Introduction to Gaussian Processes for Machine Learning

Nikola Popović

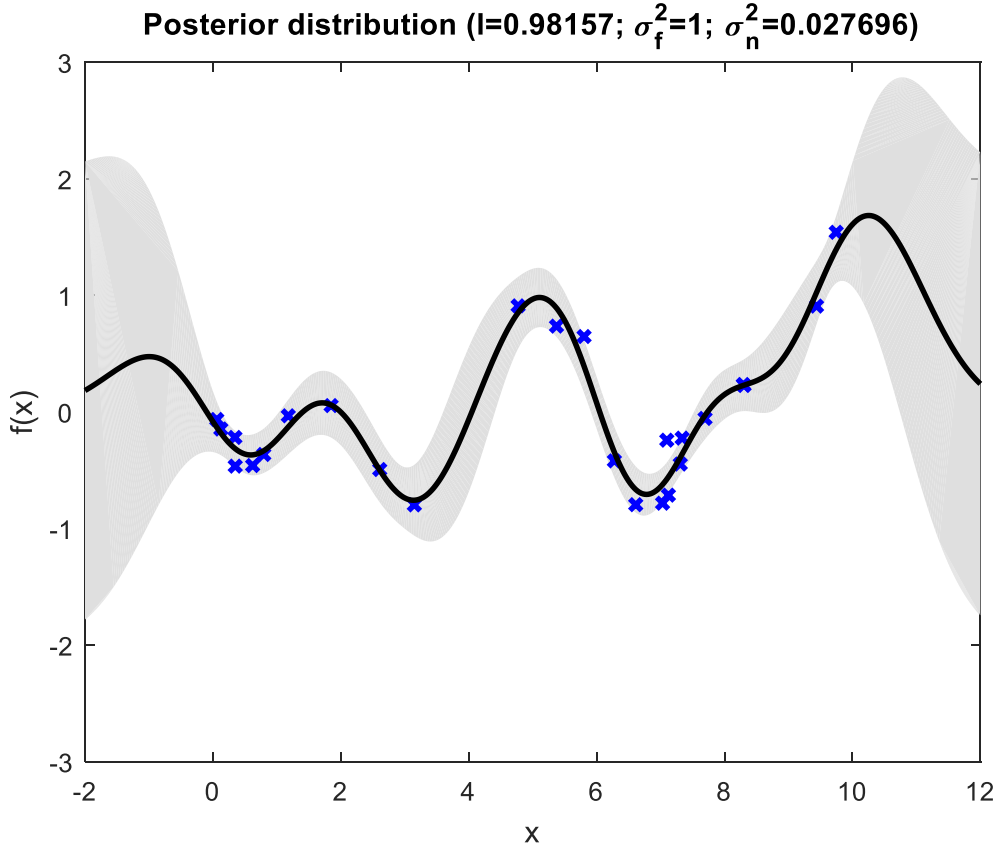
npopovic3794@gmail.com

Bayesian Inference

- It can be applied to classical ML models like linear/logistic regression, Deep NN, etc...
- Model parameters are now probability distributions, not regular numbers
- We can naturally get a predictive uncertainty and the uncertainty in the model

Bayesian Inference

- An example of a predictive distribution for a 1D input space



Linear Regression

- Classical linear regression

$$\mathrm{y} = f(\mathbf{x}) + \epsilon$$
$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$$
$$\epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

- ϵ is white noise (independent and indentically distributed)

- Likelihood

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{n} p(\mathrm{y_i}|\mathbf{x_i}, \mathbf{w}) = \cdots = \mathcal{N}(\mathbf{X}^{\mathrm{T}}\mathbf{w}, \sigma_n^2\mathbf{I})$$

Bayesian Linear Regression

- We specify the prior beleifs about the model

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{p}})$$

- We obtain the posterior parameter distribution using the dataset

$$\boxed{p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}}$$

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})\mathrm{d}\mathbf{w}$$

...

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \sim \mathcal{N}\left(\frac{1}{\sigma_n^2}\boldsymbol{\Sigma}_{\mathbf{w}}\mathbf{X}\mathbf{y}, \boldsymbol{\Sigma}_{\mathbf{w}}\right)$$

$$\boldsymbol{\Sigma}_{\mathbf{w}} = \left(\frac{1}{\sigma_n^2}\mathbf{X}\mathbf{X}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\boldsymbol{p}}^{-1}\right)^{-1}$$

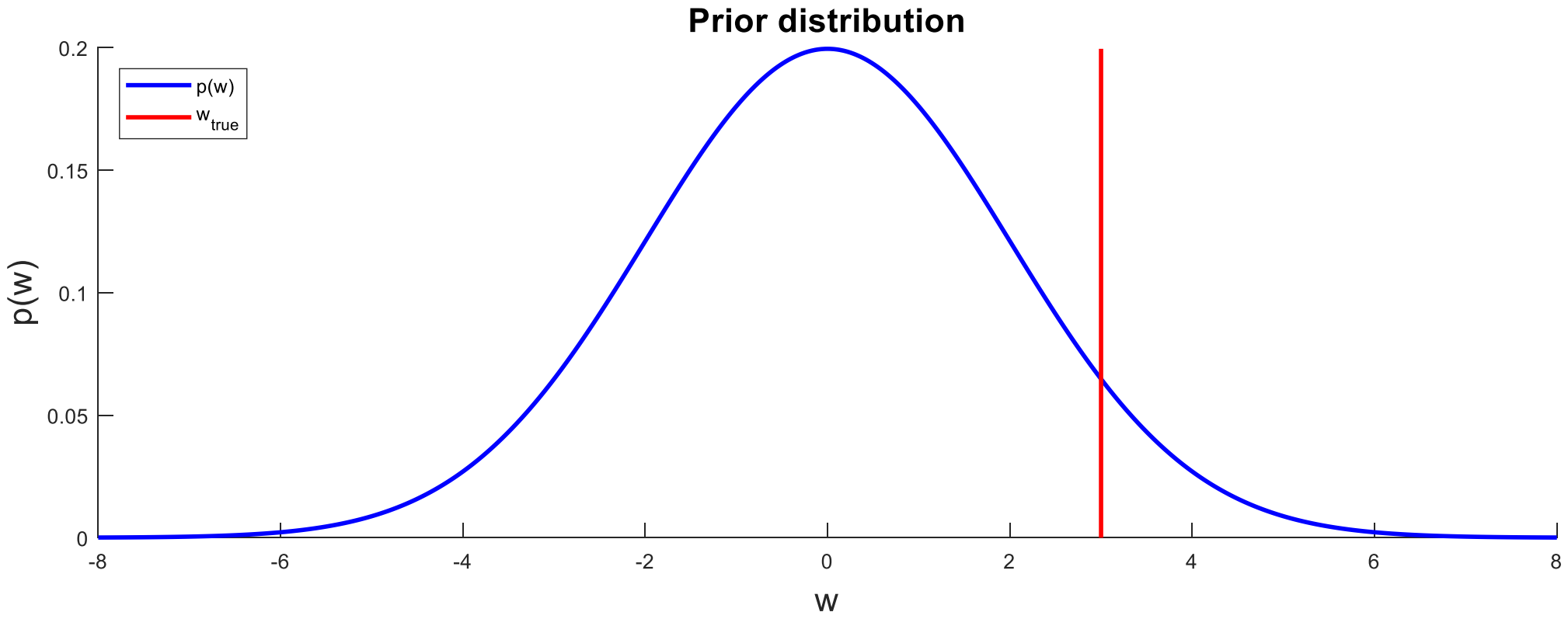
Bayesian Linear Regression

- Example with a 1D input and no intercept term

$$\mathrm{y} = wx + \epsilon$$
$$\epsilon \sim \mathcal{N}(0, \sigma_n^2)$$
$$w \sim \mathcal{N}(0, \sigma_p^2)$$

