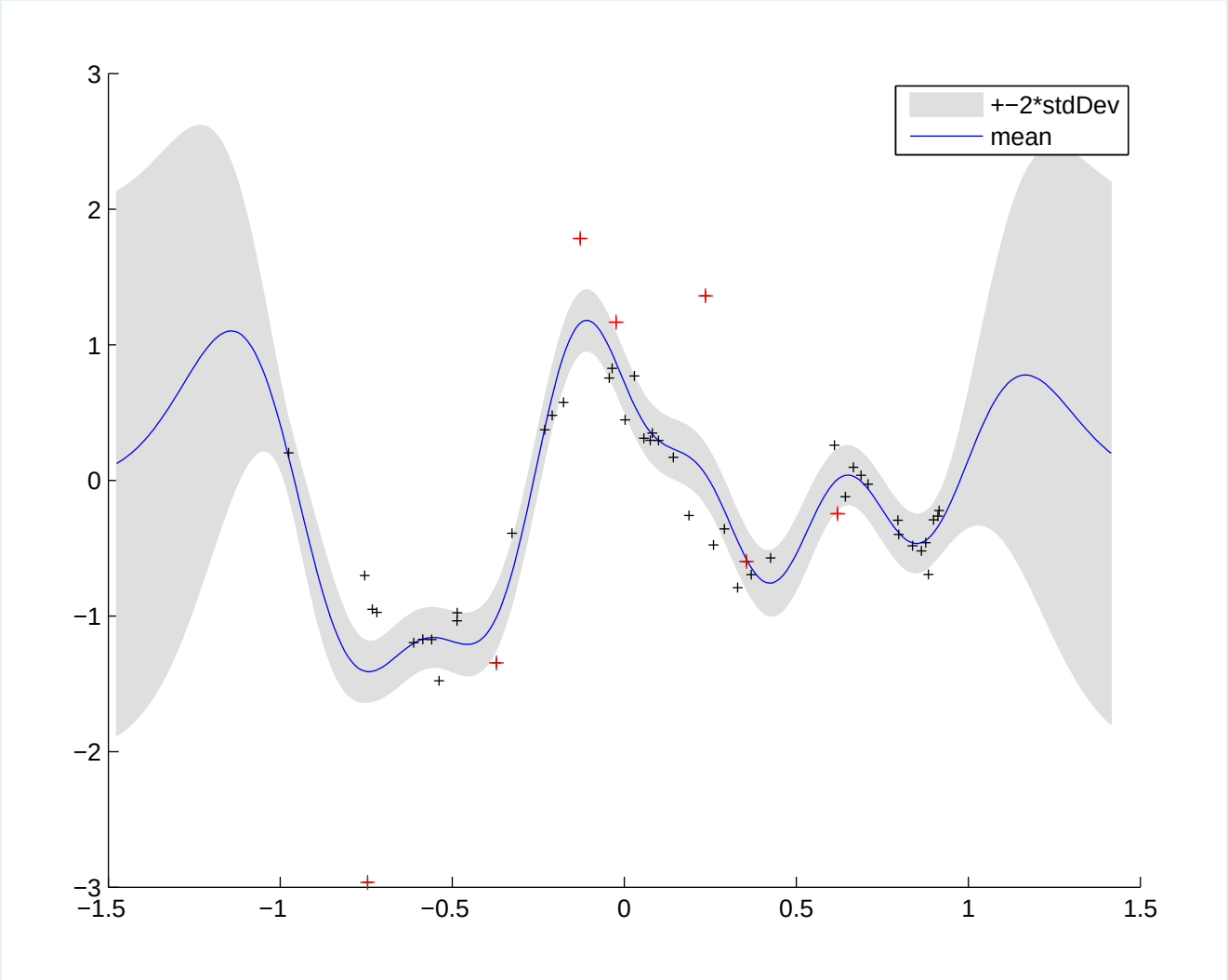
$$KL\Big[\overbrace{\mathcal{N}\left(\mathbf{f}\,|\,\mathbf{0},\mathbf{K}_{\mathbf{X},\mathbf{K}}(\boldsymbol{\theta}_\mathsf{K})\right)}^{\text{Prior}}\times\overbrace{\prod_{n=1}^{N}p(y_n\,|\,f_n)}^{\text{exact likelihood}}\,\Big|\Big|$$
$$\underbrace{\mathcal{N}\left(\mathbf{f}\,|\,\mathbf{0},\mathbf{K}_{\mathbf{X},\mathbf{X}}(\boldsymbol{\theta}_\mathsf{K})\right)}_{\text{Prior}}\times\underbrace{\prod_{n=1}^{N}\mathcal{N}\left(f_n\,|\,\tilde{\mu}_n,\tilde{\sigma}_n\right)}_{\text{approximation}}\Big]$$

