Gaussian Processes

Evgeny Burnaev

Skoltech, Moscow, Russia

Outline

1. Motivation
2. Gaussian Process model
3. GP regression
4. Learning Gaussian Process model
5. Gaussian Process classification
6. Take-home messages
7. Afterword: Random Features. Numerical Stability. Non-stationary GP

1 Motivation

2 Gaussian Process model

3 GP regression

4 Learning Gaussian Process model

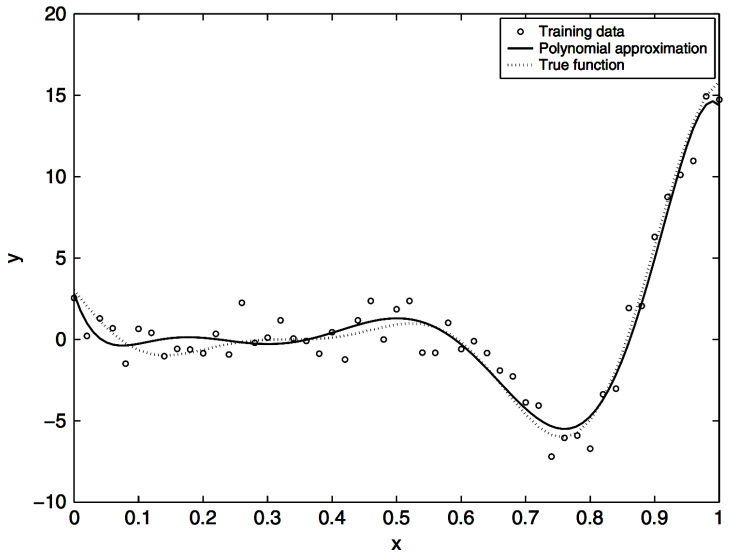
5 Gaussian Process classification

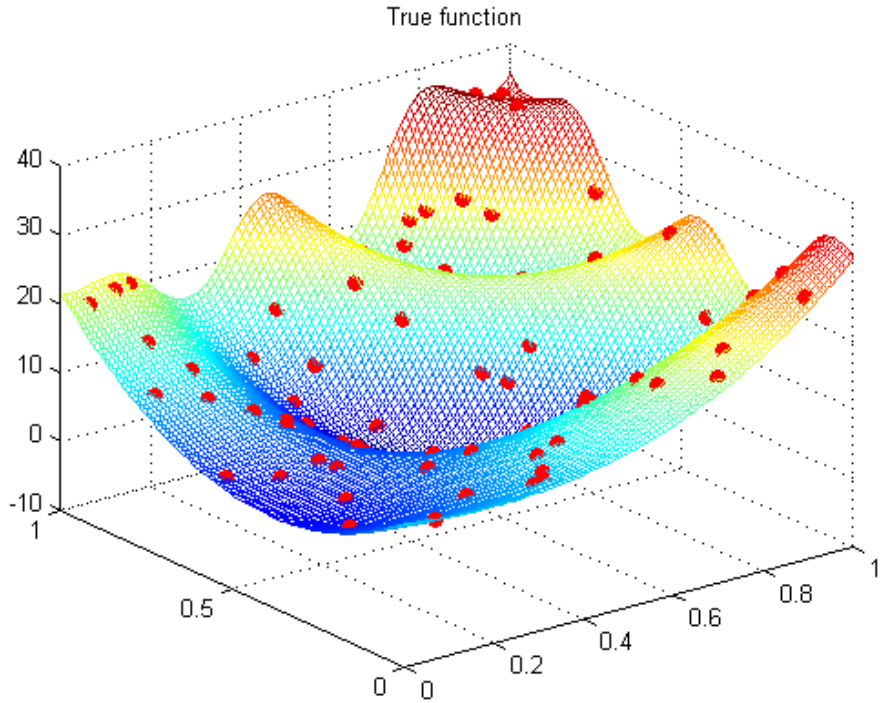
6 Take-home messages

7 Afterword: Random Features. Numerical Stability. Non-stationary GP

The problem

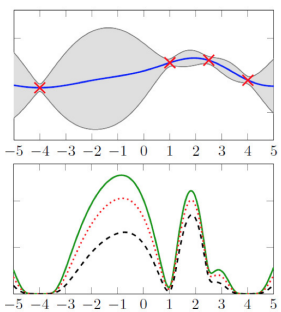
- Learn scalar function $f(\mathbf{x})$ of vector $\mathbf{x}$
- We have (possibly noisy) sample $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$

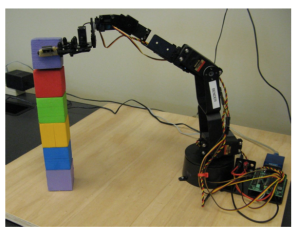


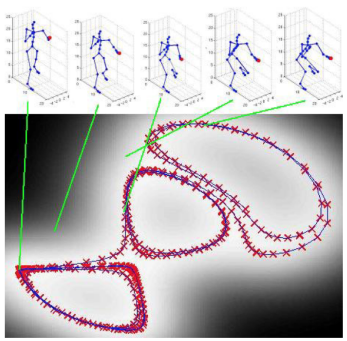


Applications

- **Real-valued regression**:
  - Model-based predictive control: predict yield, quality, losses, etc.
  - Surrogate surfaces for optimization or simulation
  - Robotics: target state $\rightarrow$ required torques
- **Classification**:
  - Recognition: handwritten digits on payment docs, etc.
  - Detection: fraud, screening in chemoinformatics
- **Ordinal Regression**:
  - Disease harmfulness prediction
  - User ratings (movies, shops, restaurants)

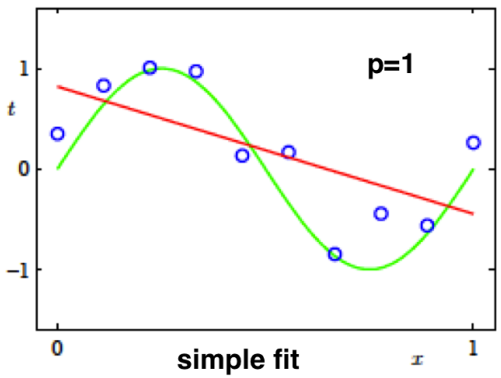


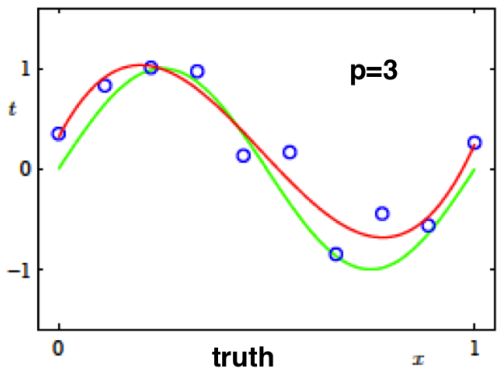


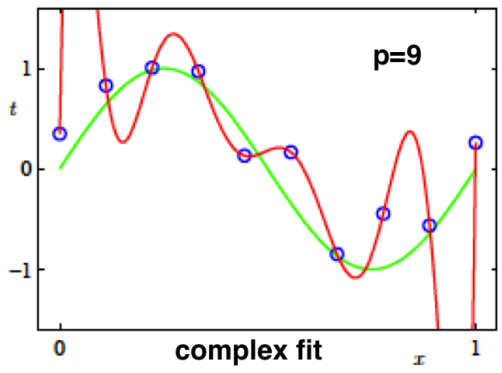


Model Complexity

- Data generation process is often unknown and can be complex:

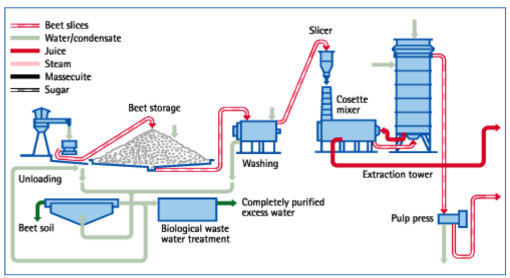




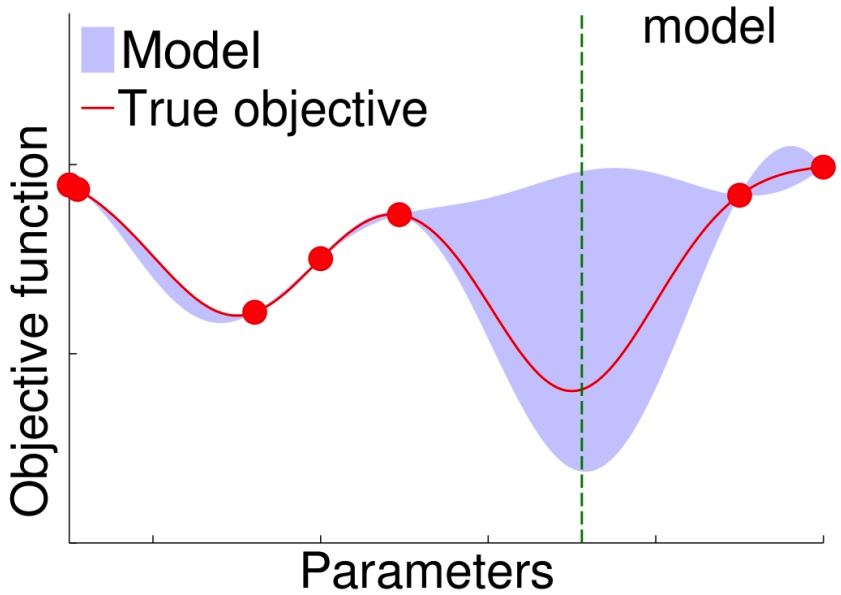


- **Problems**:
  - Fitting complicated models can be hard
  - How to find an appropriate model?
  - How to avoid over-fitting?
  - Uncertainty of predictions?

- **Problem**:
  - **Object**: production of one batch of a product
  - **Input** $\mathbf{x}$: beet chips shape, temperature; wash water temperature, pH and flux, etc.
  - **Output** y: costs, losses, efficiency
- **Challenges**:
  - Heterogeneous data and noise,
  - Large volumes of high-dimensional stream data,
  - Missing values, outliers, etc.
- **Knowing error bars is very important**:
  - Setting $\mathbf{x}_1 \rightarrow$ profit of 40 ± 10 units
  - Setting $\mathbf{x}_2 \rightarrow$ profit of 60 ± 40 units
  - **Which are the best setting, $\mathbf{x}_1$ or $\mathbf{x}_2$?**



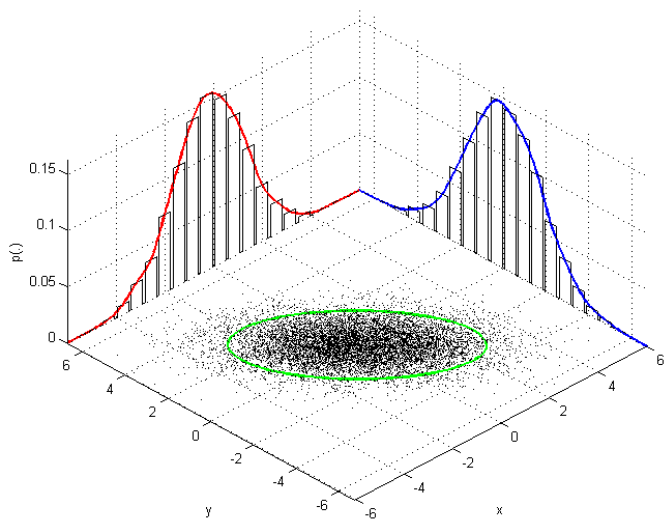
- In high-dimensional case we need many functions evaluations
- Often each evaluation is costly, e.g. in case of experiments



- Error bars are needed to see if a region is still promising

- Parametric family of functions $f(\mathbf{x}; \boldsymbol{\theta})$
- Define a prior over $\boldsymbol{\theta}$
- Perform predictions and estimate uncertainty
- For flexible models we a.s. get intractable integrals over $\boldsymbol{\theta}$

For Gaussian case usually everything is explicit



Solution of complex ML problems with simple Gaussian models?

1 Motivation

2 Gaussian Process model

3 GP regression

4 Learning Gaussian Process model

5 Gaussian Process classification

6 Take-home messages

7 Afterword: Random Features. Numerical Stability. Non-stationary GP

Functional classes

- **Hypothesis set** $\mathcal{F} \subset Y^X$ is a subset of functions out of which the learner selects his/her hypothesis
  - represents a prior knowledge about the task at hand
  - depends on available features
- Typical examples (from statistics)
  - Sobolev-type classes:

$$\mathcal{F}_L^k = \left\{ f : \int \left\| \frac{\partial^k f(\mathbf{x})}{\partial \mathbf{x}^k} \right\|^2 d\mathbf{x} \leq L \right\}$$

  - Lipschitz classes:

$$\mathcal{F}_L = \{ f : |f(\mathbf{x}) - f(\mathbf{x}')| \leq L\rho(\mathbf{x}, \mathbf{x}') \}$$

- How to impose regularity on a function in Bayesian case?

- **Hypothesis set** $\mathcal{F} \subset Y^X$ is a subset of functions out of which the learner selects his/her hypothesis
  - represents a prior knowledge about the task at hand
  - depends on available features
- Typical examples (from statistics)
  - Sobolev-type classes:

$$\mathcal{F}_L^k = \left\{ f : \int \left\| \frac{\partial^k f(\mathbf{x})}{\partial \mathbf{x}^k} \right\|^2 d\mathbf{x} \leq L \right\}$$

  - Lipschitz classes:

$$\mathcal{F}_L = \{ f : |f(\mathbf{x}) - f(\mathbf{x}')| \leq L\rho(\mathbf{x}, \mathbf{x}') \}$$

- How to impose regularity on a function in Bayesian case?

- **Hypothesis set** $\mathcal{F} \subset Y^X$ is a subset of functions out of which the learner selects his/her hypothesis
  - represents a prior knowledge about the task at hand
  - depends on available features
- Typical examples (from statistics)
  - Sobolev-type classes:

$$\mathcal{F}_L^k = \left\{ f : \int \left\| \frac{\partial^k f(\mathbf{x})}{\partial \mathbf{x}^k} \right\|^2 d\mathbf{x} \leq L \right\}$$

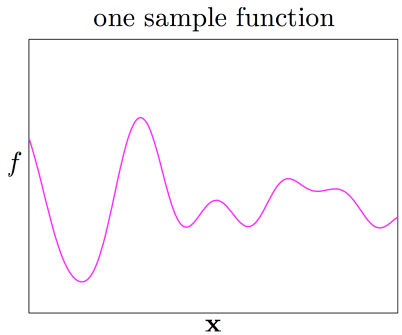
  - Lipschitz classes:

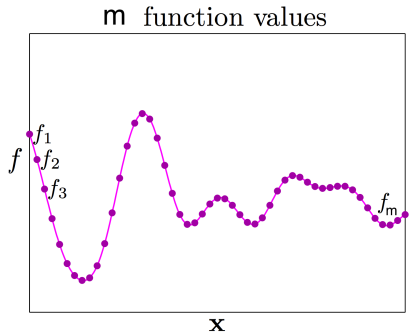
$$\mathcal{F}_L = \{ f : |f(\mathbf{x}) - f(\mathbf{x}')| \leq L\rho(\mathbf{x}, \mathbf{x}') \}$$

- How to impose regularity on a function in Bayesian case?

What is a Gaussian process?