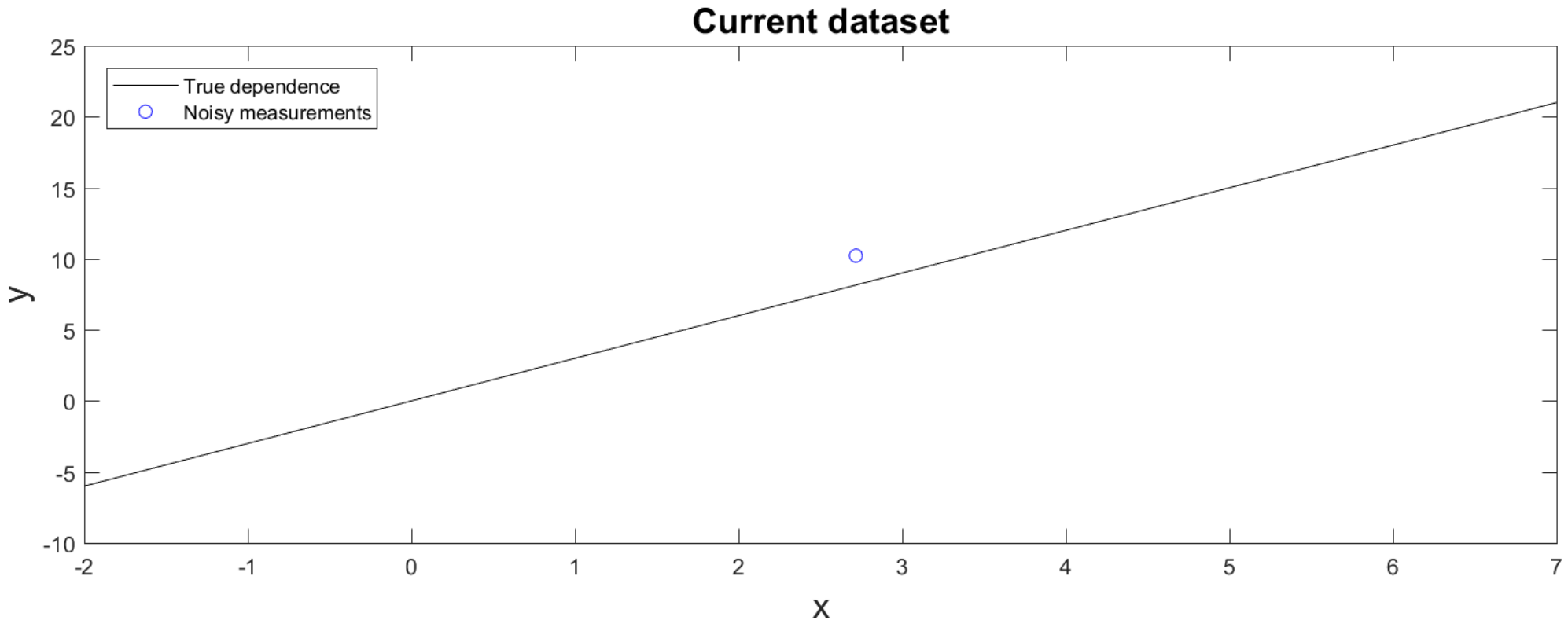
Bayesian Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$



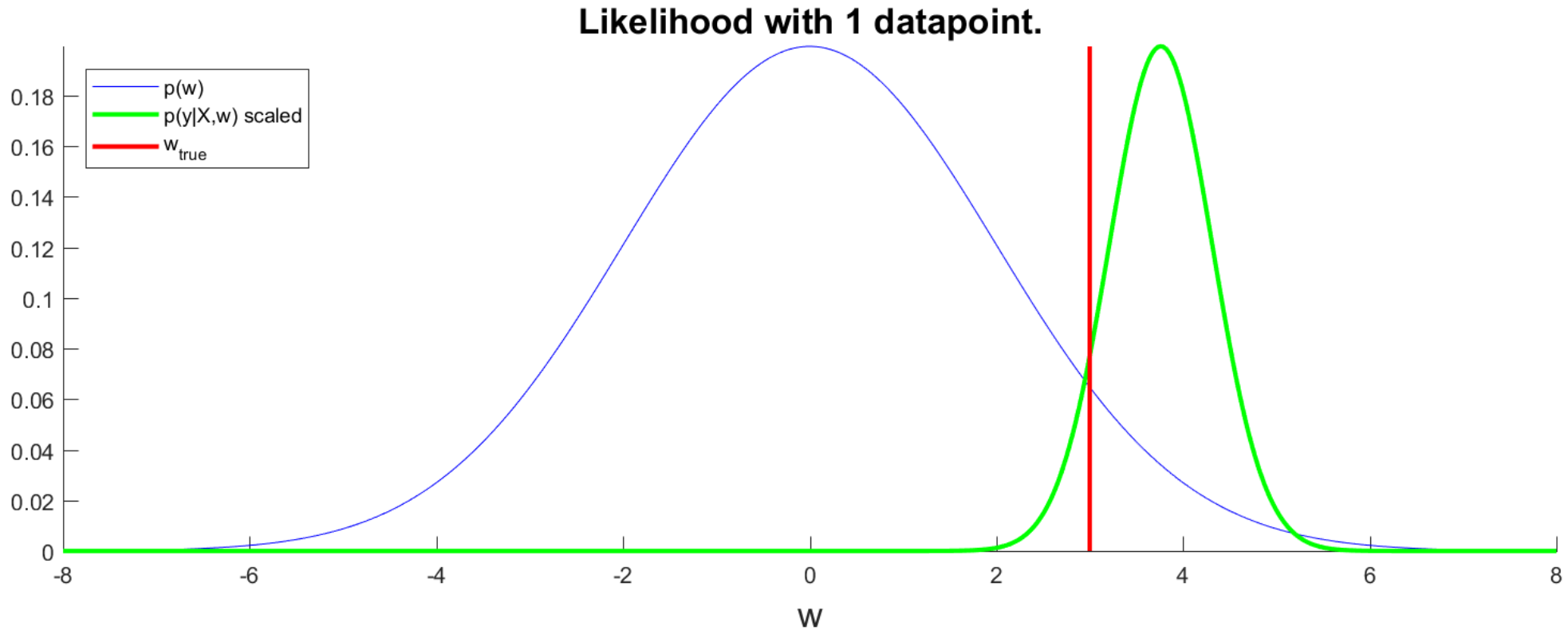
Bayesian Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$