- The KL divergence is a common measure of approximation accuracy

$$KL[P||Q]=\int_{\boldsymbol{\theta}}P(\boldsymbol{\theta})\frac{P(\boldsymbol{\theta}\,|\,\mathcal{D})}{Q(\boldsymbol{\theta})}$$

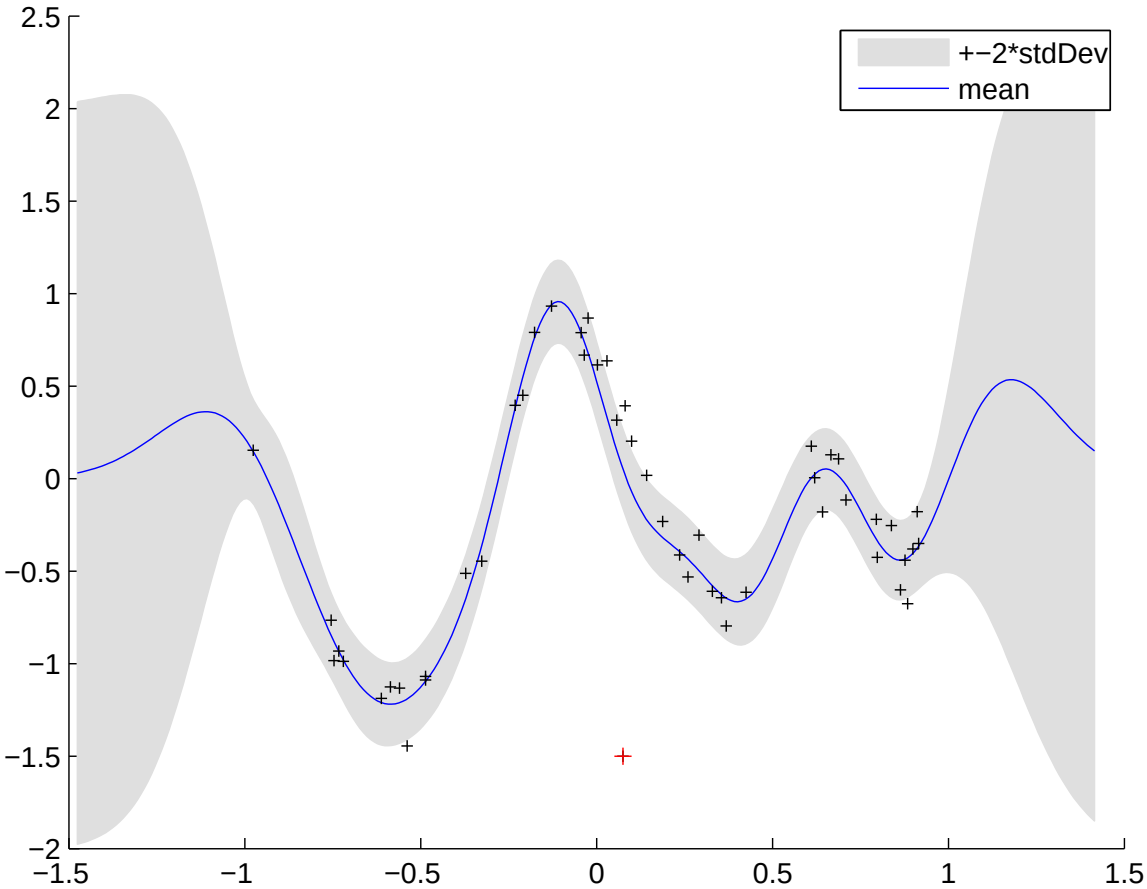
Robust regression

Regression with 15% outliers



Robust regression

Regression with 1% outliers



Robust regression

Mixture likelihood model

- Naive: filtering.
- We rather would like the likelihood model to empobdy the belief that a fraction of datapoints is "useless".

$$p(y_n \mid f_n) = \pi_{ok} \mathcal{N}\left(y_n \mid f_n, \sigma^2\right) + (1 - \pi_{ok}) \mathcal{N}\left(y_n \mid f_n, \sigma^2_\infty\right)$$