- Continuous stochastic process — random function — a set of random variables $f(\mathbf{x})$ indexed by a continuous variable $\mathbf{x}$





- Let us denote by
  - $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ a set of d-dimensional inputs,
  - $\mathbf{f} = \{f_1, \ldots, f_m\}$ a set of random function values $f_i = f(\mathbf{x}_i)$

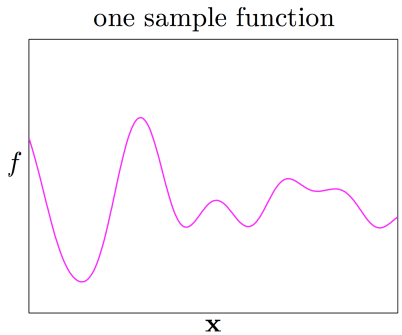
- GP: Any set of function variables $\{f_i\}_{i=1}^m$ has joint Gaussian distribution

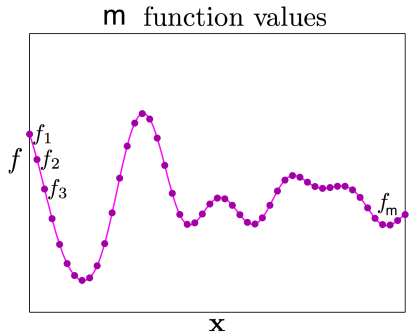
$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

- Conditional model — density of inputs is not modeled

What is a Gaussian process?

- Continuous stochastic process — random function — a set of random variables $f(\mathbf{x})$ indexed by a continuous variable $\mathbf{x}$





- Let us denote by
  - $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ a set of d-dimensional inputs,
  - $\mathbf{f} = \{f_1, \ldots, f_m\}$ a set of random function values $f_i = f(\mathbf{x}_i)$

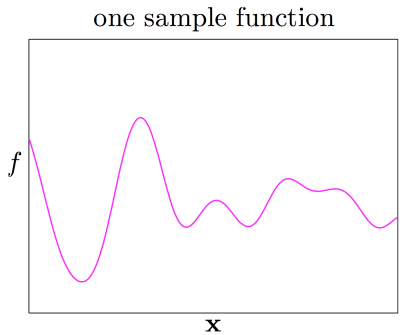
- GP: Any set of function variables $\{f_i\}_{i=1}^m$ has joint Gaussian distribution

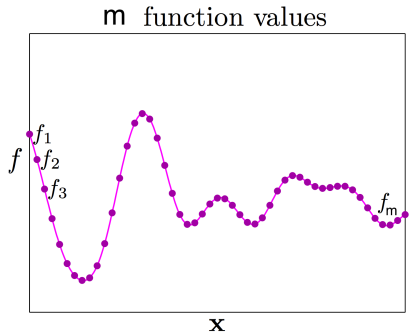
$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

- Conditional model — density of inputs is not modeled

What is a Gaussian process?

- Continuous stochastic process — random function — a set of random variables $f(\mathbf{x})$ indexed by a continuous variable $\mathbf{x}$





- Let us denote by
  - $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ a set of d-dimensional inputs,
  - $\mathbf{f} = \{f_1, \ldots, f_m\}$ a set of random function values $f_i = f(\mathbf{x}_i)$

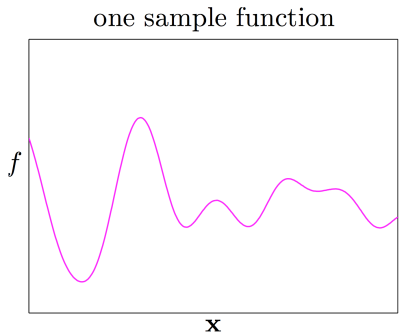
- GP: Any set of function variables $\{f_i\}_{i=1}^m$ has joint Gaussian distribution

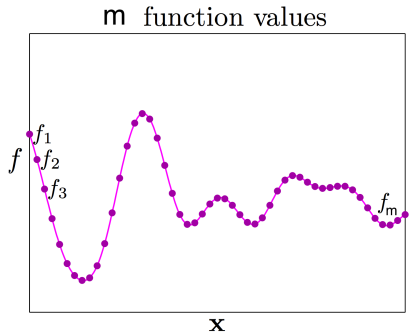
$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

- Conditional model — density of inputs is not modeled

What is a Gaussian process?

- Continuous stochastic process — random function — a set of random variables $f(\mathbf{x})$ indexed by a continuous variable $\mathbf{x}$





- Let us denote by
  - $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ a set of d-dimensional inputs,
  - $\mathbf{f} = \{f_1, \ldots, f_m\}$ a set of random function values $f_i = f(\mathbf{x}_i)$

- **GP**: Any set of function variables $\{f_i\}_{i=1}^m$ has joint Gaussian distribution

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$$

- Conditional model — density of inputs is not modeled

Mean values and Covariances

- Mean value is constructed from a priori given deterministic function

$$\boldsymbol{\mu} = \{\mu_i\} = \{\mu(\mathbf{x}_i)\}$$

- Covariance matrix is constructed from covariance function

$$\mathbf{K} = \{\mathbf{K}_{ij}\} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}$$

- Covariance function characterizes covariance between points in the process

$$K(\mathbf{x}, \mathbf{x}') = \operatorname{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \mathbb{E}\left(f(\mathbf{x}) - \mu(\mathbf{x})\right)\left(f(\mathbf{x}') - \mu(\mathbf{x}')\right)$$

Mean values and Covariances

- Mean value is constructed from a priori given deterministic function

$$\boldsymbol{\mu} = \{\mu_i\} = \{\mu(\mathbf{x}_i)\}$$

- Covariance matrix is constructed from covariance function

$$\mathbf{K} = \{\mathbf{K}_{ij}\} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}$$

- Covariance function characterizes covariance between points in the process

$$K(\mathbf{x}, \mathbf{x}') = \operatorname{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \mathbb{E}\left(f(\mathbf{x}) - \mu(\mathbf{x})\right)\left(f(\mathbf{x}') - \mu(\mathbf{x}')\right)$$

- Mean value is constructed from a priori given deterministic function

$$\boldsymbol{\mu} = \{\mu_i\} = \{\mu(\mathbf{x}_i)\}$$

- Covariance matrix is constructed from covariance function

$$\mathbf{K} = \{\mathbf{K}_{ij}\} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}$$

- Covariance function characterizes covariance between points in the process

$$K(\mathbf{x}, \mathbf{x}') = \operatorname{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \mathbb{E}\left(f(\mathbf{x}) - \mu(\mathbf{x})\right)\left(f(\mathbf{x}') - \mu(\mathbf{x}')\right)$$

- Mean value is constructed from a priori given deterministic function

$$\boldsymbol{\mu} = \{\mu_i\} = \{\mu(\mathbf{x}_i)\}$$

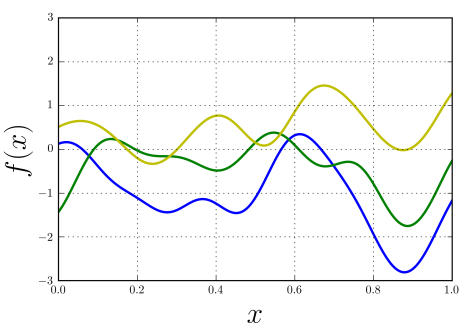
- Covariance matrix is constructed from covariance function

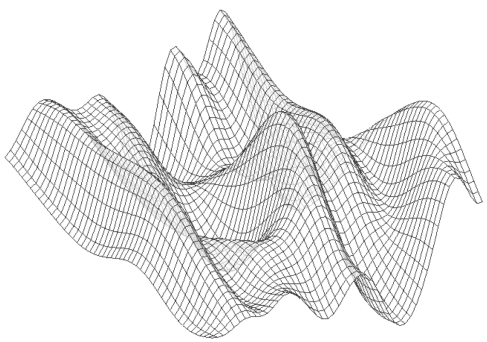
$$\mathbf{K} = \{\mathbf{K}_{ij}\} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}$$

- Covariance function characterizes covariance between points in the process

$$K(\mathbf{x}, \mathbf{x}') = \operatorname{cov}(f(\mathbf{x}), f(\mathbf{x}')) = \mathbb{E}\left(f(\mathbf{x}) - \mu(\mathbf{x})\right)\left(f(\mathbf{x}') - \mu(\mathbf{x}')\right)$$

$$f(\mathbf{x}) \sim \mathcal{GP}(\cdot|\mu(\mathbf{x}), K(\mathbf{x}, \mathbf{x}'))$$





Example: squared-exponential kernel

- The most commonly-used kernel in machine learning

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2r_i^2} \right\}$$

- *A GP need not use the Gaussian kernel. There are a lot of other choices*
- Consider $\mathbf{x} = \mathbf{x}' \rightarrow$ marginal function variance is σ_f^2

Example: squared-exponential kernel

- The most commonly-used kernel in machine learning

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2r_i^2} \right\}$$

- *A GP need not use the Gaussian kernel. There are a lot of other choices*
- Consider $\mathbf{x} = \mathbf{x}' \rightarrow$ marginal function variance is σ_f^2

Example: squared-exponential kernel

- The most commonly-used kernel in machine learning

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2r_i^2} \right\}$$

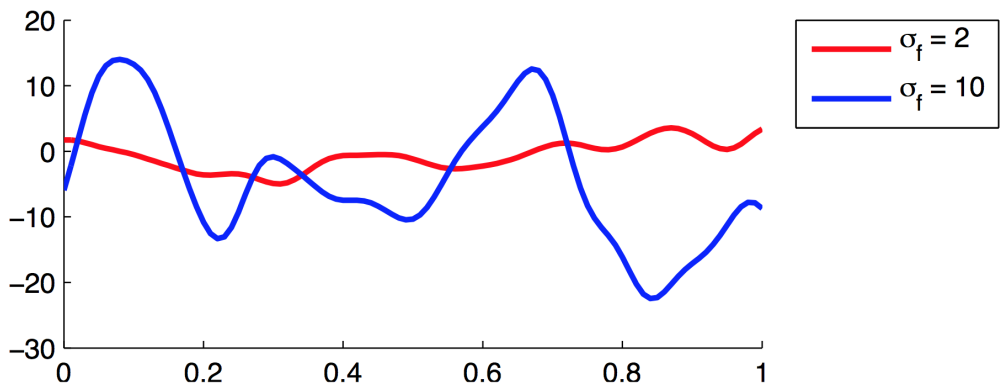
- *A GP need not use the Gaussian kernel. There are a lot of other choices*
- Consider $\mathbf{x} = \mathbf{x}' \rightarrow$ marginal function variance is σ_f^2

Example: squared-exponential kernel

- The most commonly-used kernel in machine learning

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2r_i^2} \right\}$$

- *A GP need not use the Gaussian kernel. There are a lot of other choices*
- Consider $\mathbf{x} = \mathbf{x}' \rightarrow$ marginal function variance is σ_f^2



Example: squared-exponential kernel

- The r_i parameters give the overall lengthscale in dimension i

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2r_i^2} \right\}$$

- Typical distance between peaks $\approx r$

Example: squared-exponential kernel

- The r_i parameters give the overall lengthscale in dimension i

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2r_i^2} \right\}$$

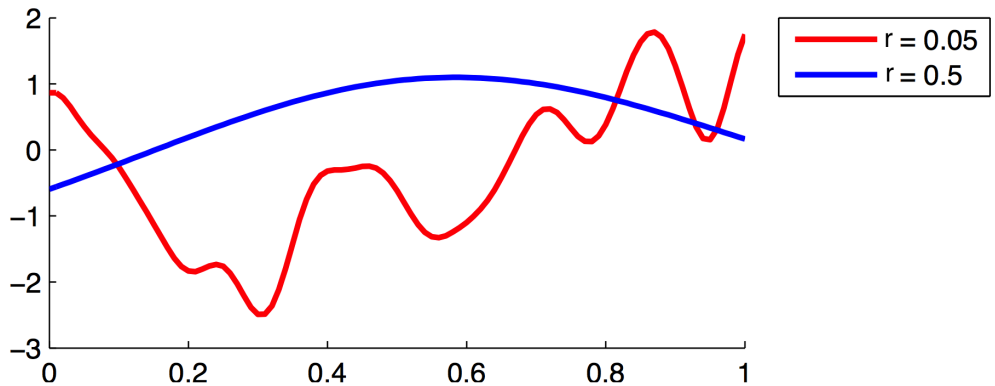
- Typical distance between peaks $\approx r$

Example: squared-exponential kernel

- The r_i parameters give the overall lengthscale in dimension i

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - x_i')^2}{2r_i^2} \right\}$$

- Typical distance between peaks $\approx r$



Example: Gaussian process covariance functions

- More general example of covariance function

$$K(x, x') = \sigma_f^2 \exp\left\{-\left(\frac{|x - x'|}{r}\right)^{\alpha}\right\} + \sigma_1^2 + \sigma_2^2 \delta(x - x')$$

- As usual covariance function parameters are interpretable
  - σ_f^2 — marginal function variance
  - σ_1^2 — variance of bias
  - σ_2^2 — noise variance
  - r — lengthscale
  - $\alpha \geq 1$ — roughness
- Once the mean and covariance functions are defined, everything else about GPs follows from the basic rules of probability applied to multivariate Gaussians

Example: Gaussian process covariance functions

- More general example of covariance function

$$K(x, x') = \sigma_f^2 \exp\left\{-\left(\frac{|x - x'|}{r}\right)^{\alpha}\right\} + \sigma_1^2 + \sigma_2^2 \delta(x - x')$$

- As usual covariance function parameters are interpretable
  - σ_f^2 — marginal function variance
  - σ_1^2 — variance of bias
  - σ_2^2 — noise variance
  - r — lengthscale
  - $\alpha \geq 1$ — roughness
- Once the mean and covariance functions are defined, everything else about GPs follows from the basic rules of probability applied to multivariate Gaussians

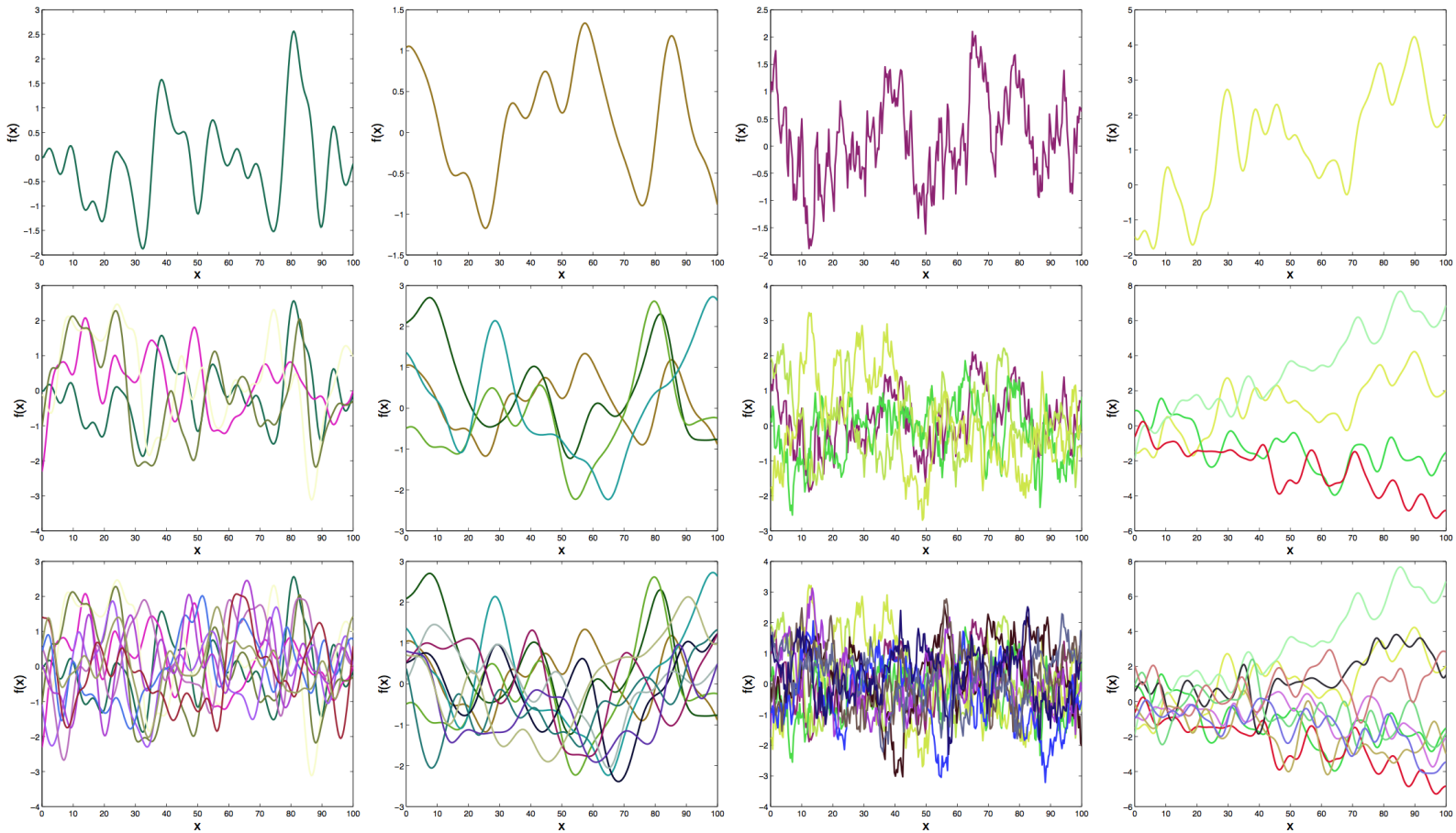
Example: Gaussian process covariance functions