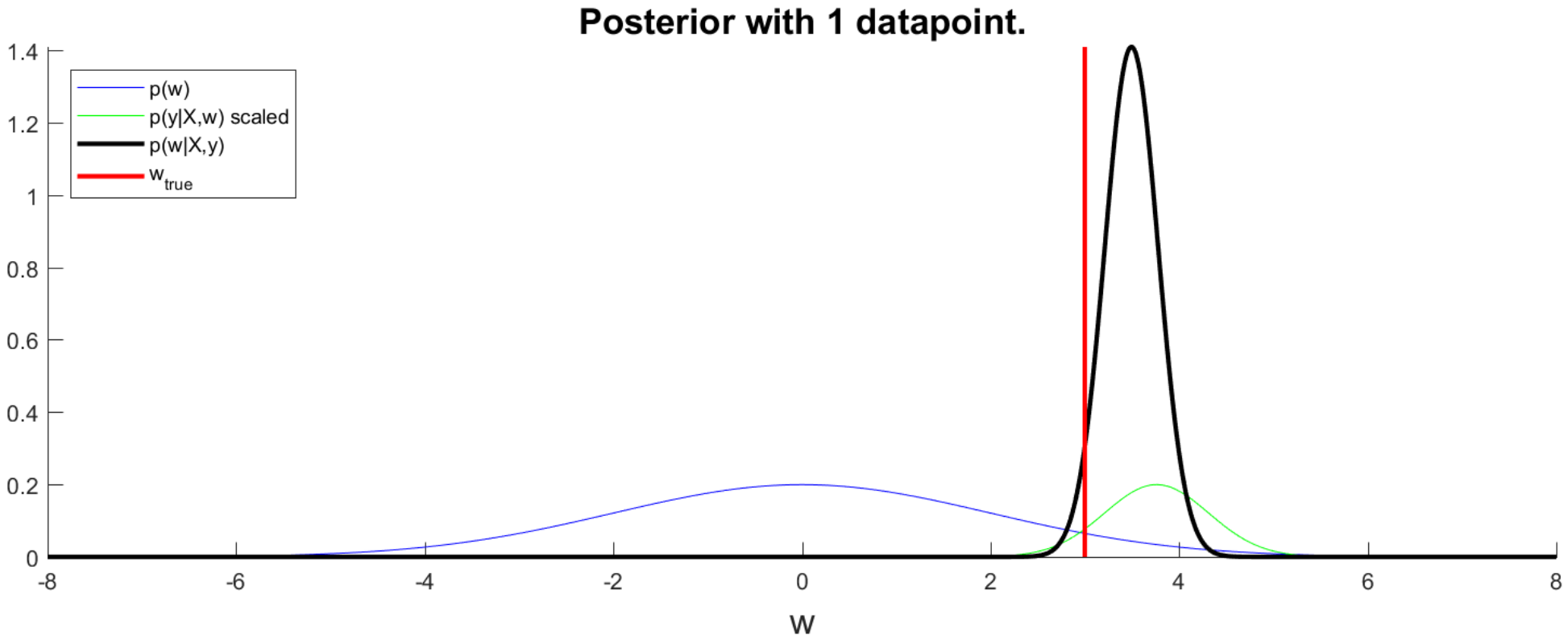
Bayesian Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$



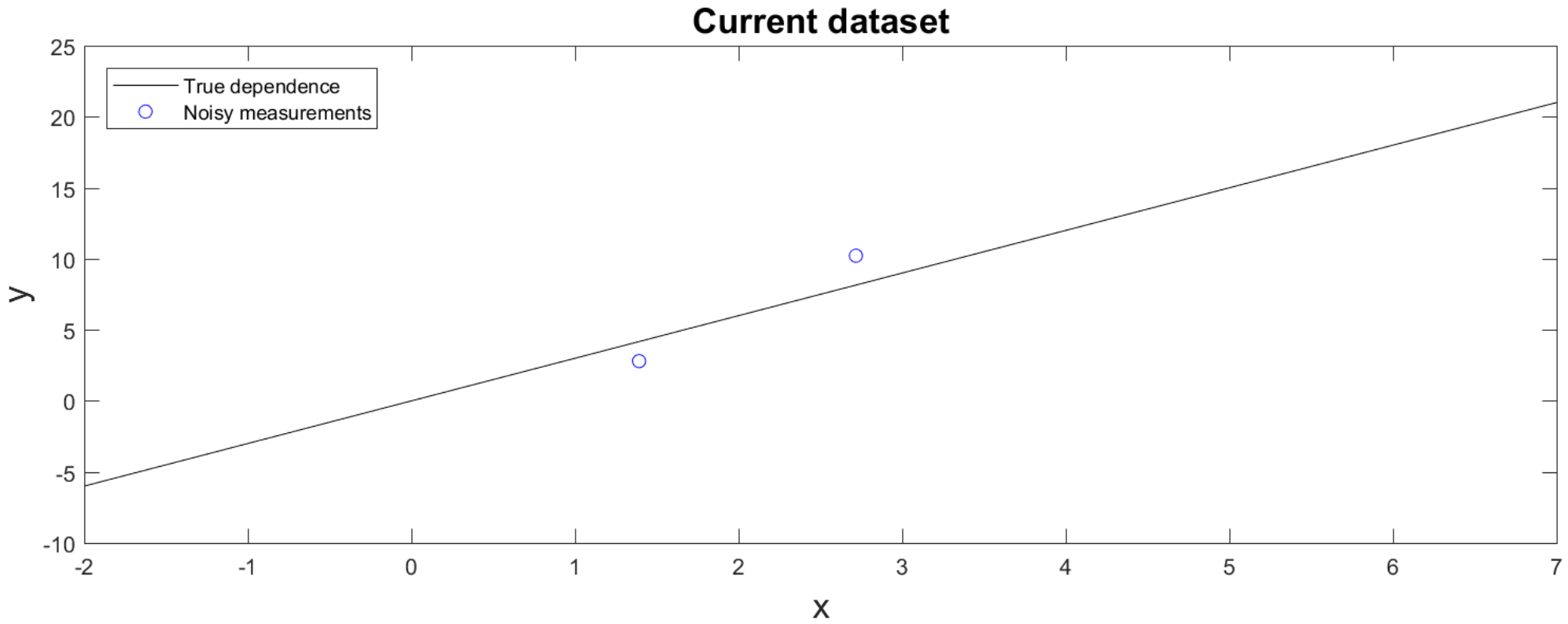
Bayesian Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$



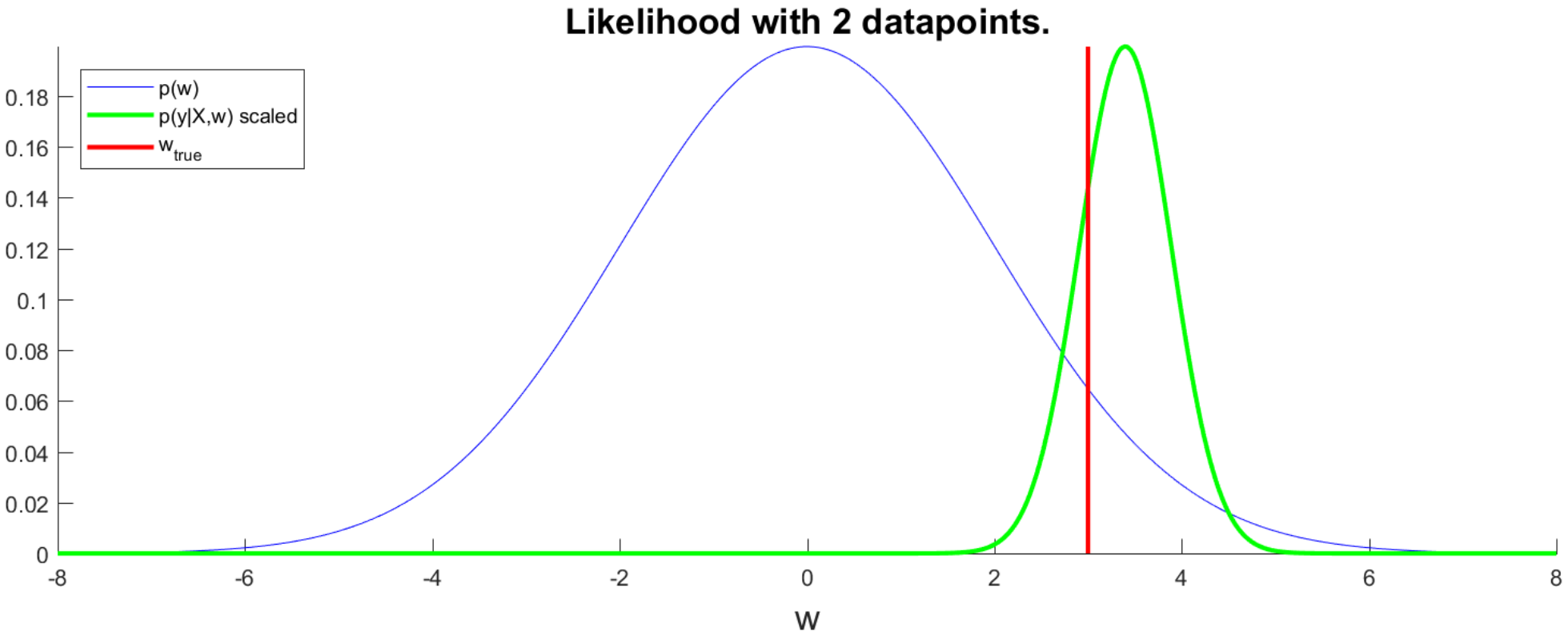
Bayesian Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$