Robust regression

Mixture likelihood model

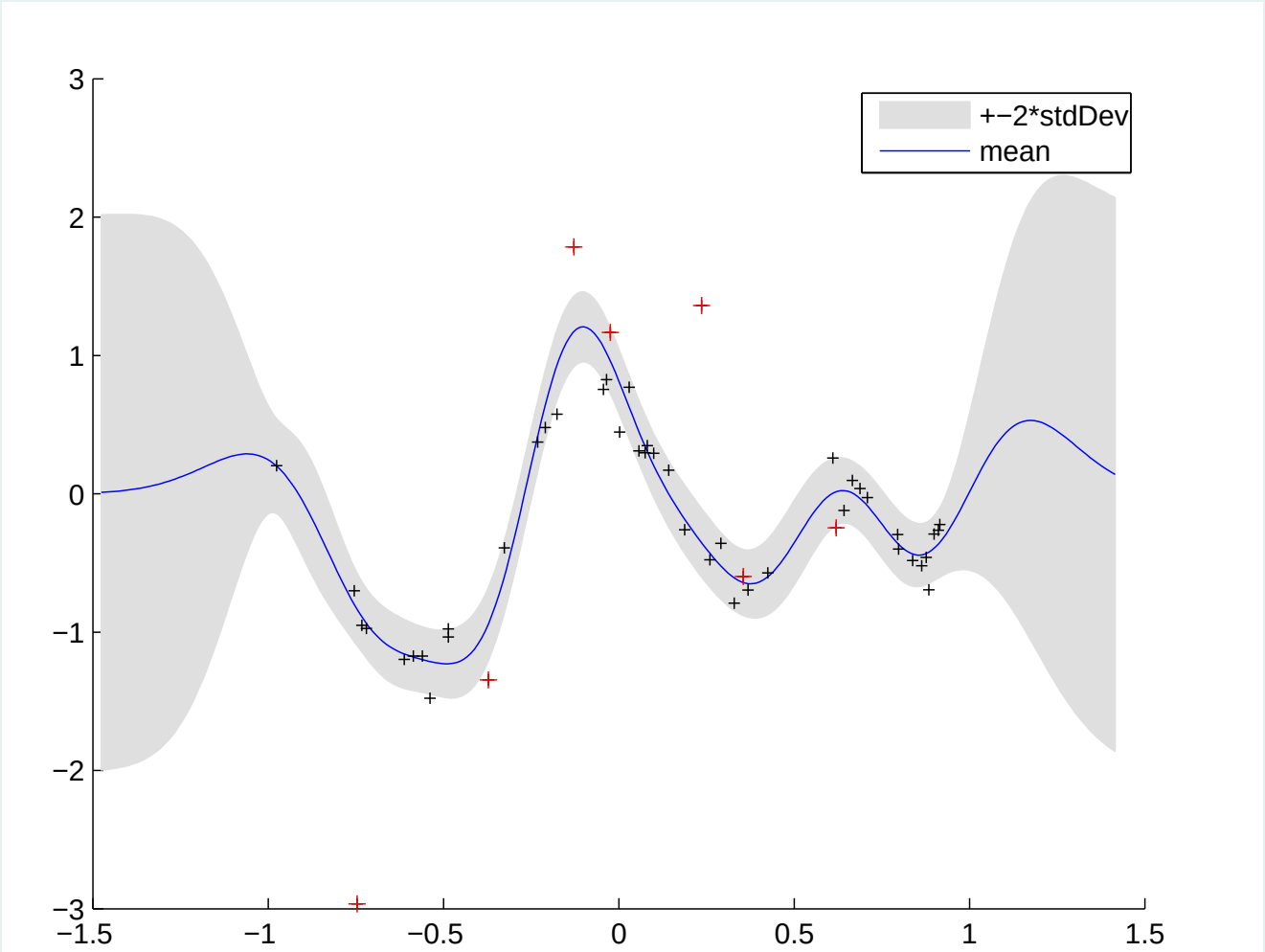
- Naive: filtering.
- We rather would like the likelihood model to empobdy the belief that a fraction of datapoints is "useless".

$$p(y_n \mid f_n) = \pi_{ok} \mathcal{N}\left(y_n \mid f_n, \sigma^2\right) + (1 - \pi_{ok}) \mathcal{N}\left(y_n \mid f_n, \sigma_\infty^2\right)$$

Robust regression

Mixture likelihood in action

Robust noise model



Why Gaussian processes and not something else?

- Tractable probabilistic model; uncertainty estimates
- Equal or better performance than other methods.
- Many other approaches as special case
  - Linear regression
  - Splines
  - Neural networks
- Kernel method; flexible choice of covariance functions.

- Major limitation: inversion of $N \times N$ matrix; scaling $\mathrm{O}(N^3)$.

Outline

Summary

- The key ingredient of a Gaussian processes is the covariance function; a recipe to construct covariance matrices.
- GP predictions boil down to conditioning joint Gaussian distributions.
- Most probable covariance function hyperparameters can be derived from the marginal likelihood.
- Non-Gaussian likelihood models allow for classification and robust regression, however require approximate inference techniques.