- More general example of covariance function

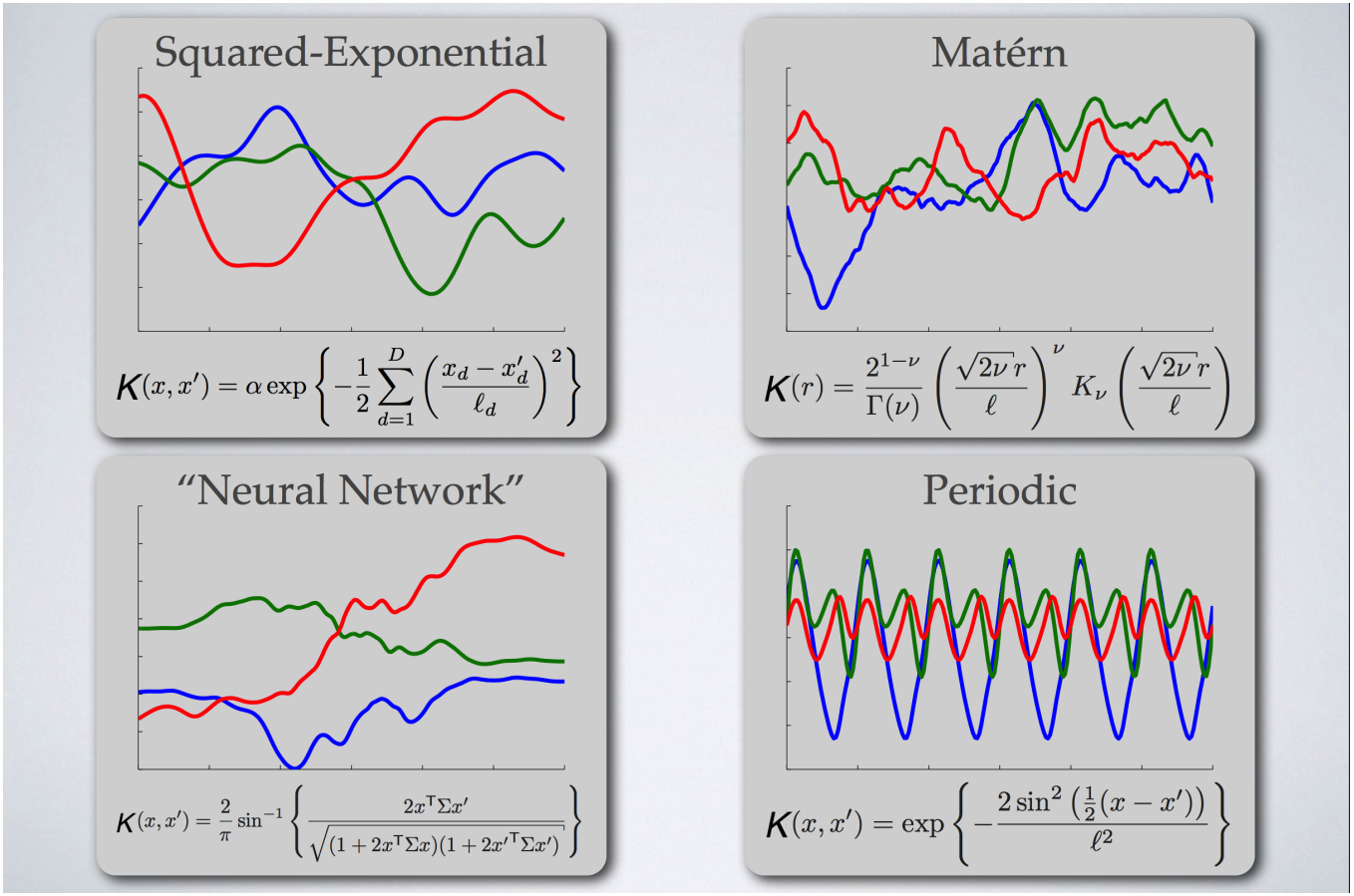
$$K(x, x') = \sigma_f^2 \exp\left\{-\left(\frac{|x - x'|}{r}\right)^{\alpha}\right\} + \sigma_1^2 + \sigma_2^2 \delta(x - x')$$

- As usual covariance function parameters are interpretable
  - σ_f^2 — marginal function variance
  - σ_1^2 — variance of bias
  - σ_2^2 — noise variance
  - r — lengthscale
  - $\alpha \geq 1$ — roughness
- Once the mean and covariance functions are defined, everything else about GPs follows from the basic rules of probability applied to multivariate Gaussians

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



Comparison of GP realizations vs. different covariance functions



Example: Constructing new covariances from old

There are several ways to combine covariances (seminar)

- Sum: $K(x, x') = K_1(x, x') + K_2(x, x')$
- Product: $K(x, x') = K_1(x, x') \cdot K_2(x, x')$
- Convolution: $K(x, x') = \int dz dz' h(x, z) K(z, z') h(x', z')$ (blurring of process with kernel h)

1 Motivation

2 Gaussian Process model

3 GP regression

4 Learning Gaussian Process model

5 Gaussian Process classification

6 Take-home messages

7 Afterword: Random Features. Numerical Stability. Non-stationary GP

Using GP for nonlinear regression

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- We can use heteroscedastic noise and/or non-Gaussian noise models

Using GP for nonlinear regression

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- We can use heteroscedastic noise and/or non-Gaussian noise models

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- We can use heteroscedastic noise and/or non-Gaussian noise models

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- We can use heteroscedastic noise and/or non-Gaussian noise models

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$
\begin{aligned}
y_i &= f(\mathbf{x}_i) + \varepsilon_i, \\
f &\sim \mathcal{GP}(\cdot|0, K) \\
\varepsilon_i &\sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}
\end{aligned}
$$

- We can use heteroscedastic noise and/or non-Gaussian noise models

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

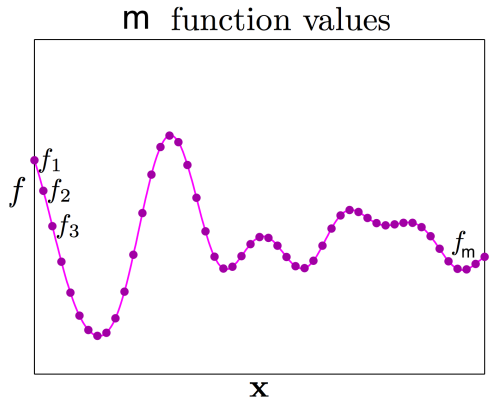
$$
\begin{aligned}
y_i &= f(\mathbf{x}_i) + \varepsilon_i, \\
f &\sim \mathcal{GP}(\cdot|0, K) \\
\varepsilon_i &\sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}
\end{aligned}
$$

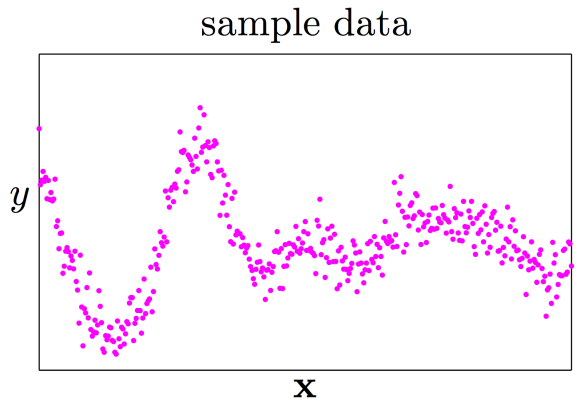
- We can use heteroscedastic noise and/or non-Gaussian noise models

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$
\begin{aligned}
y_i &= f(\mathbf{x}_i) + \varepsilon_i, \\
f &\sim \mathcal{GP}(\cdot|0, K) \\
\varepsilon_i &\sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}
\end{aligned}
$$

- We can use heteroscedastic noise and/or non-Gaussian noise models





Using GP for nonlinear regression

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- The prior is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The noise model, or likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma^2 \mathrm{I}_m\right)$$

- Integrating over the function variables $\mathbf{f}$ we get the marginal likelihood

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f} = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathrm{I}_m)$$

Using GP for nonlinear regression

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- The prior is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The noise model, or likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma^2 \mathrm{I}_m\right)$$

- Integrating over the function variables $\mathbf{f}$ we get the marginal likelihood

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f} = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathrm{I}_m)$$

Using GP for nonlinear regression

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- The prior is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The noise model, or likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma^2 \mathrm{I}_m\right)$$

- Integrating over the function variables $\mathbf{f}$ we get the marginal likelihood

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f} = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathrm{I}_m)$$

Using GP for nonlinear regression

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- The prior is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The noise model, or likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma^2 \mathrm{I}_m\right)$$

- Integrating over the function variables $\mathbf{f}$ we get the marginal likelihood

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f} = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathrm{I}_m)$$

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- The prior is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The noise model, or likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma^2 \mathrm{I}_m\right)$$

- Integrating over the function variables $\mathbf{f}$ we get the marginal likelihood

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f} = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathrm{I}_m)$$

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- The prior is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The noise model, or likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma^2 \mathrm{I}_m\right)$$

- Integrating over the function variables $\mathbf{f}$ we get the marginal likelihood

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f} = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathrm{I}_m)$$

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$
$$f \sim \mathcal{GP}(\cdot|0, K)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}$$

- The prior is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The noise model, or likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma^2 \mathrm{I}_m\right)$$

- Integrating over the function variables $\mathbf{f}$ we get the marginal likelihood

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f} = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathrm{I}_m)$$

- Training data set $S_m = \{\mathbf{X}, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Model:

$$
\begin{aligned}
y_i &= f(\mathbf{x}_i) + \varepsilon_i, \\
f &\sim \mathcal{GP}(\cdot|0, K) \\
\varepsilon_i &\sim \mathcal{N}(0, \sigma^2) \text{ is a white noise}
\end{aligned}
$$

- The prior is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$$

- The noise model, or likelihood is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma^2 \mathrm{I}_m\right)$$

- Integrating over the function variables $\mathbf{f}$ we get the marginal likelihood

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f} = \mathcal{N}(\mathbf{0},\, \mathbf{K} + \sigma^2 \mathrm{I}_m)$$

Prediction

- Let us denote input test point as $\mathbf{x}_*$, and output

$$y_* = f_* + \varepsilon_*, \;\; f_* = f(\mathbf{x}_*)$$

- Consider joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$

- What we know about noiseless value $f(\mathbf{x}_*)$?

Prediction

- Let us denote input test point as $\mathbf{x}_*$, and output

$$y_* = f_* + \varepsilon_*, \;\; f_* = f(\mathbf{x}_*)$$

- Consider joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$

- What we know about noiseless value $f(\mathbf{x}_*)$?

Prediction

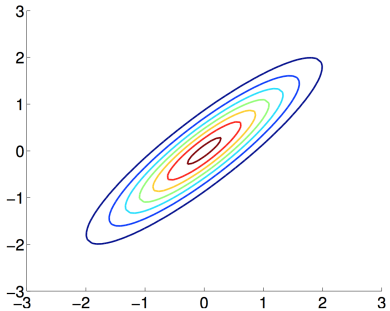
- Let us denote input test point as $\mathbf{x}_*$, and output

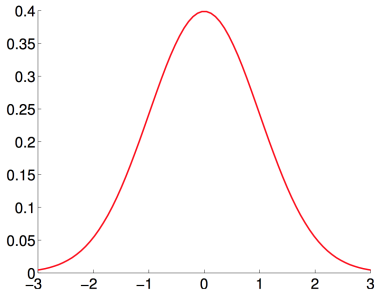
$$y_* = f_* + \varepsilon_*, \;\; f_* = f(\mathbf{x}_*)$$

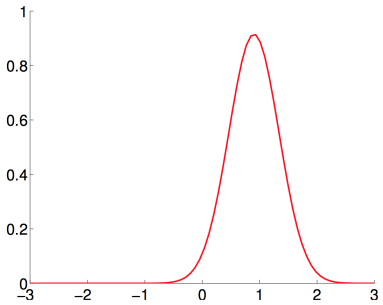
- Consider joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$
- What we know about noiseless value $f(\mathbf{x}_*)$?