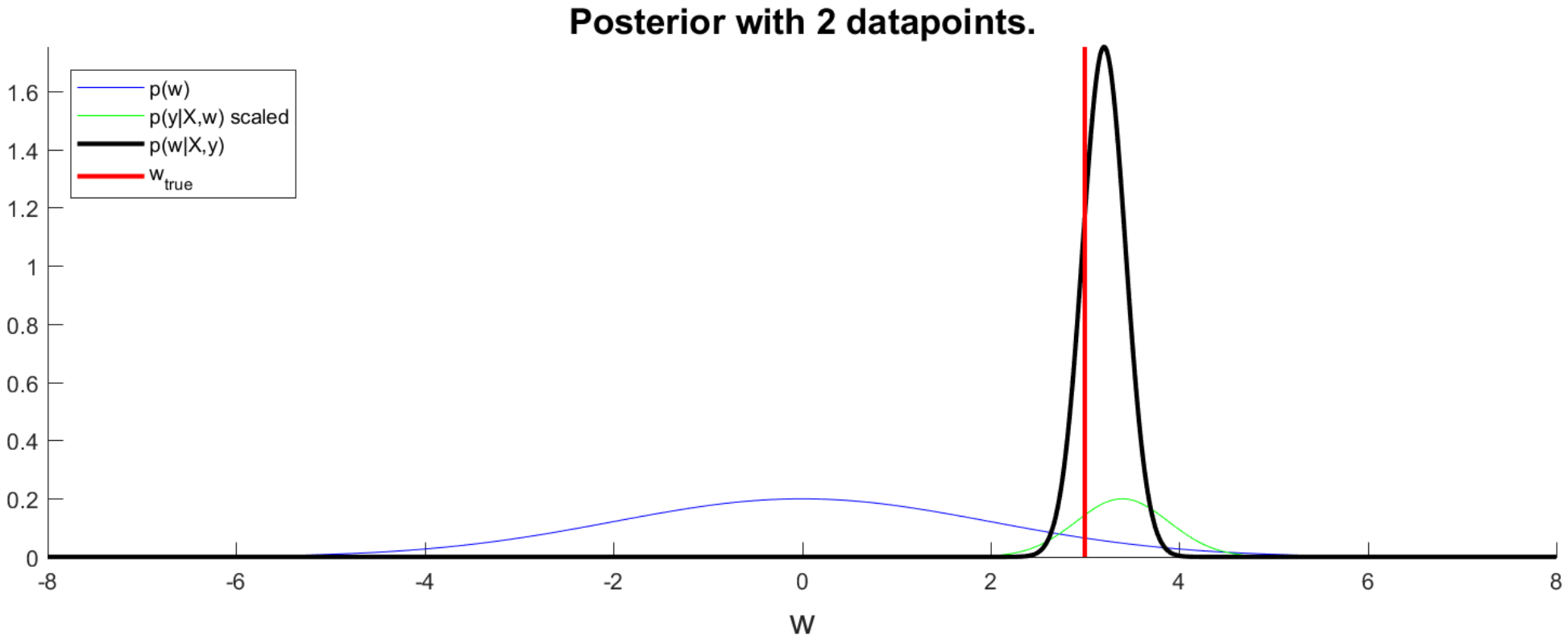
Bayesian Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$



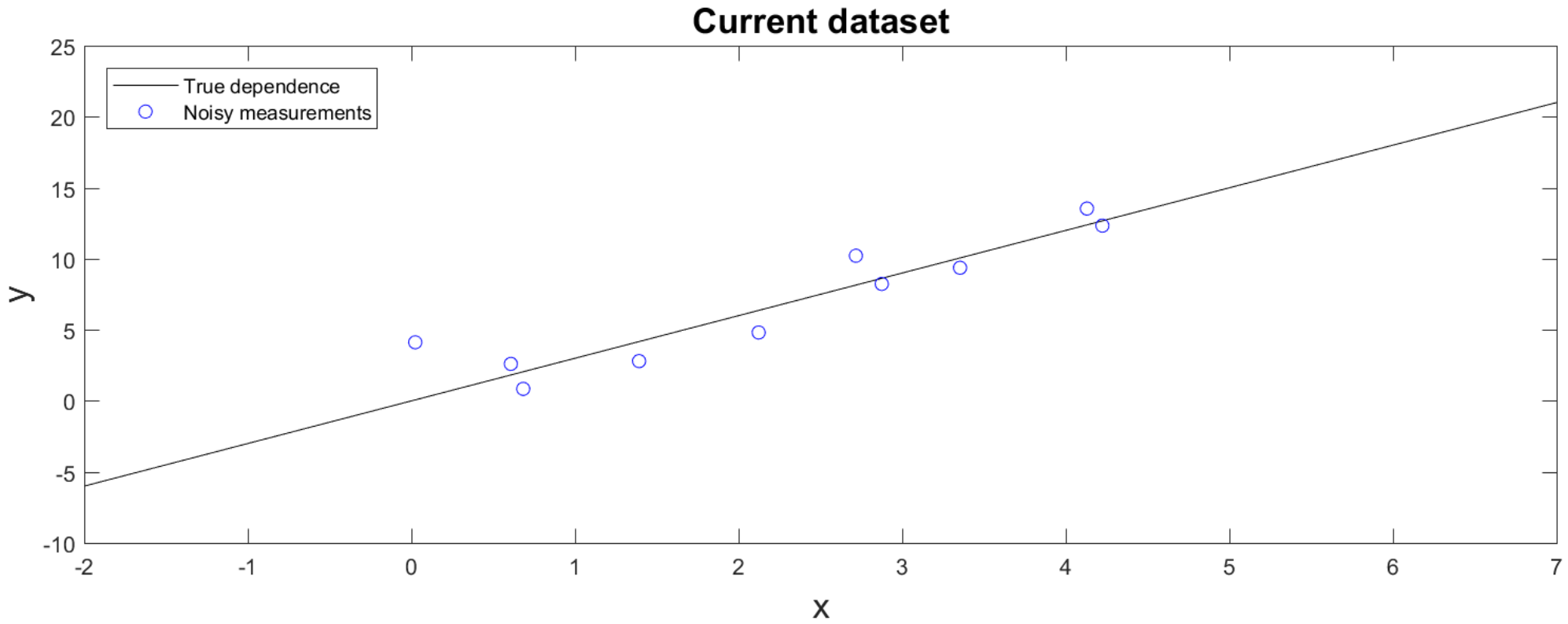
Bayesian Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$



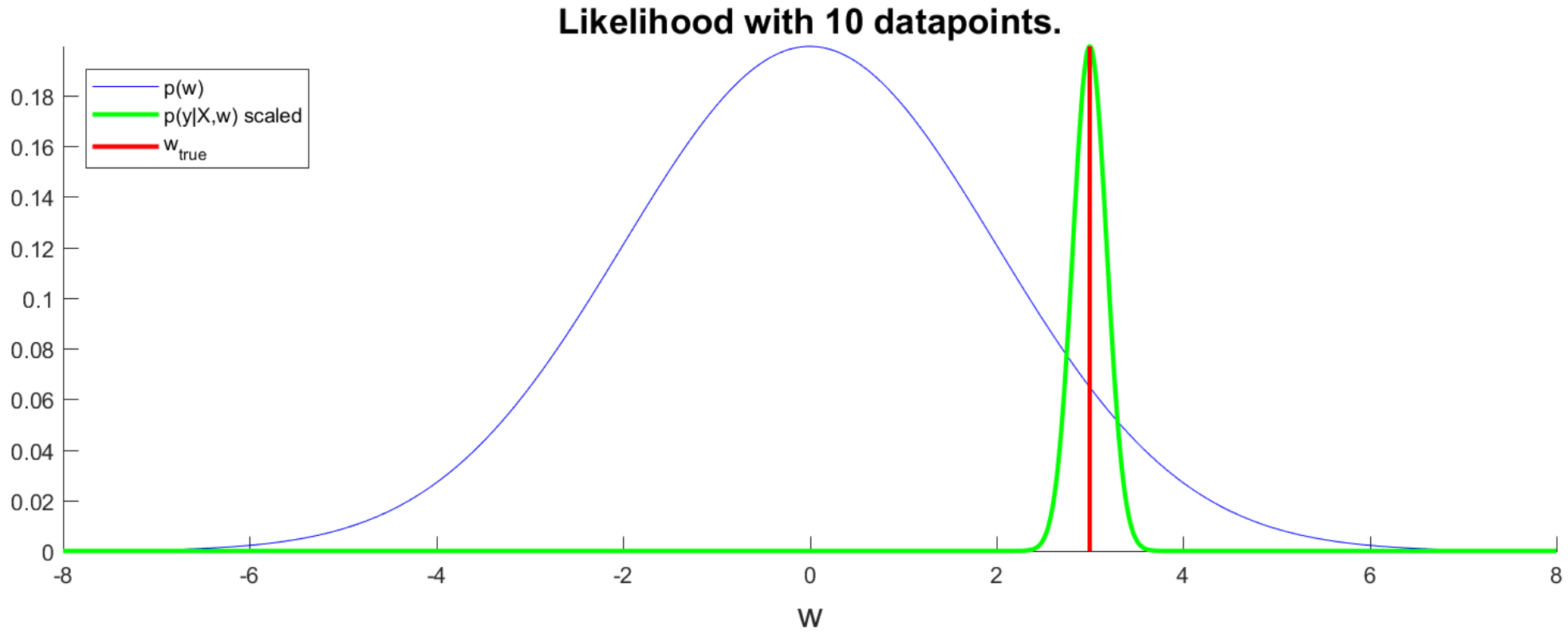
Bayesian Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$



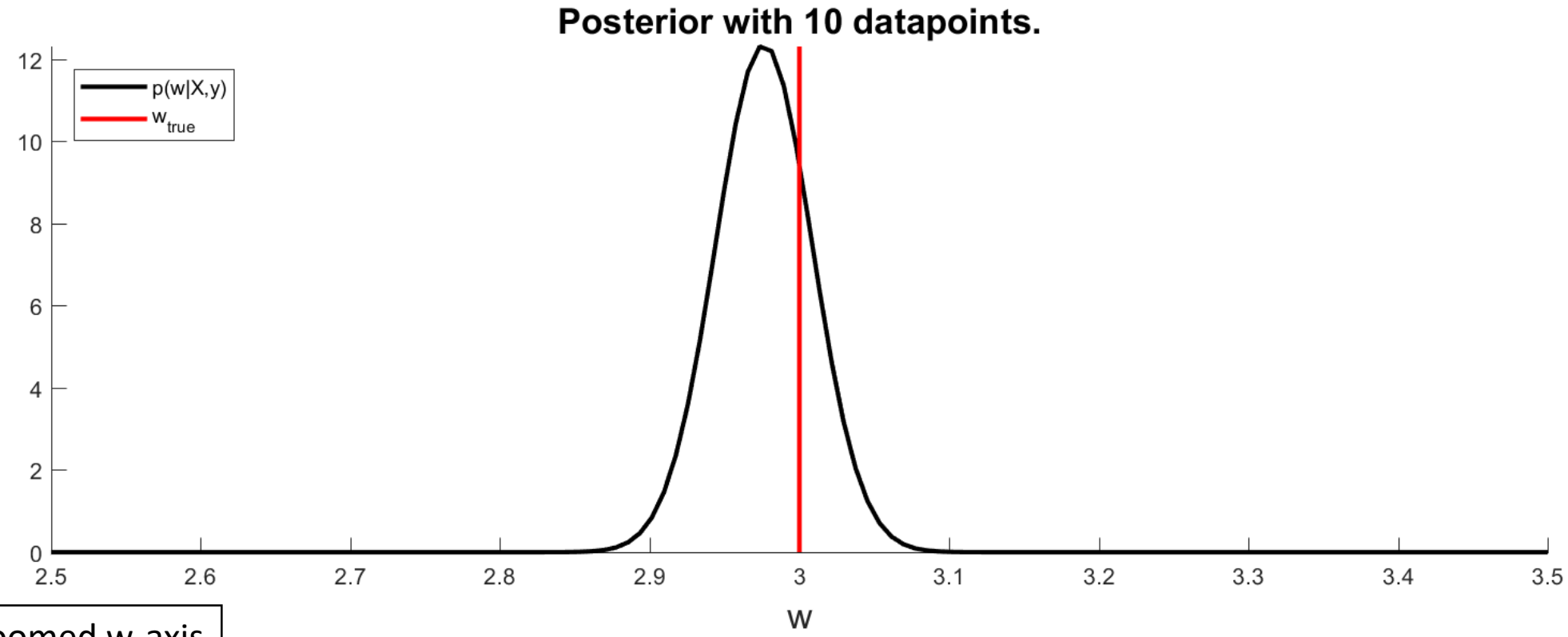
Bayesian Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$



Bayesian Linear Regression

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$



Zoomed w-axis

Bayesian Linear Regression

- We have learned the parameter posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$