$p(\mathbf{f}_1, \mathbf{f}_2) \sim \mathcal{N}(\mathbf{f}_1, \mathbf{f}_2 | \boldsymbol{\mu}, \boldsymbol{\Sigma})$

Joint

$p(\mathbf{f}_1)$

Marginal

$p(\mathbf{f}_1 | \mathbf{f}_2)$

Conditional

The **marginal** and **conditional** distributions are also Gaussians:

$$\begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^\top & \boldsymbol{\Sigma}_{22} \end{bmatrix} \right),$$

$$p(\mathbf{f}_1) = \int p(\mathbf{f}_1, \mathbf{f}_2) d\mathbf{f}_2 = \mathcal{N}\left(\mathbf{f}_1 | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}\right)$$

$$p(\mathbf{f}_1 | \mathbf{f}_2) = \mathcal{N}\left(\mathbf{f}_1 | \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{f}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{12}^\top\right)$$

Prediction

- Joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$

- Condition on training outputs we get

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$

$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- In fact μ_* has the form

$$\mu_* = \sum_{i=1}^m \alpha_i K(\mathbf{x}_*, \mathbf{x}_i), \quad \boldsymbol{\alpha} = [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y}$$

- Computational complexity: $O(m^3)$ for inversion in $\boldsymbol{\alpha}$, $O(m)$ for μ_* and $O(m^2)$ for σ_*^2

Prediction

- Joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$
- Condition on training outputs we get

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- In fact μ_* has the form

$$\mu_* = \sum_{i=1}^m \alpha_i K(\mathbf{x}_*, \mathbf{x}_i), \quad \boldsymbol{\alpha} = [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y}$$

- Computational complexity: $O(m^3)$ for inversion in $\boldsymbol{\alpha}$, $O(m)$ for μ_* and $O(m^2)$ for σ_*^2

Prediction

- Joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$

- Condition on training outputs we get

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$

$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- In fact μ_* has the form

$$\mu_* = \sum_{i=1}^m \alpha_i K(\mathbf{x}_*, \mathbf{x}_i), \quad \boldsymbol{\alpha} = [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y}$$

- Computational complexity: $O(m^3)$ for inversion in $\boldsymbol{\alpha}$, $O(m)$ for μ_* and $O(m^2)$ for σ_*^2

Prediction

- Joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$

- Condition on training outputs we get

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- In fact μ_* has the form

$$\mu_* = \sum_{i=1}^m \alpha_i K(\mathbf{x}_*, \mathbf{x}_i), \quad \boldsymbol{\alpha} = [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y}$$

- Computational complexity: $O(m^3)$ for inversion in $\boldsymbol{\alpha}$, $O(m)$ for μ_* and $O(m^2)$ for σ_*^2

Prediction

- Joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$

- Condition on training outputs we get

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- In fact μ_* has the form

$$\mu_* = \sum_{i=1}^m \alpha_i K(\mathbf{x}_*, \mathbf{x}_i), \quad \boldsymbol{\alpha} = [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y}$$

- Computational complexity: $O(m^3)$ for inversion in $\boldsymbol{\alpha}$, $O(m)$ for μ_* and $O(m^2)$ for σ_*^2

Prediction

- Joint training and test marginal likelihood

$$p(\mathbf{y}, f_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathrm{I}_m & \mathbf{k}_* \\ \mathbf{k}_*^\top & K_{**} \end{bmatrix}\right),$$

  where $\mathbf{k}_* = \{K(\mathbf{x}_*, \mathbf{x}_i)\}_{i=1}^m$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$
- Condition on training outputs we get

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- In fact μ_* has the form

$$\mu_* = \sum_{i=1}^m \alpha_i K(\mathbf{x}_*, \mathbf{x}_i), \quad \boldsymbol{\alpha} = [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y}$$

- Computational complexity: $O(m^3)$ for inversion in $\boldsymbol{\alpha}$, $O(m)$ for μ_* and $O(m^2)$ for σ_*^2

Interpolation

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$

$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

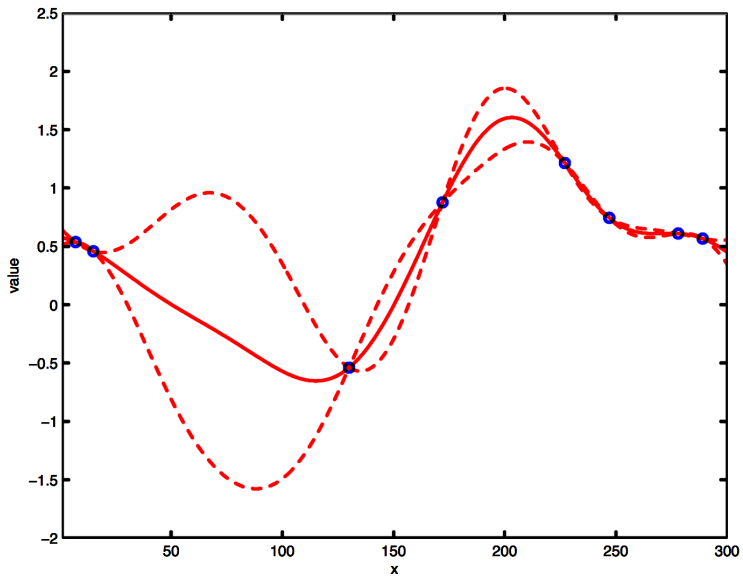
- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)



Smoothing

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)

- $p(y_*|\mathbf{y}) = \mathcal{N}\left(y_*|\mu_*, \sigma_*^2 + \sigma^2\right)$ predicts what we'll see next

Smoothing

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)

- $p(y_*|\mathbf{y}) = \mathcal{N}\left(y_*|\mu_*, \sigma_*^2 + \sigma^2\right)$ predicts what we'll see next

Smoothing

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$

$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)

- $p(y_*|\mathbf{y}) = \mathcal{N}\left(y_*|\mu_*, \sigma_*^2 + \sigma^2\right)$ predicts what we'll see next

Smoothing

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)

- $p(y_*|\mathbf{y}) = \mathcal{N}\left(y_*|\mu_*, \sigma_*^2 + \sigma^2\right)$ predicts what we'll see next

Smoothing

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)

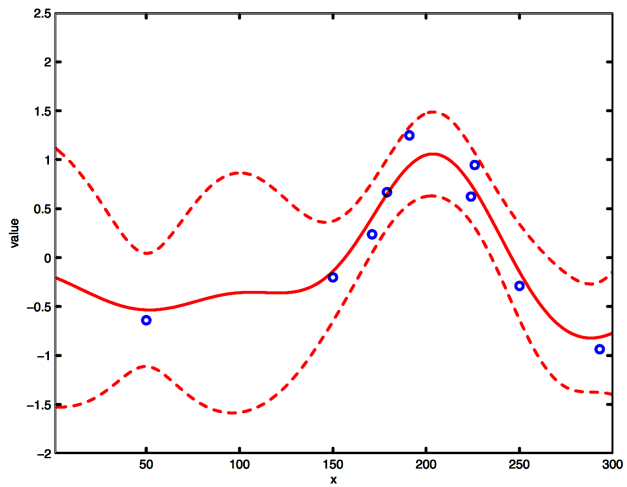
- $p(y_*|\mathbf{y}) = \mathcal{N}\left(y_*|\mu_*, \sigma_*^2 + \sigma^2\right)$ predicts what we'll see next

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)



- $p(y_*|\mathbf{y}) = \mathcal{N}\left(y_*|\mu_*, \sigma_*^2 + \sigma^2\right)$ predicts what we'll see next

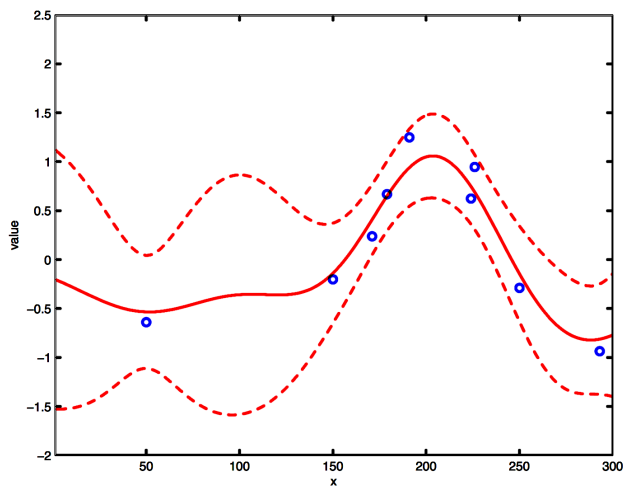
Smoothing

- Conditional distribution

$$p(f_*|\mathbf{y}) = \mathcal{N}\left(f_*|\mu_*, \sigma_*^2\right),$$

$$\mu_* = \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{y},$$
$$\sigma_*^2 = K_{**} - \mathbf{k}_*^\top [\mathbf{K} + \sigma^2 \mathrm{I}_m]^{-1} \mathbf{k}_*$$

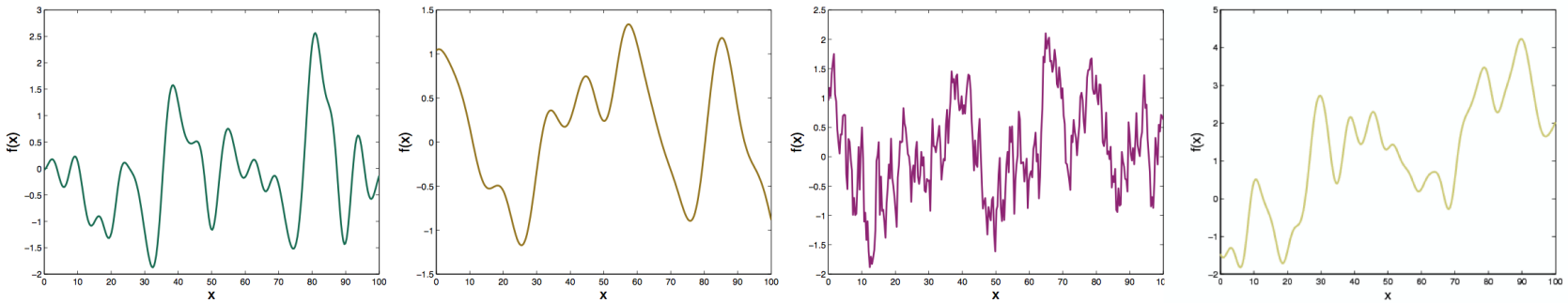
- What we know about noiseless value $f_* = f(\mathbf{x}_*)$? Let us assume that $\sigma = 0$ (interpolation)



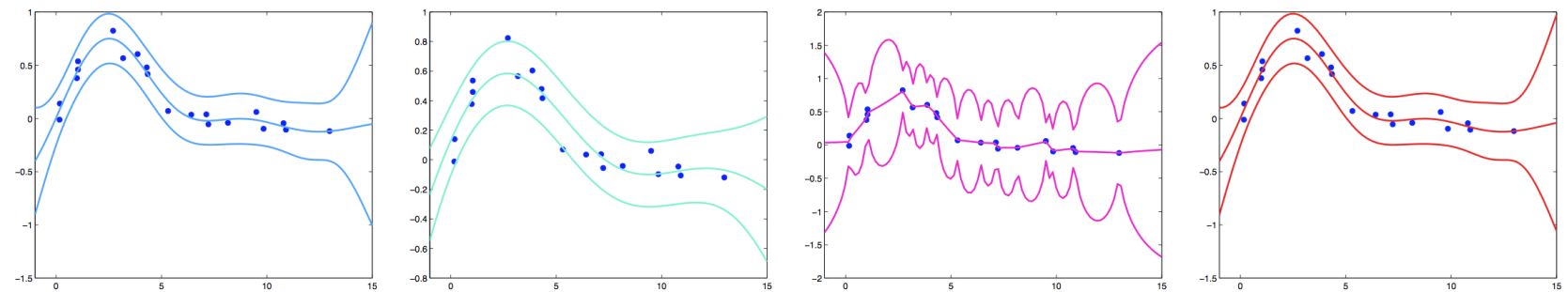
- $p(y_*|\mathbf{y}) = \mathcal{N}\left(y_*|\mu_*, \sigma_*^2 + \sigma^2\right)$ predicts what we'll see next

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

A sample from the prior for each covariance function:



Corresponding predictions, mean with two standard deviations:



1 Motivation

2 Gaussian Process model

3 GP regression

4 Learning Gaussian Process model

5 Gaussian Process classification

6 Take-home messages

7 Afterword: Random Features. Numerical Stability. Non-stationary GP

Determination of covariance function parameters

- Advantage of the probabilistic GP framework — ability to choose hyperparameters and covariances directly from the training data
- Other models, e.g. SVMs, require cross-validation
- GP: minimize negative log marginal likelihood $\mathcal{L}(\boldsymbol{\theta})$ wrt covariance function parameters and noise level $\boldsymbol{\theta}$. Since

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathrm{I}_m),$$

then

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\log\det \mathbf{C}(\boldsymbol{\theta})}_{regularization} + \underbrace{\frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}(\boldsymbol{\theta})\mathbf{y}}_{data-fit} + \frac{m}{2}\log(2\pi),$$

where $\mathbf{C} = \mathbf{K} + \sigma^2 \mathrm{I}_m$

- Uncertainty in function variables $\mathbf{f}$ is taken into account

Determination of covariance function parameters

- Advantage of the probabilistic GP framework — ability to choose hyperparameters and covariances directly from the training data
- Other models, e.g. SVMs, require cross-validation
- GP: minimize negative log marginal likelihood $\mathcal{L}(\boldsymbol{\theta})$ wrt covariance function parameters and noise level $\boldsymbol{\theta}$. Since

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathrm{I}_m),$$

then

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\log\det \mathbf{C}(\boldsymbol{\theta})}_{regularization} + \underbrace{\frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}(\boldsymbol{\theta})\mathbf{y}}_{data-fit} + \frac{m}{2}\log(2\pi),$$

where $\mathbf{C} = \mathbf{K} + \sigma^2 \mathrm{I}_m$

- Uncertainty in function variables $\mathbf{f}$ is taken into account

- Advantage of the probabilistic GP framework — ability to choose hyperparameters and covariances directly from the training data
- Other models, e.g. SVMs, require cross-validation
- **GP**: minimize negative log marginal likelihood $\mathcal{L}(\boldsymbol{\theta})$ wrt covariance function parameters and noise level $\boldsymbol{\theta}$. Since

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0},\, \mathbf{K} + \sigma^2 \mathrm{I}_m),$$

then

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\log\det\mathbf{C}(\boldsymbol{\theta})}_{regularization} + \underbrace{\frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}(\boldsymbol{\theta})\mathbf{y}}_{data-fit} + \frac{m}{2}\log(2\pi),$$

where $\mathbf{C} = \mathbf{K} + \sigma^2 \mathrm{I}_m$

- Uncertainty in function variables $\mathbf{f}$ is taken into account

- Advantage of the probabilistic GP framework — ability to choose hyperparameters and covariances directly from the training data
- Other models, e.g. SVMs, require cross-validation
- **GP**: minimize negative log marginal likelihood $\mathcal{L}(\boldsymbol{\theta})$ wrt covariance function parameters and noise level $\boldsymbol{\theta}$. Since

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0},\, \mathbf{K} + \sigma^2 \mathrm{I}_m),$$

then

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\log\det\mathbf{C}(\boldsymbol{\theta})}_{regularization} + \underbrace{\frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}(\boldsymbol{\theta})\mathbf{y}}_{data-fit} + \frac{m}{2}\log(2\pi),$$

where $\mathbf{C} = \mathbf{K} + \sigma^2 \mathrm{I}_m$

- Uncertainty in function variables $\mathbf{f}$ is taken into account

- Advantage of the probabilistic GP framework — ability to choose hyperparameters and covariances directly from the training data
- Other models, e.g. SVMs, require cross-validation
- **GP**: minimize negative log marginal likelihood $\mathcal{L}(\boldsymbol{\theta})$ wrt covariance function parameters and noise level $\boldsymbol{\theta}$. Since

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0},\, \mathbf{K} + \sigma^2 \mathrm{I}_m),$$

then

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\log\det\mathbf{C}(\boldsymbol{\theta})}_{regularization} + \underbrace{\frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}(\boldsymbol{\theta})\mathbf{y}}_{data-fit} + \frac{m}{2}\log(2\pi),$$

where $\mathbf{C} = \mathbf{K} + \sigma^2 \mathrm{I}_m$

- Uncertainty in function variables $\mathbf{f}$ is taken into account

- Advantage of the probabilistic GP framework — ability to choose hyperparameters and covariances directly from the training data
- Other models, e.g. SVMs, require cross-validation
- **GP**: minimize negative log marginal likelihood $\mathcal{L}(\boldsymbol{\theta})$ wrt covariance function parameters and noise level $\boldsymbol{\theta}$. Since

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0},\, \mathbf{K} + \sigma^2 \mathrm{I}_m),$$

then

$$\mathcal{L} = -\log p(\mathbf{y}|\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\log\det\mathbf{C}(\boldsymbol{\theta})}_{regularization} + \underbrace{\frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}(\boldsymbol{\theta})\mathbf{y}}_{data-fit} + \frac{m}{2}\log(2\pi),$$

where $\mathbf{C} = \mathbf{K} + \sigma^2 \mathrm{I}_m$

- Uncertainty in function variables $\mathbf{f}$ is taken into account

Gradient based optimization

- Minimization of $\mathcal{L}(\boldsymbol{\theta})$ is a non-convex optimization problem
- Standard gradient based techniques, such as CG or quasi-Newton
- Gradients:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\right) - \frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{C}^{-1}\mathbf{y}$$