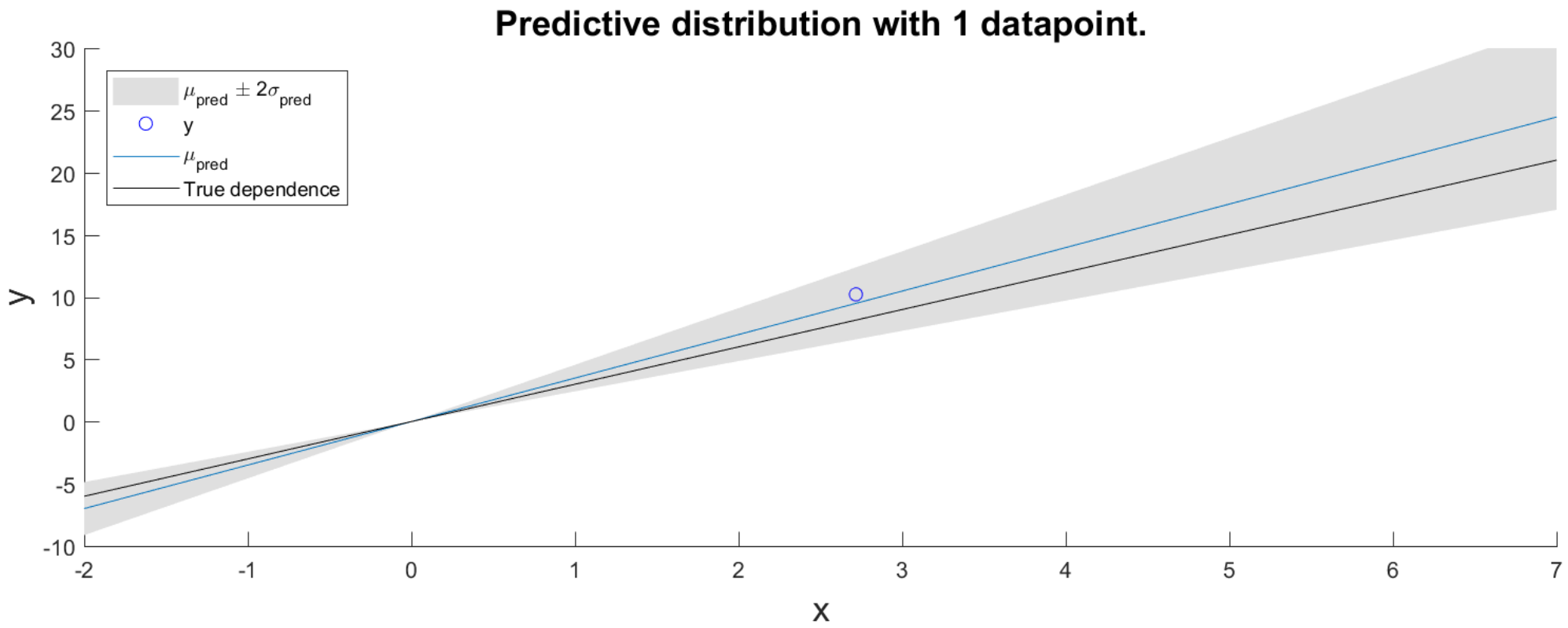
- We have a new input $\mathbf{x}_*$ and we want to make a prediction
- Insted of a point-wise prediction we get a predictive distribution

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int p(f_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \, \mathrm{d}\mathbf{w}$$

$$\sim \mathcal{N}\left(\frac{1}{\sigma_n^2} \mathbf{x}_*^{\mathrm{T}} \mathbf{\Sigma}_{\mathbf{w}} \mathbf{X}\mathbf{y}, \mathbf{x}_*^{\mathrm{T}} \mathbf{\Sigma}_{\mathbf{w}} \mathbf{x}_*\right)$$

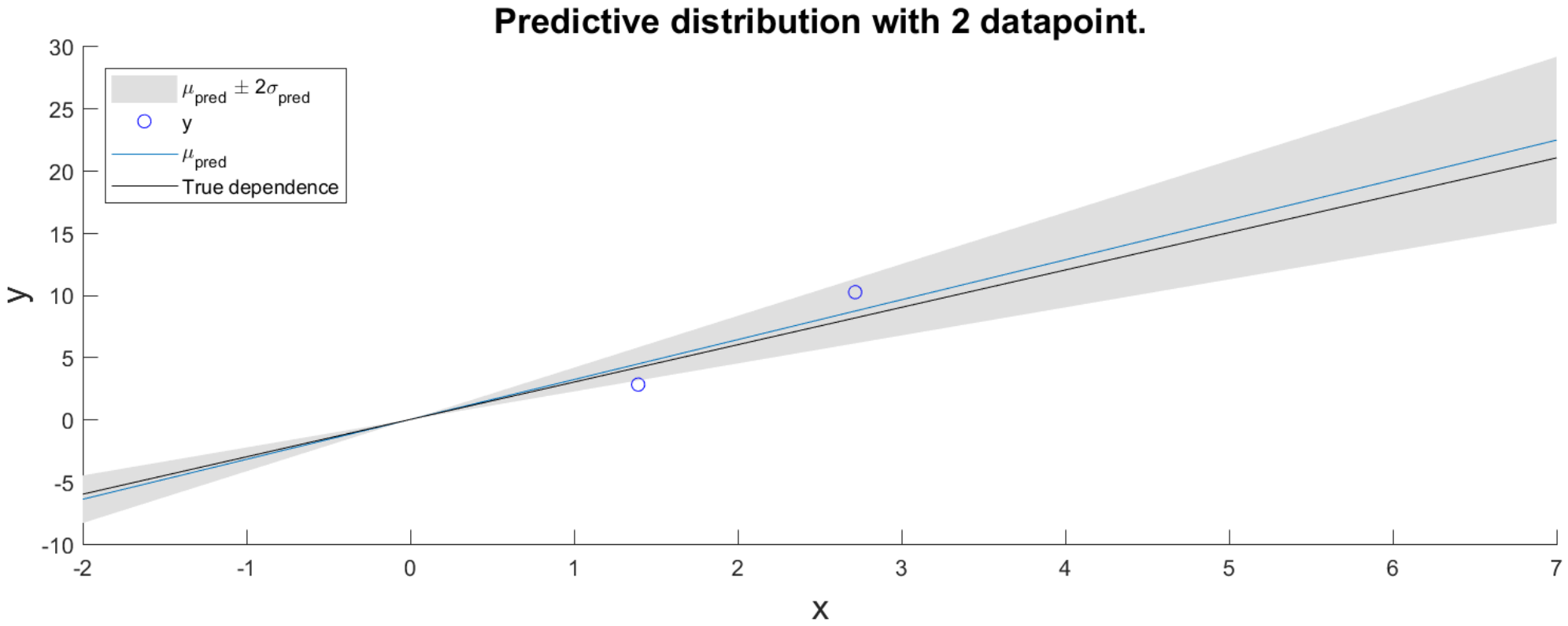
Bayesian Linear Regression

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int p(f_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \, \mathrm{d}\mathbf{w}$$



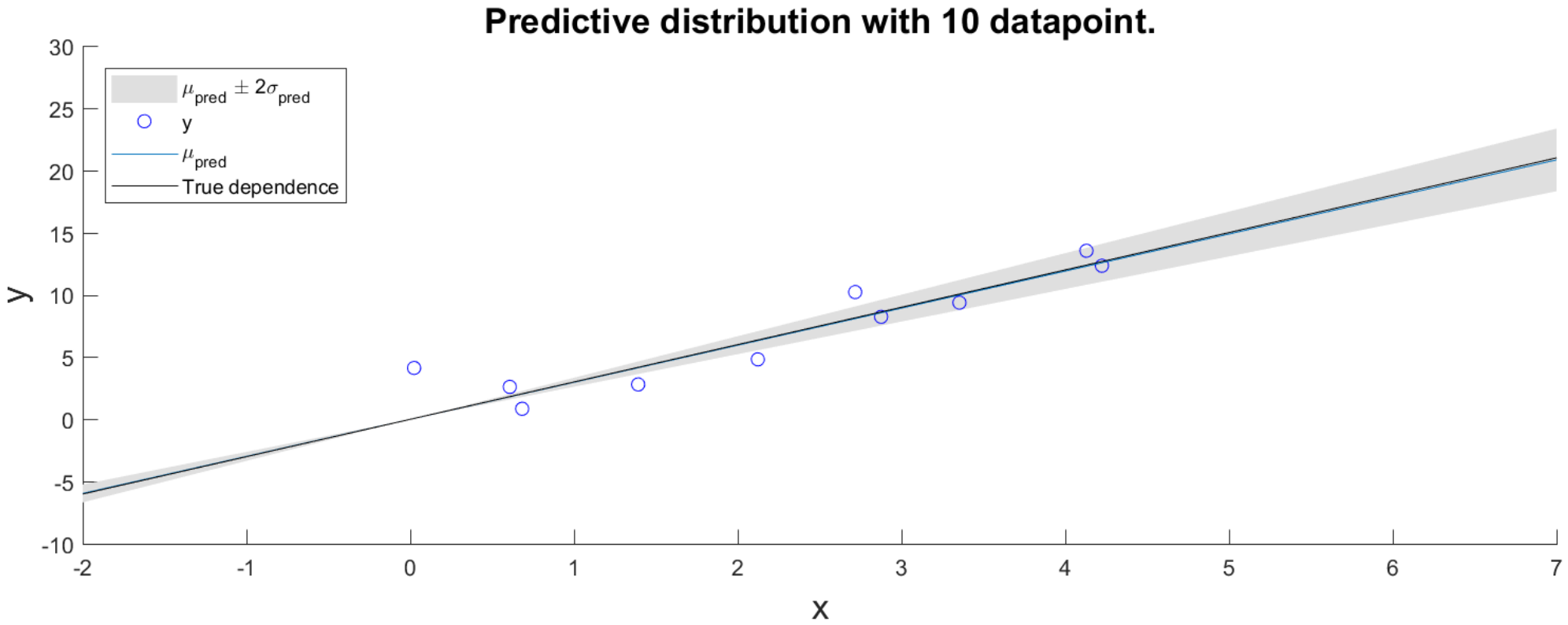
Bayesian Linear Regression

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int p(f_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \, \mathrm{d}\mathbf{w}$$



Bayesian Linear Regression

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int p(f_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \, \mathrm{d}\mathbf{w}$$



Bayesian Linear Regression

- We can use a feature mapping ϕ: $\mathbb{R}^D \mapsto \mathbb{R}^N$ on the input

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}$$

- For example $\phi(x) = [1, x, x^2, x^3, \dots]$
- We just need to replace $\mathbf{x}$ with $\phi(\mathbf{x})$ in the equations

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) \sim \mathcal{N}\left(\frac{1}{\sigma_n^2}\boldsymbol{\phi}_*^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{w}}\boldsymbol{\Phi}\mathbf{y}, \boldsymbol{\phi}_*^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{w}}\boldsymbol{\phi}_*\right)$$

$$\boldsymbol{\Sigma}_{\mathbf{w}} = \left(\frac{1}{\sigma_n^2}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} + \boldsymbol{\Sigma}_{\boldsymbol{p}}^{-1}\right)^{-1}$$

- Computational complexity is dominated with inverting a $N \times N$ matrix, which is not convinient if N is very large

Bayesian Linear Regression

- An alternative form of the predictive distribution

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$$

$$\sim \mathcal{N}\left(\boxed{\boldsymbol{\phi}_*^{\mathrm{T}}\boldsymbol{\Sigma}_p\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}_p\boldsymbol{\Phi}+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}}, \boxed{\boldsymbol{\phi}_*^{\mathrm{T}}\boldsymbol{\Sigma}_p\boldsymbol{\phi}_* - \boldsymbol{\phi}_*^{\mathrm{T}}\boldsymbol{\Sigma}_p\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}_p\boldsymbol{\Phi}+\sigma_n^2\mathbf{I}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Sigma}_p\boldsymbol{\phi}_*}\right)$$

- The features are present in equations only in the following form

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}}\boldsymbol{\Sigma}_p\phi(\mathbf{x}')$$

- We can use the kernel trick

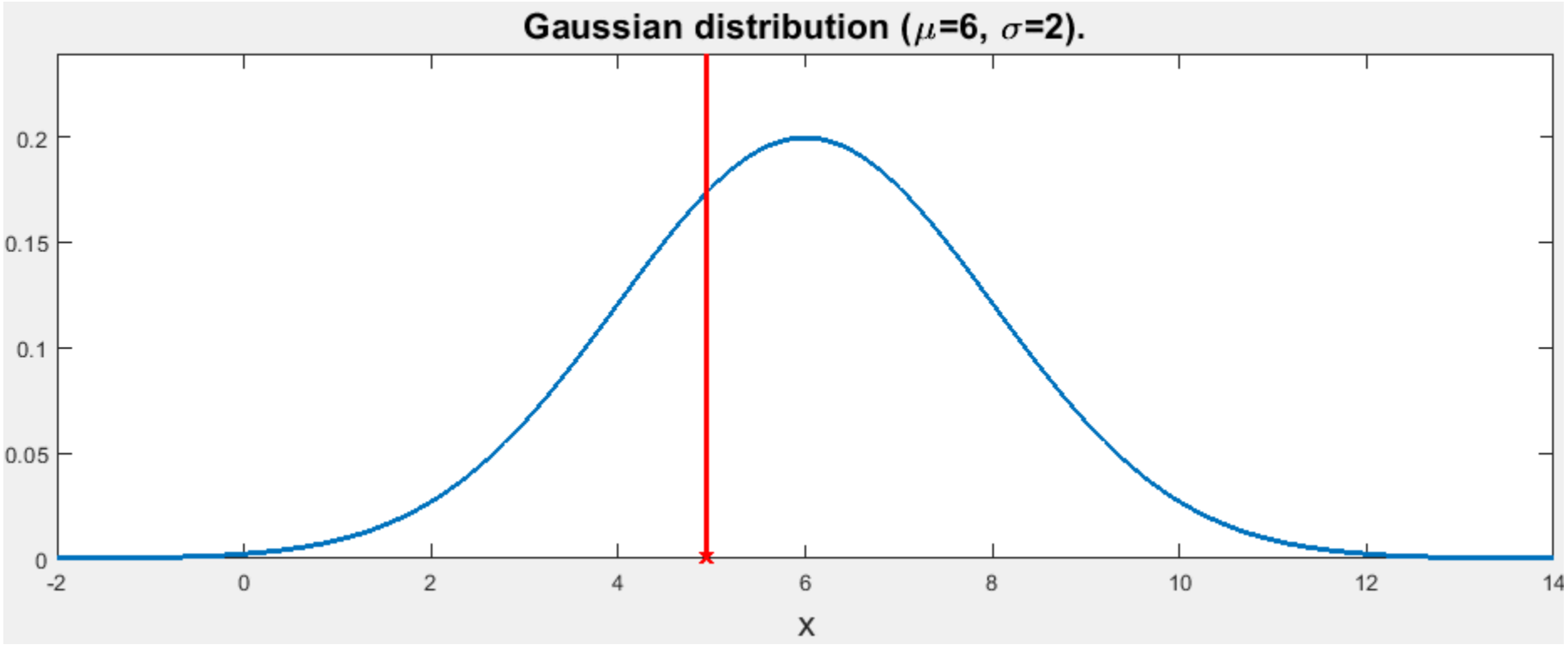
$$\psi(\mathbf{x}) = \boldsymbol{\Sigma}_p^{1/2}\phi(\mathbf{x}) \Longrightarrow k(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x}) \cdot \psi(\mathbf{x}')$$

- We dont need to calculate $\phi(\mathbf{x})$ explicitly, just the kernel $k(\mathbf{x}, \mathbf{x}')$