- Local minima, but usually not much of a problem with few parameters, and/or using restarts
- Use weighted sums of covariances and let ML choose
- Each iteration of optimization costs $O(m^3)$

Gradient based optimization

- Minimization of $\mathcal{L}(\boldsymbol{\theta})$ is a non-convex optimization problem
- Standard gradient based techniques, such as CG or quasi-Newton
- Gradients:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\right) - \frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{C}^{-1}\mathbf{y}$$

- Local minima, but usually not much of a problem with few parameters, and/or using restarts
- Use weighted sums of covariances and let ML choose
- Each iteration of optimization costs $O(m^3)$

- Minimization of $\mathcal{L}(\boldsymbol{\theta})$ is a non-convex optimization problem
- Standard gradient based techniques, such as CG or quasi-Newton
- Gradients:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\right) - \frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{C}^{-1}\mathbf{y}$$

- Local minima, but usually not much of a problem with few parameters, and/or using restarts
- Use weighted sums of covariances and let ML choose
- Each iteration of optimization costs $O(m^3)$

- Minimization of $\mathcal{L}(\boldsymbol{\theta})$ is a non-convex optimization problem
- Standard gradient based techniques, such as CG or quasi-Newton
- Gradients:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\right) - \frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{C}^{-1}\mathbf{y}$$

- Local minima, but usually not much of a problem with few parameters, and/or using restarts
- Use weighted sums of covariances and let ML choose
- Each iteration of optimization costs $O(m^3)$

- Minimization of $\mathcal{L}(\boldsymbol{\theta})$ is a non-convex optimization problem
- Standard gradient based techniques, such as CG or quasi-Newton
- Gradients:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\right) - \frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{C}^{-1}\mathbf{y}$$

- Local minima, but usually not much of a problem with few parameters, and/or using restarts
- Use weighted sums of covariances and let ML choose
- Each iteration of optimization costs $O(m^3)$

- Minimization of $\mathcal{L}(\boldsymbol{\theta})$ is a non-convex optimization problem
- Standard gradient based techniques, such as CG or quasi-Newton
- Gradients:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{2}\mathrm{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\right) - \frac{1}{2}\mathbf{y}^\top \mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \theta_i}\mathbf{C}^{-1}\mathbf{y}$$

- Local minima, but usually not much of a problem with few parameters, and/or using restarts
- Use weighted sums of covariances and let ML choose
- Each iteration of optimization costs $O(m^3)$

1 Motivation

2 Gaussian Process model

3 GP regression

4 Learning Gaussian Process model

5 Gaussian Process classification

6 Take-home messages

7 Afterword: Random Features. Numerical Stability. Non-stationary GP

Logistic and probit regression

- Binary classification task: $y = \pm 1$
- GLM likelihood: $p(y = +1|\mathbf{x}, \mathbf{w}) = \pi(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x}^\top)$
- $\sigma(z)$ — sigmoid function such as the logistic or cumulative normal
- Weight space viewpoint: prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w)$
- Function space viewpoint: let $f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})^\top$, then likelihood $\pi(\mathbf{x}) = \sigma(f(\mathbf{x}))$, Gaussian prior on $\mathbf{f}$

Logistic and probit regression

- Binary classification task: $y = \pm 1$
- GLM likelihood: $p(y = +1|\mathbf{x}, \mathbf{w}) = \pi(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x}^\top)$
- $\sigma(z)$ — sigmoid function such as the logistic or cumulative normal
- Weight space viewpoint: prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w)$
- Function space viewpoint: let $f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})^\top$, then likelihood $\pi(\mathbf{x}) = \sigma(f(\mathbf{x}))$, Gaussian prior on $\mathbf{f}$

Logistic and probit regression

- Binary classification task: $y = \pm 1$
- GLM likelihood: $p(y = +1|\mathbf{x}, \mathbf{w}) = \pi(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x}^\top)$
- $\sigma(z)$ — sigmoid function such as the logistic or cumulative normal
- Weight space viewpoint: prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w)$
- Function space viewpoint: let $f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})^\top$, then likelihood $\pi(\mathbf{x}) = \sigma(f(\mathbf{x}))$, Gaussian prior on $\mathbf{f}$

Logistic and probit regression

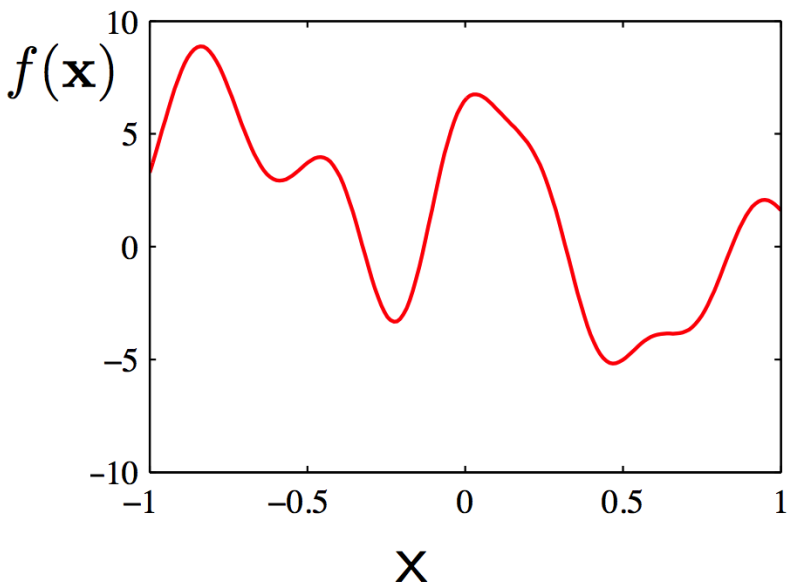
- Binary classification task: $y = \pm 1$
- GLM likelihood: $p(y = +1|\mathbf{x}, \mathbf{w}) = \pi(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x}^\top)$
- $\sigma(z)$ — sigmoid function such as the logistic or cumulative normal
- Weight space viewpoint: prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w)$
- Function space viewpoint: let $f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})^\top$, then likelihood $\pi(\mathbf{x}) = \sigma(f(\mathbf{x}))$, Gaussian prior on $\mathbf{f}$

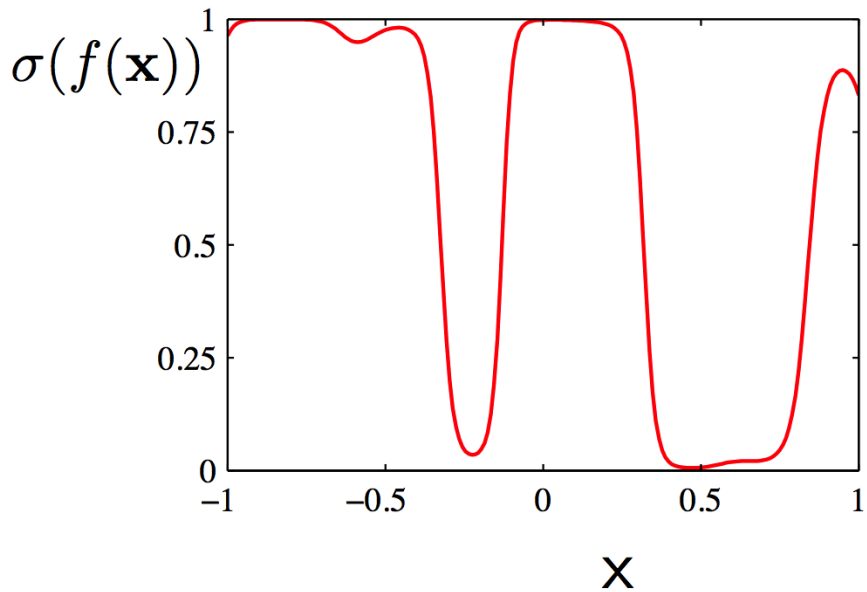
Logistic and probit regression

- Binary classification task: $y = \pm 1$
- GLM likelihood: $p(y = +1|\mathbf{x}, \mathbf{w}) = \pi(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x}^\top)$
- $\sigma(z)$ — sigmoid function such as the logistic or cumulative normal
- Weight space viewpoint: prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w)$
- Function space viewpoint: let $f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})^\top$, then likelihood $\pi(\mathbf{x}) = \sigma(f(\mathbf{x}))$, Gaussian prior on $\mathbf{f}$

Logistic and probit regression

- Binary classification task: $y = \pm 1$
- GLM likelihood: $p(y = +1|\mathbf{x}, \mathbf{w}) = \pi(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x}^\top)$
- $\sigma(z)$ — sigmoid function such as the logistic or cumulative normal
- Weight space viewpoint: prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w)$
- Function space viewpoint: let $f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})^\top$, then likelihood $\pi(\mathbf{x}) = \sigma(f(\mathbf{x}))$, Gaussian prior on $\mathbf{f}$





GP classification

- Place a GP prior directly on $f(\mathbf{x})$
- Use a sigmoidal likelihood: $p(y = +1|f) = \sigma(f)$
- Non-Gaussian likelihood makes integrating over $\mathbf{f}$ intractable:

$$p(f_*|\mathbf{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f},$$

where the posterior $p(\mathbf{f}|\mathbf{y}) \sim p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$

- Make tractable by using a Gaussian approximation to posterior.

Then prediction:

$$p(y_* = +1|\mathbf{y}) = \int \sigma(f_*)p(f_*|\mathbf{y})df_*$$

GP classification

- Place a GP prior directly on $f(\mathbf{x})$
- Use a sigmoidal likelihood: $p(y = +1|f) = \sigma(f)$
- Non-Gaussian likelihood makes integrating over $\mathbf{f}$ intractable:

$$p(f_*|\mathbf{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f},$$

where the posterior $p(\mathbf{f}|\mathbf{y}) \sim p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$

- Make tractable by using a Gaussian approximation to posterior.
Then prediction:

$$p(y_* = +1|\mathbf{y}) = \int \sigma(f_*)p(f_*|\mathbf{y})df_*$$

- Place a GP prior directly on $f(\mathbf{x})$
- Use a sigmoidal likelihood: $p(y = +1|f) = \sigma(f)$
- Non-Gaussian likelihood makes integrating over $\mathbf{f}$ intractable:

$$p(f_*|\mathbf{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f},$$

where the posterior $p(\mathbf{f}|\mathbf{y}) \sim p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$

- Make tractable by using a Gaussian approximation to posterior.

Then prediction:

$$p(y_* = +1|\mathbf{y}) = \int \sigma(f_*)p(f_*|\mathbf{y})df_*$$

- Place a GP prior directly on $f(\mathbf{x})$
- Use a sigmoidal likelihood: $p(y = +1|f) = \sigma(f)$
- Non-Gaussian likelihood makes integrating over $\mathbf{f}$ intractable:

$$p(f_*|\mathbf{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f},$$

where the posterior $p(\mathbf{f}|\mathbf{y}) \sim p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$

- Make tractable by using a Gaussian approximation to posterior.

Then prediction:

$$p(y_* = +1|\mathbf{y}) = \int \sigma(f_*)p(f_*|\mathbf{y})df_*$$

- Place a GP prior directly on $f(\mathbf{x})$
- Use a sigmoidal likelihood: $p(y = +1|f) = \sigma(f)$
- Non-Gaussian likelihood makes integrating over $\mathbf{f}$ intractable:

$$p(f_*|\mathbf{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f},$$

  where the posterior $p(\mathbf{f}|\mathbf{y}) \sim p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$
- Make tractable by using a Gaussian approximation to posterior.

  Then prediction:

$$p(y_* = +1|\mathbf{y}) = \int \sigma(f_*)p(f_*|\mathbf{y})df_*$$

- Place a GP prior directly on $f(\mathbf{x})$
- Use a sigmoidal likelihood: $p(y = +1|f) = \sigma(f)$
- Non-Gaussian likelihood makes integrating over $\mathbf{f}$ intractable:

$$p(f_*|\mathbf{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f},$$

  where the posterior $p(\mathbf{f}|\mathbf{y}) \sim p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$
- Make tractable by using a Gaussian approximation to posterior.
  Then prediction:

$$p(y_* = +1|\mathbf{y}) = \int \sigma(f_*)p(f_*|\mathbf{y})df_*$$

- Place a GP prior directly on $f(\mathbf{x})$
- Use a sigmoidal likelihood: $p(y = +1|f) = \sigma(f)$
- Non-Gaussian likelihood makes integrating over $\mathbf{f}$ intractable:

$$p(f_*|\mathbf{y}) = \int p(f_*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f},$$

  where the posterior $p(\mathbf{f}|\mathbf{y}) \sim p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$
- Make tractable by using a Gaussian approximation to posterior. Then prediction:

$$p(y_* = +1|\mathbf{y}) = \int \sigma(f_*)p(f_*|\mathbf{y})df_*$$

Two common ways to make Gaussian approximation to posterior:

1. Laplace approximation. Second order Taylor approximation about mode of posterior
2. Expectation propagation (EP). EP can be thought of as approximately minimizing $\mathrm{KL}\left[p(\mathbf{f}|\mathbf{y})||q(\mathbf{f}|\mathbf{y})\right]$ by an iterative procedure

- Kuss and Rasmussen [2005] evaluate both methods experimentally and find EP to be often better
- Classification accuracy on digits data sets comparable to SVMs
- Advantages: uncertainty quantification, probabilistic predictions, kernel parameters by ML

Two common ways to make Gaussian approximation to posterior:

1. Laplace approximation. Second order Taylor approximation about mode of posterior
2. Expectation propagation (EP). EP can be thought of as approximately minimizing $\mathrm{KL}\left[p(\mathbf{f}|\mathbf{y})||q(\mathbf{f}|\mathbf{y})\right]$ by an iterative procedure

- Kuss and Rasmussen [2005] evaluate both methods experimentally and find EP to be often better
- Classification accuracy on digits data sets comparable to SVMs
- Advantages: uncertainty quantification, probabilistic predictions, kernel parameters by ML

Two common ways to make Gaussian approximation to posterior:

1. Laplace approximation. Second order Taylor approximation about mode of posterior
2. Expectation propagation (EP). EP can be thought of as approximately minimizing $\mathrm{KL}\left[p(\mathbf{f}|\mathbf{y})||q(\mathbf{f}|\mathbf{y})\right]$ by an iterative procedure

- Kuss and Rasmussen [2005] evaluate both methods experimentally and find EP to be often better
- Classification accuracy on digits data sets comparable to SVMs
- Advantages: uncertainty quantification, probabilistic predictions, kernel parameters by ML

Two common ways to make Gaussian approximation to posterior:

1. Laplace approximation. Second order Taylor approximation about mode of posterior
2. Expectation propagation (EP). EP can be thought of as approximately minimizing $\mathrm{KL}\left[p(\mathbf{f}|\mathbf{y})||q(\mathbf{f}|\mathbf{y})\right]$ by an iterative procedure

- Kuss and Rasmussen [2005] evaluate both methods experimentally and find EP to be often better
- Classification accuracy on digits data sets comparable to SVMs
- Advantages: uncertainty quantification, probabilistic predictions, kernel parameters by ML

Two common ways to make Gaussian approximation to posterior:

1. Laplace approximation. Second order Taylor approximation about mode of posterior
2. Expectation propagation (EP). EP can be thought of as approximately minimizing $\mathrm{KL}\left[p(\mathbf{f}|\mathbf{y})||q(\mathbf{f}|\mathbf{y})\right]$ by an iterative procedure

- Kuss and Rasmussen [2005] evaluate both methods experimentally and find EP to be often better
- Classification accuracy on digits data sets comparable to SVMs
- Advantages: uncertainty quantification, probabilistic predictions, kernel parameters by ML

1 Motivation

2 Gaussian Process model

3 GP regression

4 Learning Gaussian Process model

5 Gaussian Process classification

6 Take-home messages

7 Afterword: Random Features. Numerical Stability. Non-stationary GP

My message...

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD

- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand

- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD
- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand
- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

- **Simple to use**
  - Just matrix operations (if likelihoods are Gaussian)
  - Few parameters: relatively easy to set or sample over
  - Predictions are often very good
  - Handle uncertainty in unknown function $\mathbf{f}$
  - Can learn kernel parameters automatically from data, no matter how flexible we wish to make the kernel
  - Can incorporate interpretable noise models and priors over functions
  - Can combine automatic feature selection with learning using ARD

- **No magic bullet**: best results need (at least) careful data scaling, which could be modelled or done by hand

- **The need for approximate inference:**
  - Sometimes Gaussian likelihoods are not enough
  - $O(m^3)$ and $O(m^2)$ costs are bad news for big problems

What's next?

- Multi-task and structured GP learning
- Large scale GP based on Random Fourier features
- Deep GP based on random Fourier features
- Deep GP with Deep NN input features
- Multi-fidelity and multi-criteria Bayesian optimization
- Learning to optimize
- Other probabilistic models, e.g. modeling and optimization based on point processes

What's next?

- Multi-task and structured GP learning
- Large scale GP based on Random Fourier features
- Deep GP based on random Fourier features
- Deep GP with Deep NN input features
- Multi-fidelity and multi-criteria Bayesian optimization
- Learning to optimize
- Other probabilistic models, e.g. modeling and optimization based on point processes

What's next?

- Multi-task and structured GP learning
- Large scale GP based on Random Fourier features
- Deep GP based on random Fourier features
- Deep GP with Deep NN input features
- Multi-fidelity and multi-criteria Bayesian optimization
- Learning to optimize
- Other probabilistic models, e.g. modeling and optimization based on point processes

What's next?

- Multi-task and structured GP learning
- Large scale GP based on Random Fourier features
- Deep GP based on random Fourier features
- Deep GP with Deep NN input features
- Multi-fidelity and multi-criteria Bayesian optimization
- Learning to optimize
- Other probabilistic models, e.g. modeling and optimization based on point processes

What's next?

- Multi-task and structured GP learning
- Large scale GP based on Random Fourier features
- Deep GP based on random Fourier features
- Deep GP with Deep NN input features
- Multi-fidelity and multi-criteria Bayesian optimization
- Learning to optimize
- Other probabilistic models, e.g. modeling and optimization based on point processes

What's next?

- Multi-task and structured GP learning
- Large scale GP based on Random Fourier features
- Deep GP based on random Fourier features
- Deep GP with Deep NN input features
- Multi-fidelity and multi-criteria Bayesian optimization
- Learning to optimize
- Other probabilistic models, e.g. modeling and optimization based on point processes

What's next?

- Multi-task and structured GP learning
- Large scale GP based on Random Fourier features
- Deep GP based on random Fourier features
- Deep GP with Deep NN input features
- Multi-fidelity and multi-criteria Bayesian optimization
- Learning to optimize
- Other probabilistic models, e.g. modeling and optimization based on point processes

1 Motivation

2 Gaussian Process model

3 GP regression

4 Learning Gaussian Process model

5 Gaussian Process classification

6 Take-home messages

7 Afterword: Random Features. Numerical Stability. Non-stationary GP

What else?

- Mercer's theorem: can always decompose covariance function into eigenfunctions and eigenvalues

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{x}')$$

- If sum is finite — back to linear regression. Often sum is infinite, an no analytical expression for eigenfunctions
- However, a lot of kernels can be represented as

$$K(\mathbf{x}, \mathbf{x}') = \int_{\mathbb{R}^d} \phi(\mathbf{x}, \mathbf{w}) \phi(\mathbf{x}', \mathbf{w}) p(\mathbf{w}) d\mathbf{w}$$

Thus we can approximate kernels using random sampling and get approximate non-linear kernel models in linear time $O(m)$

Kernel features

- Mercer's theorem: can always decompose covariance function into eigenfunctions and eigenvalues

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{x}')$$

- If sum is finite — back to linear regression. Often sum is infinite, an no analytical expression for eigenfunctions
- However, a lot of kernels can be represented as

$$K(\mathbf{x}, \mathbf{x}') = \int_{\mathbb{R}^d} \phi(\mathbf{x}, \mathbf{w}) \phi(\mathbf{x}', \mathbf{w}) p(\mathbf{w}) d\mathbf{w}$$

Thus we can approximate kernels using random sampling and get approximate non-linear kernel models in linear time $O(m)$

- Mercer's theorem: can always decompose covariance function into eigenfunctions and eigenvalues

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x}) \psi_i(\mathbf{x}')$$

- If sum is finite — back to linear regression. Often sum is infinite, an no analytical expression for eigenfunctions
- However, a lot of kernels can be represented as

$$K(\mathbf{x}, \mathbf{x}') = \int_{\mathbb{R}^d} \phi(\mathbf{x}, \mathbf{w}) \phi(\mathbf{x}', \mathbf{w}) p(\mathbf{w}) d\mathbf{w}$$

Thus we can approximate kernels using random sampling and get approximate non-linear kernel models in linear time $O(m)$

Degenerate behaviour

Let us consider the squared exponential covariance function

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sum_{i=1}^{d} \theta_i^2 (x_i - x_i')^2\right)$$

When optimizing the likelihood wrt covariance function parameters $\{\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d), \sigma_f, \sigma\}$, the following weird behavior can occur

- MLE optimum is located in the area where $\|\boldsymbol{\theta}\| \sim 0$, and the conditional number of $\mathbf{K}$ is big
- For $\|\boldsymbol{\theta}\| \to \infty$ matrix $\mathbf{K} \to \mathrm{I}_m$. Thus, we have a degenerate approximation
- Dependence of the likelihood $\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ on $\boldsymbol{\theta}$ can be weak

Let us consider the squared exponential covariance function

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sum_{i=1}^{d} \theta_i^2 (x_i - x_i')^2\right)$$

When optimizing the likelihood wrt covariance function parameters $\{\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d), \sigma_f, \sigma\}$, the following weird behavior can occur

- MLE optimum is located in the area where $\|\boldsymbol{\theta}\| \sim 0$, and the conditional number of $\mathbf{K}$ is big
- For $\|\boldsymbol{\theta}\| \to \infty$ matrix $\mathbf{K} \to \mathrm{I}_m$. Thus, we have a degenerate approximation
- Dependence of the likelihood $\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ on $\boldsymbol{\theta}$ can be weak

Let us consider the squared exponential covariance function

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sum_{i=1}^{d} \theta_i^2 (x_i - x_i')^2\right)$$

When optimizing the likelihood wrt covariance function parameters $\{\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d), \sigma_f, \sigma\}$, the following weird behavior can occur

- MLE optimum is located in the area where $\|\boldsymbol{\theta}\| \sim 0$, and the conditional number of $\mathbf{K}$ is big
- For $\|\boldsymbol{\theta}\| \to \infty$ matrix $\mathbf{K} \to \mathrm{I}_m$. Thus, we have a degenerate approximation
- Dependence of the likelihood $\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ on $\boldsymbol{\theta}$ can be weak

Let us consider the squared exponential covariance function

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sum_{i=1}^{d} \theta_i^2 (x_i - x_i')^2\right)$$

When optimizing the likelihood wrt covariance function parameters $\{\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d), \sigma_f, \sigma\}$, the following weird behavior can occur

- MLE optimum is located in the area where $\|\boldsymbol{\theta}\| \sim 0$, and the conditional number of $\mathbf{K}$ is big
- For $\|\boldsymbol{\theta}\| \to \infty$ matrix $\mathbf{K} \to \mathrm{I}_m$. Thus, we have a degenerate approximation
- Dependence of the likelihood $\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ on $\boldsymbol{\theta}$ can be weak

Let us consider the squared exponential covariance function

$$K(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sum_{i=1}^{d} \theta_i^2 (x_i - x_i')^2\right)$$

When optimizing the likelihood wrt covariance function parameters $\left\{\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d), \sigma_f, \sigma\right\}$, the following weird behavior can occur

- MLE optimum is located in the area where $\|\boldsymbol{\theta}\| \sim 0$, and the conditional number of $\mathbf{K}$ is big
- For $\|\boldsymbol{\theta}\| \to \infty$ matrix $\mathbf{K} \to \mathrm{I}_m$. Thus, we have a degenerate approximation
- Dependence of the likelihood $\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$ on $\boldsymbol{\theta}$ can be weak

Degenerate approximation

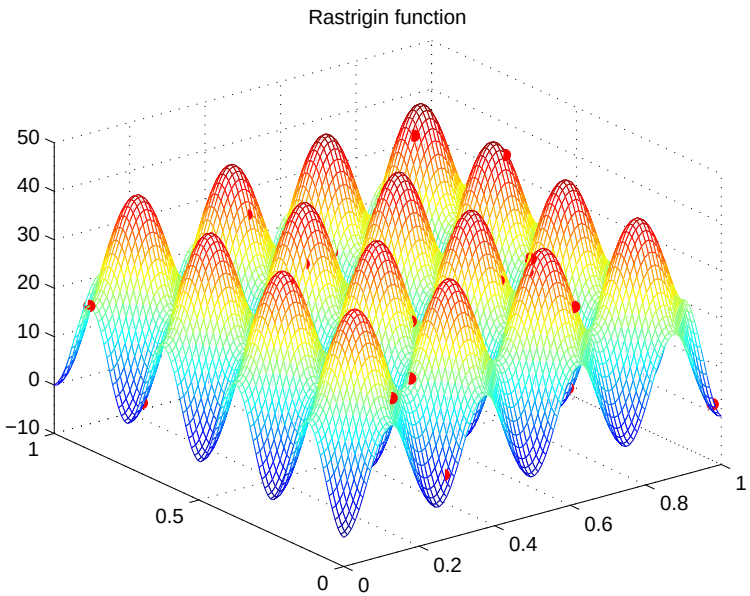


Figure – Rastrigin function

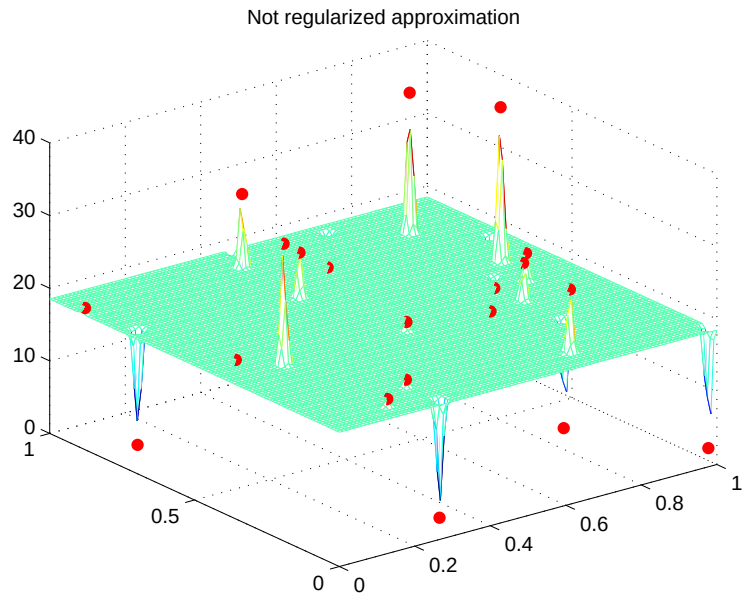
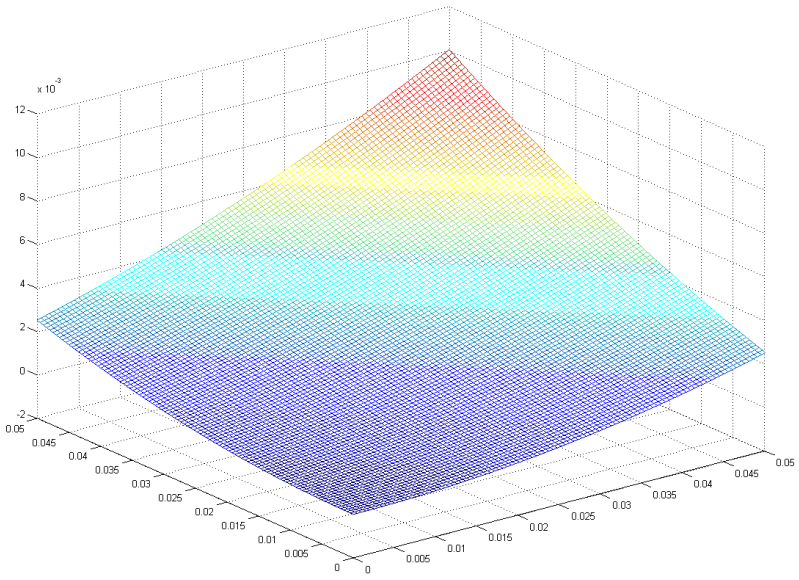
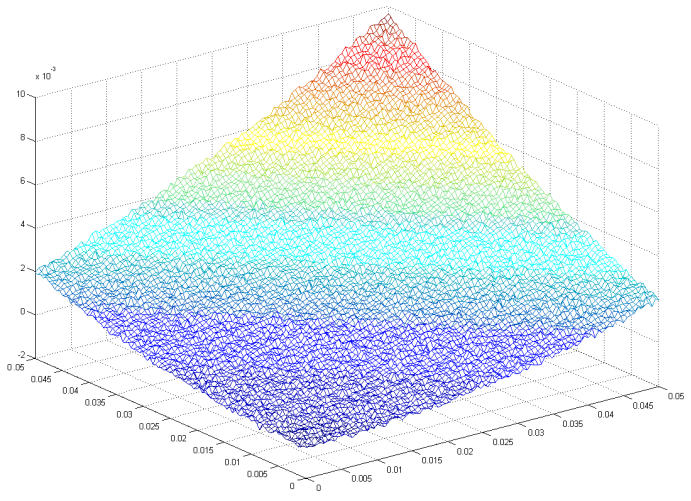


Figure – An approximation of Rastrigin function

Numerical noise





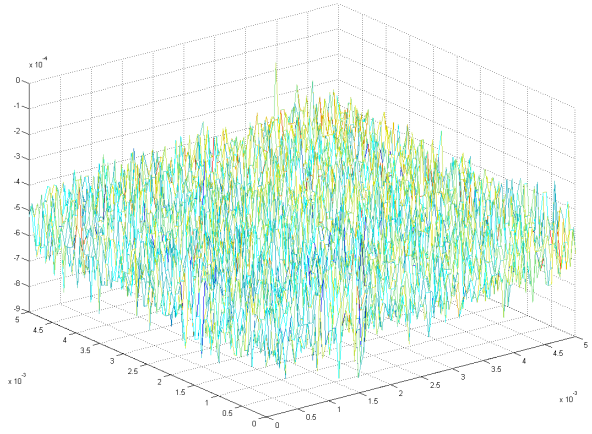


Figure – $y(\mathbf{x}) = (x_1 + x_2)^2$

Bayesian regularization

Let us introduce the following priors on covariance function parameters:

$$\theta_i^2 \in \Gamma(\alpha, \beta), i = 1, \ldots, d$$
$$\sigma^2 \in \Gamma(\alpha_\sigma, \beta_\sigma)$$

Thus log-prior of parameters has the form

$$\log p(\{\boldsymbol{\theta}, \sigma\}) = m(\alpha \log \beta - \log \Gamma(\alpha)) + 2(\alpha - 1) \sum_{i=1}^{m} \log \theta_i +$$
$$- \beta \sum_{i=1}^{m} \theta_i^2 + (\alpha_\sigma \log \beta_\sigma - \log \Gamma(\alpha_\sigma)) + 2(\alpha_\sigma - 1) \log \sigma - \beta_\sigma \sigma^2$$

We obtain MAP estimates for $\{\boldsymbol{\theta}, \sigma\}$:

$$\log p(\mathbf{y}|\mathbf{X}, \{\boldsymbol{\theta}, \sigma\}) + \log p_{\alpha,\beta,\alpha_\sigma,\beta_\sigma}(\{\boldsymbol{\theta}, \sigma\})$$
$$+ \log p(\alpha, \beta, \alpha_\sigma, \beta_\sigma) \to \max_{\{\boldsymbol{\theta},\sigma\},\{\alpha,\beta,\alpha_\sigma,\beta_\sigma\}}$$

Here parameters $\{\alpha, \beta, \alpha_\sigma, \beta_\sigma\}$ are also optimized along with imposed hyper-prior $p(\alpha, \beta, \alpha_\sigma, \beta_\sigma)$, parameters of which are fixed to some constants

Let us introduce the following priors on covariance function parameters:

$$\theta_i^2 \in \Gamma(\alpha, \beta), i = 1, \ldots, d$$
$$\sigma^2 \in \Gamma(\alpha_\sigma, \beta_\sigma)$$

Thus log-prior of parameters has the form

$$\log p(\{\boldsymbol{\theta}, \sigma\}) = m(\alpha \log \beta - \log \Gamma(\alpha)) + 2(\alpha - 1) \sum_{i=1}^{m} \log \theta_i +$$
$$- \beta \sum_{i=1}^{m} \theta_i^2 + (\alpha_\sigma \log \beta_\sigma - \log \Gamma(\alpha_\sigma)) + 2(\alpha_\sigma - 1) \log \sigma - \beta_\sigma \sigma^2$$

We obtain MAP estimates for $\{\boldsymbol{\theta}, \sigma\}$:

$$\log p(\mathbf{y}|\mathbf{X}, \{\boldsymbol{\theta}, \sigma\}) + \log p_{\alpha,\beta,\alpha_\sigma,\beta_\sigma}(\{\boldsymbol{\theta}, \sigma\})$$
$$+ \log p(\alpha, \beta, \alpha_\sigma, \beta_\sigma) \to \max_{\{\boldsymbol{\theta},\sigma\},\{\alpha,\beta,\alpha_\sigma,\beta_\sigma\}}$$

Here parameters $\{\alpha, \beta, \alpha_\sigma, \beta_\sigma\}$ are also optimized along with imposed hyper-prior $p(\alpha, \beta, \alpha_\sigma, \beta_\sigma)$, parameters of which are fixed to some constants

Let us introduce the following priors on covariance function parameters:

$$\theta_i^2 \in \Gamma(\alpha, \beta), i = 1, \ldots, d$$
$$\sigma^2 \in \Gamma(\alpha_\sigma, \beta_\sigma)$$

Thus log-prior of parameters has the form

$$\log p(\{\boldsymbol{\theta}, \sigma\}) = m(\alpha \log \beta - \log \Gamma(\alpha)) + 2(\alpha - 1) \sum_{i=1}^{m} \log \theta_i +$$
$$- \beta \sum_{i=1}^{m} \theta_i^2 + (\alpha_\sigma \log \beta_\sigma - \log \Gamma(\alpha_\sigma)) + 2(\alpha_\sigma - 1) \log \sigma - \beta_\sigma \sigma^2$$

We obtain MAP estimates for $\{\boldsymbol{\theta}, \sigma\}$:

$$\log p(\mathbf{y}|\mathbf{X}, \{\boldsymbol{\theta}, \sigma\}) + \log p_{\alpha,\beta,\alpha_\sigma,\beta_\sigma}(\{\boldsymbol{\theta}, \sigma\})$$
$$+ \log p(\alpha, \beta, \alpha_\sigma, \beta_\sigma) \to \max_{\{\boldsymbol{\theta},\sigma\},\{\alpha,\beta,\alpha_\sigma,\beta_\sigma\}}$$

Here parameters $\{\alpha, \beta, \alpha_\sigma, \beta_\sigma\}$ are also optimized along with imposed hyper-prior $p(\alpha, \beta, \alpha_\sigma, \beta_\sigma)$, parameters of which are fixed to some constants

Let us introduce the following priors on covariance function parameters:

$$\theta_i^2 \in \Gamma(\alpha, \beta), i = 1, \ldots, d$$
$$\sigma^2 \in \Gamma(\alpha_\sigma, \beta_\sigma)$$

Thus log-prior of parameters has the form

$$\log p(\{\boldsymbol{\theta}, \sigma\}) = m(\alpha \log \beta - \log \Gamma(\alpha)) + 2(\alpha - 1) \sum_{i=1}^{m} \log \theta_i +$$
$$- \beta \sum_{i=1}^{m} \theta_i^2 + (\alpha_\sigma \log \beta_\sigma - \log \Gamma(\alpha_\sigma)) + 2(\alpha_\sigma - 1) \log \sigma - \beta_\sigma \sigma^2$$

We obtain MAP estimates for $\{\boldsymbol{\theta}, \sigma\}$:

$$\log p(\mathbf{y}|\mathbf{X}, \{\boldsymbol{\theta}, \sigma\}) + \log p_{\alpha,\beta,\alpha_\sigma,\beta_\sigma}(\{\boldsymbol{\theta}, \sigma\})$$
$$+ \log p(\alpha, \beta, \alpha_\sigma, \beta_\sigma) \to \max_{\{\boldsymbol{\theta},\sigma\},\{\alpha,\beta,\alpha_\sigma,\beta_\sigma\}}$$

Here parameters $\{\alpha, \beta, \alpha_\sigma, \beta_\sigma\}$ are also optimized along with imposed hyper-prior $p(\alpha, \beta, \alpha_\sigma, \beta_\sigma)$, parameters of which are fixed to some constants

Regularized approximation vs. degeneracy

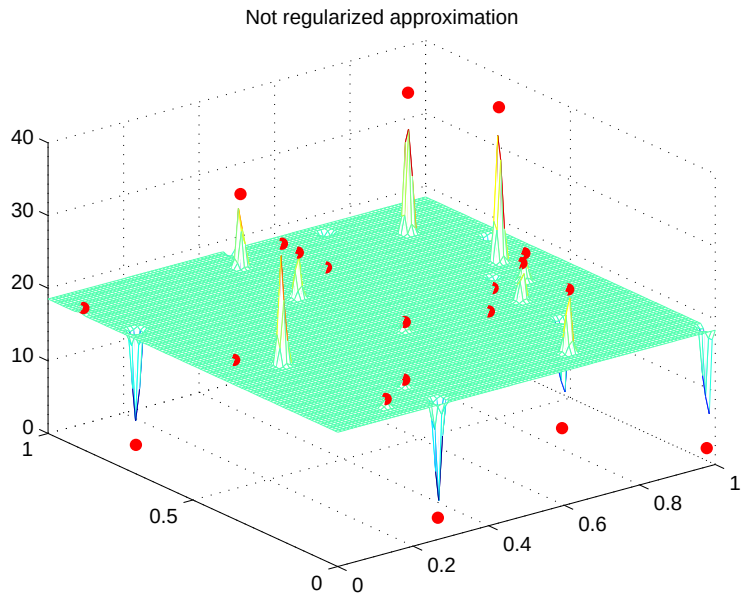


Figure – Degenerate approximation

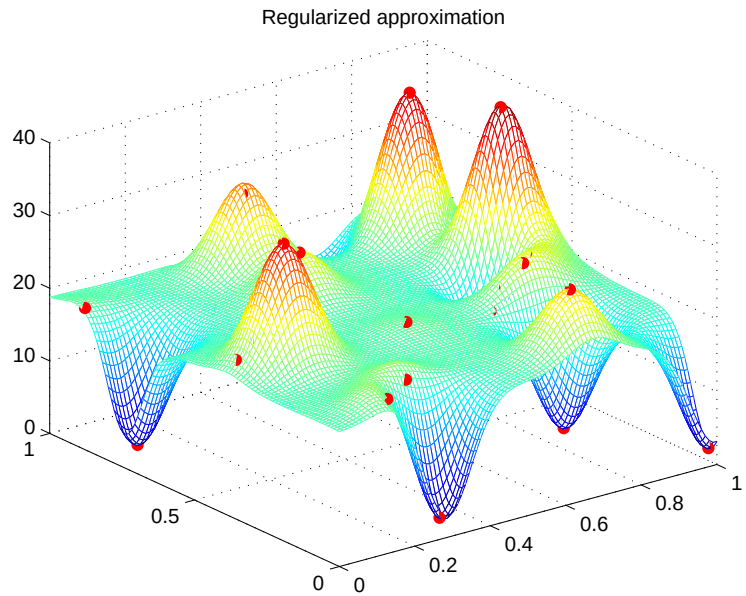
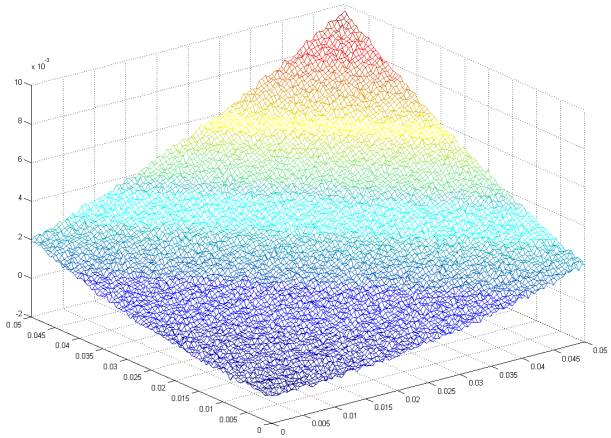
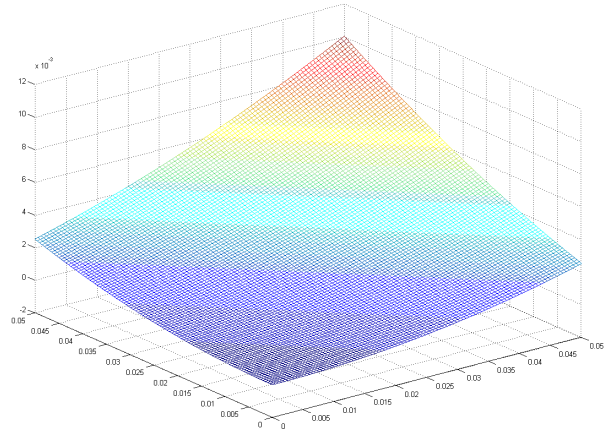
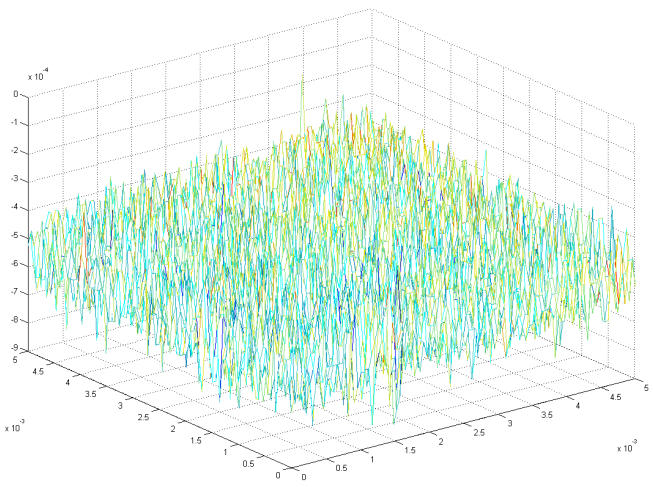


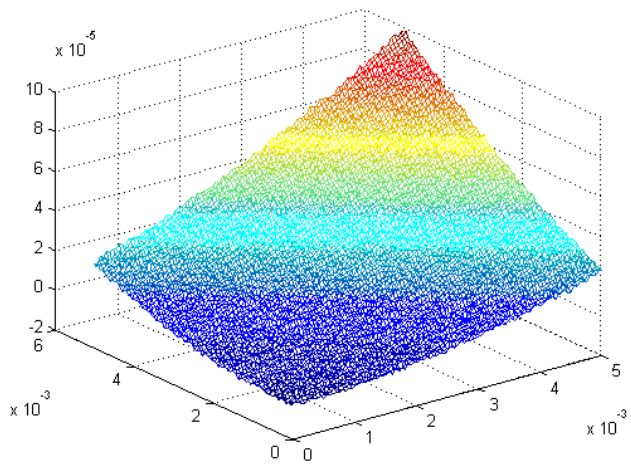
Figure – Approximation with imposed Bayesian regularization

Regularized approximation vs. numerical noise

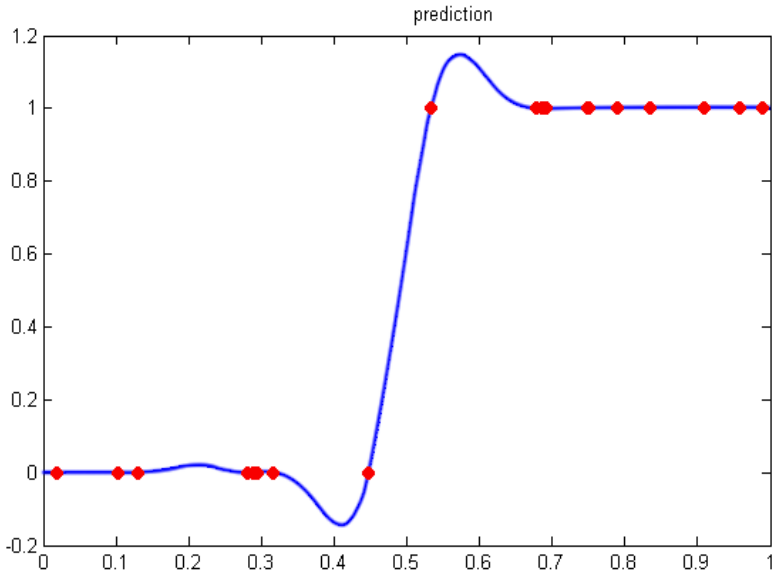








Approximation of Heaviside function for $x \in [0, 1]$, training sample size $m = 20$



Let us assume that

$$y(\mathbf{x}) = \sum_{i=1}^{Q} \alpha_i \psi_i(\mathbf{x}) + f(\mathbf{x}) + \varepsilon(\mathbf{x}),$$

where

- $f(\mathbf{x})$ is a GP,
- $\varepsilon(\mathbf{x})$ is a Gaussian white noise,
- $\{\alpha_i\}_{i=1}^{Q}$ are i.i.d. r.v. with zero mean and variance $\frac{\sigma_0^2}{Q}$,
- $\left\{\psi_i(\mathbf{x})\right\}_{i=1}^{Q}$ is a set of functions

Thus the covariance function of $y(\mathbf{x})$ has the form:

$$\overline{K}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + \sigma_0^2 \boldsymbol{\psi}^\top(\mathbf{x}) \boldsymbol{\psi}(\mathbf{x}') + \sigma^2 \delta(\mathbf{x}, \mathbf{x}'),$$

where $\boldsymbol{\psi}(\mathbf{x}) = \{\psi_i(\mathbf{x}), i = 1, \ldots, Q\}$

Let us assume that

$$y(\mathbf{x}) = \sum_{i=1}^{Q} \alpha_i \psi_i(\mathbf{x}) + f(\mathbf{x}) + \varepsilon(\mathbf{x}),$$

where

- $f(\mathbf{x})$ is a GP,
- $\varepsilon(\mathbf{x})$ is a Gaussian white noise,
- $\{\alpha_i\}_{i=1}^{Q}$ are i.i.d. r.v. with zero mean and variance $\frac{\sigma_0^2}{Q}$,
- $\{\psi_i(\mathbf{x})\}_{i=1}^{Q}$ is a set of functions

Thus the covariance function of $y(\mathbf{x})$ has the form:

$$\overline{K}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + \sigma_0^2 \boldsymbol{\psi}^\top(\mathbf{x}) \boldsymbol{\psi}(\mathbf{x}') + \sigma^2 \delta(\mathbf{x}, \mathbf{x}'),$$

where $\boldsymbol{\psi}(\mathbf{x}) = \{\psi_i(\mathbf{x}), i = 1, \ldots, Q\}$

Let us assume that

$$y(\mathbf{x}) = \sum_{i=1}^{Q} \alpha_i \psi_i(\mathbf{x}) + f(\mathbf{x}) + \varepsilon(\mathbf{x}),$$

where

- $f(\mathbf{x})$ is a GP,
- $\varepsilon(\mathbf{x})$ is a Gaussian white noise,
- $\{\alpha_i\}_{i=1}^{Q}$ are i.i.d. r.v. with zero mean and variance $\frac{\sigma_0^2}{Q}$,
- $\left\{\psi_i(\mathbf{x})\right\}_{i=1}^{Q}$ is a set of functions

Thus the covariance function of $y(\mathbf{x})$ has the form:

$$\overline{K}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + \sigma_0^2 \boldsymbol{\psi}^\top(\mathbf{x}) \boldsymbol{\psi}(\mathbf{x}') + \sigma^2 \delta(\mathbf{x}, \mathbf{x}'),$$

where $\boldsymbol{\psi}(\mathbf{x}) = \{\psi_i(\mathbf{x}), i = 1, \ldots, Q\}$

Let us assume that

$$y(\mathbf{x}) = \sum_{i=1}^{Q} \alpha_i \psi_i(\mathbf{x}) + f(\mathbf{x}) + \varepsilon(\mathbf{x}),$$

where

- $f(\mathbf{x})$ is a GP,
- $\varepsilon(\mathbf{x})$ is a Gaussian white noise,
- $\{\alpha_i\}_{i=1}^{Q}$ are i.i.d. r.v. with zero mean and variance $\frac{\sigma_0^2}{Q}$,
- $\{\psi_i(\mathbf{x})\}_{i=1}^{Q}$ is a set of functions

Thus the covariance function of $y(\mathbf{x})$ has the form:

$$\overline{K}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + \sigma_0^2 \boldsymbol{\psi}^\top(\mathbf{x}) \boldsymbol{\psi}(\mathbf{x}') + \sigma^2 \delta(\mathbf{x}, \mathbf{x}'),$$

where $\boldsymbol{\psi}(\mathbf{x}) = \{\psi_i(\mathbf{x}), i = 1, \ldots, Q\}$

Let us assume that

$$y(\mathbf{x}) = \sum_{i=1}^{Q} \alpha_i \psi_i(\mathbf{x}) + f(\mathbf{x}) + \varepsilon(\mathbf{x}),$$

where

- $f(\mathbf{x})$ is a GP,
- $\varepsilon(\mathbf{x})$ is a Gaussian white noise,
- $\{\alpha_i\}_{i=1}^{Q}$ are i.i.d. r.v. with zero mean and variance $\frac{\sigma_0^2}{Q}$,
- $\left\{\psi_i(\mathbf{x})\right\}_{i=1}^{Q}$ is a set of functions

Thus the covariance function of $y(\mathbf{x})$ has the form:

$$\overline{K}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + \sigma_0^2 \boldsymbol{\psi}^\top(\mathbf{x}) \boldsymbol{\psi}(\mathbf{x}') + \sigma^2 \delta(\mathbf{x}, \mathbf{x}'),$$

where $\boldsymbol{\psi}(\mathbf{x}) = \{\psi_i(\mathbf{x}), i = 1, \ldots, Q\}$

Let us assume that

$$y(\mathbf{x}) = \sum_{i=1}^{Q} \alpha_i \psi_i(\mathbf{x}) + f(\mathbf{x}) + \varepsilon(\mathbf{x}),$$

where

- $f(\mathbf{x})$ is a GP,
- $\varepsilon(\mathbf{x})$ is a Gaussian white noise,
- $\{\alpha_i\}_{i=1}^{Q}$ are i.i.d. r.v. with zero mean and variance $\frac{\sigma_0^2}{Q}$,
- $\left\{\psi_i(\mathbf{x})\right\}_{i=1}^{Q}$ is a set of functions

Thus the covariance function of $y(\mathbf{x})$ has the form:

$$\overline{K}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + \sigma_0^2 \boldsymbol{\psi}^\top(\mathbf{x}) \boldsymbol{\psi}(\mathbf{x}') + \sigma^2 \delta(\mathbf{x}, \mathbf{x}'),$$

where $\boldsymbol{\psi}(\mathbf{x}) = \{\psi_i(\mathbf{x}), i = 1, \ldots, Q\}$

Let us assume that

$$y(\mathbf{x}) = \sum_{i=1}^{Q} \alpha_i \psi_i(\mathbf{x}) + f(\mathbf{x}) + \varepsilon(\mathbf{x}),$$

where

- $f(\mathbf{x})$ is a GP,
- $\varepsilon(\mathbf{x})$ is a Gaussian white noise,
- $\{\alpha_i\}_{i=1}^{Q}$ are i.i.d. r.v. with zero mean and variance $\frac{\sigma_0^2}{Q}$,
- $\left\{\psi_i(\mathbf{x})\right\}_{i=1}^{Q}$ is a set of functions

Thus the covariance function of $y(\mathbf{x})$ has the form:

$$\overline{K}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') + \sigma_0^2 \boldsymbol{\psi}^\top(\mathbf{x}) \boldsymbol{\psi}(\mathbf{x}') + \sigma^2 \delta(\mathbf{x}, \mathbf{x}'),$$

where $\boldsymbol{\psi}(\mathbf{x}) = \{\psi_i(\mathbf{x}), i = 1, \ldots, Q\}$

Posterior parameters

Posterior mean of $y(\mathbf{x})$ has the form:

$$\mu_*(\mathbf{x}) = \overline{\mathbf{k}}(\mathbf{x})\overline{\mathbf{K}}^{-1}\mathbf{y},$$

where

- $\Psi = \{\boldsymbol{\psi}(\mathbf{x}_i), i = 1, \ldots, m\}$,
- $\overline{\mathbf{k}}(\mathbf{x}) = \mathbf{k}(\mathbf{x}) + \sigma_0^2\boldsymbol{\psi}^\top(\mathbf{x})\Psi$,
- $\overline{\mathbf{K}} = \mathbf{K} + \sigma_0^2\Psi^\top\Psi + \sigma^2\mathrm{I}_m$

Posterior variance has the form:

$$\sigma_*^2(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}) + \sigma^2 + \sigma_0^2\boldsymbol{\psi}^\top(\mathbf{x})\boldsymbol{\psi}(\mathbf{x}) - \overline{\mathbf{k}}^\top(\mathbf{x})\overline{\mathbf{K}}^{-1}\overline{\mathbf{k}}(\mathbf{x})$$

Posterior mean of $y(\mathbf{x})$ has the form:

$$\mu_*(\mathbf{x}) = \overline{\mathbf{k}}(\mathbf{x})\overline{\mathbf{K}}^{-1}\mathbf{y},$$

where

- $\Psi = \{\boldsymbol{\psi}(\mathbf{x}_i), i = 1, \ldots, m\}$,
- $\overline{\mathbf{k}}(\mathbf{x}) = \mathbf{k}(\mathbf{x}) + \sigma_0^2\boldsymbol{\psi}^\top(\mathbf{x})\Psi$,
- $\overline{\mathbf{K}} = \mathbf{K} + \sigma_0^2\Psi^\top\Psi + \sigma^2\mathrm{I}_m$

Posterior variance has the form:

$$\sigma_*^2(\mathbf{x}) = K(\mathbf{x},\mathbf{x}) + \sigma^2 + \sigma_0^2\boldsymbol{\psi}^\top(\mathbf{x})\boldsymbol{\psi}(\mathbf{x}) - \overline{\mathbf{k}}^\top(\mathbf{x})\overline{\mathbf{K}}^{-1}\overline{\mathbf{k}}(\mathbf{x})$$

Posterior parameters

Posterior mean of $y(\mathbf{x})$ has the form:

$$\mu_*(\mathbf{x}) = \overline{\mathbf{k}}(\mathbf{x})\overline{\mathbf{K}}^{-1}\mathbf{y},$$

where

- $\Psi = \{\boldsymbol{\psi}(\mathbf{x}_i), i = 1, \ldots, m\}$,
- $\overline{\mathbf{k}}(\mathbf{x}) = \mathbf{k}(\mathbf{x}) + \sigma_0^2\boldsymbol{\psi}^\top(\mathbf{x})\Psi$,
- $\overline{\mathbf{K}} = \mathbf{K} + \sigma_0^2\Psi^\top\Psi + \sigma^2\mathrm{I}_m$

Posterior variance has the form:

$$\sigma_*^2(\mathbf{x}) = K(\mathbf{x},\mathbf{x}) + \sigma^2 + \sigma_0^2\boldsymbol{\psi}^\top(\mathbf{x})\boldsymbol{\psi}(\mathbf{x}) - \overline{\mathbf{k}}^\top(\mathbf{x})\overline{\mathbf{K}}^{-1}\overline{\mathbf{k}}(\mathbf{x})$$

Posterior mean of $y(\mathbf{x})$ has the form:

$$\mu_*(\mathbf{x}) = \overline{\mathbf{k}}(\mathbf{x})\overline{\mathbf{K}}^{-1}\mathbf{y},$$

where

- $\Psi = \{\boldsymbol{\psi}(\mathbf{x}_i), i = 1, \ldots, m\}$,
- $\overline{\mathbf{k}}(\mathbf{x}) = \mathbf{k}(\mathbf{x}) + \sigma_0^2\boldsymbol{\psi}^\top(\mathbf{x})\Psi$,
- $\overline{\mathbf{K}} = \mathbf{K} + \sigma_0^2\Psi^\top\Psi + \sigma^2\mathrm{I}_m$

Posterior variance has the form:

$$\sigma_*^2(\mathbf{x}) = K(\mathbf{x},\mathbf{x}) + \sigma^2 + \sigma_0^2\boldsymbol{\psi}^\top(\mathbf{x})\boldsymbol{\psi}(\mathbf{x}) - \overline{\mathbf{k}}^\top(\mathbf{x})\overline{\mathbf{K}}^{-1}\overline{\mathbf{k}}(\mathbf{x})$$

Posterior mean of $y(\mathbf{x})$ has the form:

$$\mu_*(\mathbf{x}) = \overline{\mathbf{k}}(\mathbf{x})\overline{\mathbf{K}}^{-1}\mathbf{y},$$

where

- $\Psi = \{\boldsymbol{\psi}(\mathbf{x}_i), i = 1, \ldots, m\}$,
- $\overline{\mathbf{k}}(\mathbf{x}) = \mathbf{k}(\mathbf{x}) + \sigma_0^2\boldsymbol{\psi}^\top(\mathbf{x})\Psi$,
- $\overline{\mathbf{K}} = \mathbf{K} + \sigma_0^2\Psi^\top\Psi + \sigma^2\mathrm{I}_m$

Posterior variance has the form:

$$\sigma_*^2(\mathbf{x}) = K(\mathbf{x},\mathbf{x}) + \sigma^2 + \sigma_0^2\boldsymbol{\psi}^\top(\mathbf{x})\boldsymbol{\psi}(\mathbf{x}) - \overline{\mathbf{k}}^\top(\mathbf{x})\overline{\mathbf{K}}^{-1}\overline{\mathbf{k}}(\mathbf{x})$$

Minus log-likelihood has a similar structure compared to a stationary process, but we should replace $\mathbf{K}$ by $\overline{\mathbf{K}}$:

$$-\log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^\top \overline{\mathbf{K}}^{-1}\mathbf{y} + \frac{1}{2}\log|\overline{\mathbf{K}}| - \frac{m}{2}\log 2\pi,$$

where $\{\boldsymbol{\theta}, \sigma_0, \sigma\}$ are parameters

If the set of functions is fixed, then estimates of parameters can be obtained via usual maximization of the log-likelihood

$$\log p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

Minus log-likelihood has a similar structure compared to a stationary process, but we should replace $\mathbf{K}$ by $\overline{\mathbf{K}}$:

$$-\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^\top \overline{\mathbf{K}}^{-1}\mathbf{y} + \frac{1}{2}\log|\overline{\mathbf{K}}| - \frac{m}{2}\log 2\pi,$$

where $\{\boldsymbol{\theta}, \sigma_0, \sigma\}$ are parameters

If the set of functions is fixed, then estimates of parameters can be obtained via usual maximization of the log-likelihood

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

Minus log-likelihood has a similar structure compared to a stationary process, but we should replace $\mathbf{K}$ by $\overline{\mathbf{K}}$:

$$-\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^\top \overline{\mathbf{K}}^{-1}\mathbf{y} + \frac{1}{2}\log|\overline{\mathbf{K}}| - \frac{m}{2}\log 2\pi,$$

where $\{\boldsymbol{\theta}, \sigma_0, \sigma\}$ are parameters

If the set of functions is fixed, then estimates of parameters can be obtained via usual maximization of the log-likelihood

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

Minus log-likelihood has a similar structure compared to a stationary process, but we should replace $\mathbf{K}$ by $\overline{\mathbf{K}}$:

$$-\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^\top \overline{\mathbf{K}}^{-1}\mathbf{y} + \frac{1}{2}\log|\overline{\mathbf{K}}| - \frac{m}{2}\log 2\pi,$$

where $\{\boldsymbol{\theta}, \sigma_0, \sigma\}$ are parameters

If the set of functions is fixed, then estimates of parameters can be obtained via usual maximization of the log-likelihood

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \to \max_{\boldsymbol{\theta}}$$

Let us consider three families of functions:

1. Sigmoid functions

$$\psi_j(\mathbf{x}) = \sigma\left(\sum_{i=1}^{d} \beta_{j,i} x_i\right),$$

where $\sigma(x) = \frac{e^x - 1}{e^x + 1}$, $\beta_{j,i} \in \mathbb{R}$ are parameters

2. Radial basis functions (RBF)

$$\psi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|_2^2}{r_j^2}\right),$$

where $\mathbf{c}_j \in \mathbb{R}^d, r_j \in \mathbb{R}$ are parameters of the function.

3. Linear basis functions

$$\psi_j(\mathbf{x}) = x_j, j = 1, 2, \ldots, d$$

Let us consider three families of functions:

1. Sigmoid functions

$$\psi_j(\mathbf{x}) = \sigma\left(\sum_{i=1}^{d} \beta_{j,i} x_i\right),$$

where $\sigma(x) = \frac{e^x - 1}{e^x + 1}$, $\beta_{j,i} \in \mathbb{R}$ are parameters

2. Radial basis functions (RBF)

$$\psi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|_2^2}{r_j^2}\right),$$

where $\mathbf{c}_j \in \mathbb{R}^d, r_j \in \mathbb{R}$ are parameters of the function.

3. Linear basis functions

$$\psi_j(\mathbf{x}) = x_j, j = 1, 2, \ldots, d$$

Let us consider three families of functions:

1. Sigmoid functions

$$\psi_j(\mathbf{x}) = \sigma\left(\sum_{i=1}^{d} \beta_{j,i} x_i\right),$$

where $\sigma(x) = \frac{e^x - 1}{e^x + 1}$, $\beta_{j,i} \in \mathbb{R}$ are parameters

2. Radial basis functions (RBF)

$$\psi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|_2^2}{r_j^2}\right),$$

where $\mathbf{c}_j \in \mathbb{R}^d, r_j \in \mathbb{R}$ are parameters of the function.

3. Linear basis functions

$$\psi_j(\mathbf{x}) = x_j, j = 1, 2, \ldots, d$$

Approximation of Heaviside function for $x \in [0, 1]$, training sample size $m = 20$