Gaussian distribution

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

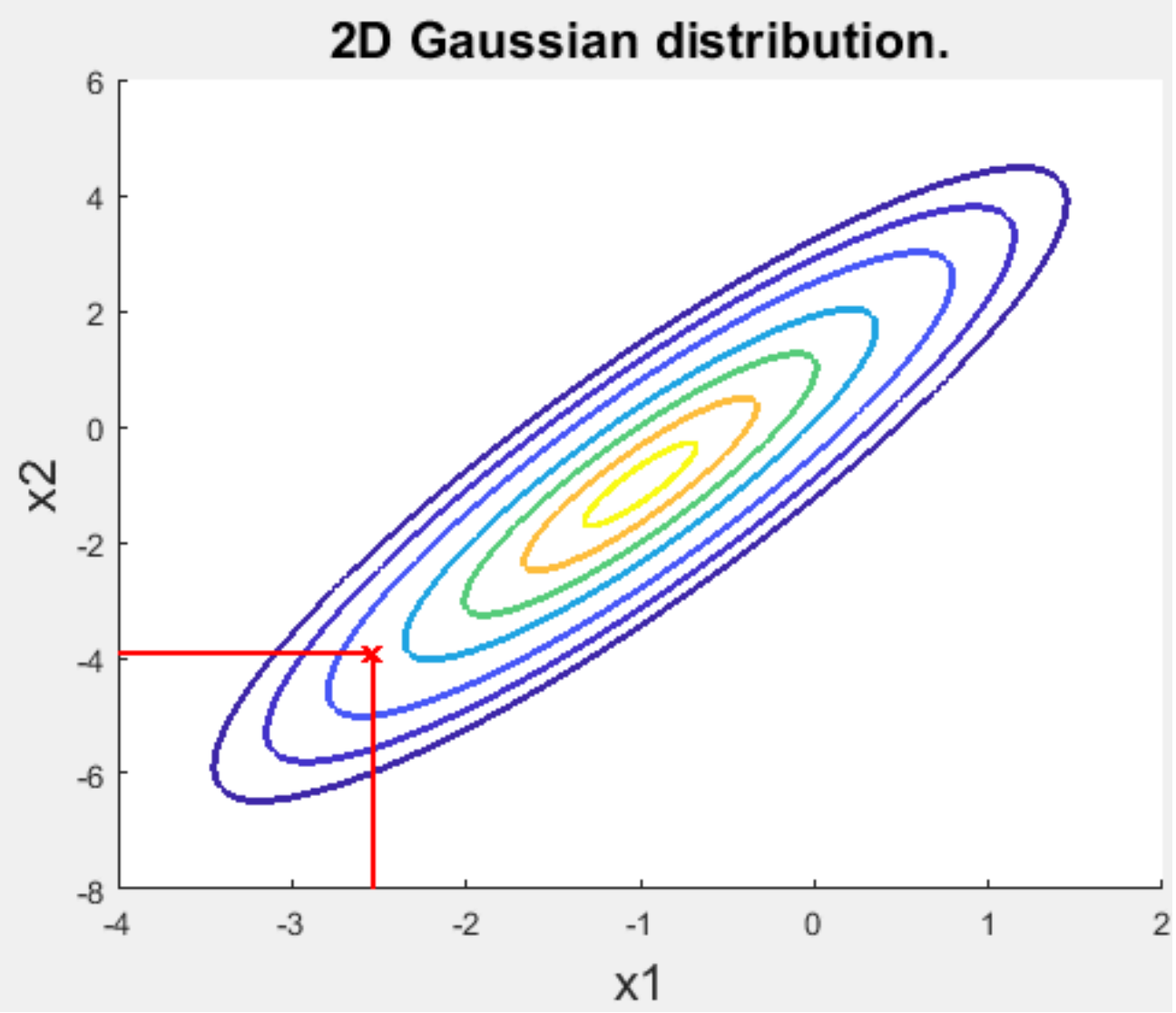
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Gaussian distribution

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$\mathbf{x} = [x_1\ x_2]^{\mathrm{T}}$$
$$\boldsymbol{\mu} = [-1\ \ -1]^{\mathrm{T}}$$
$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix}$$

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$



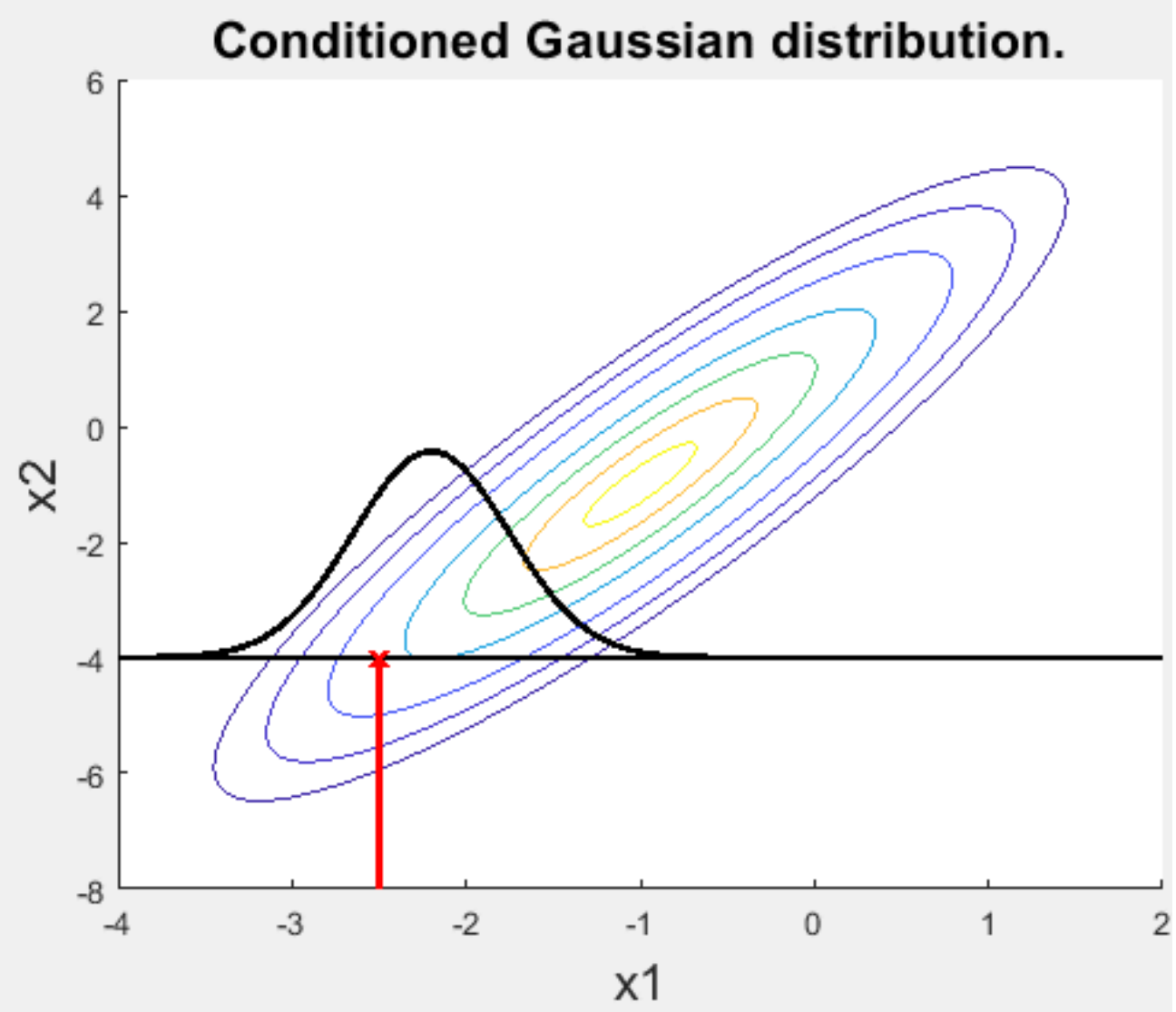
Gaussian distribution

$$x_1|x_2 = -4 \sim \mathcal{N}(\bar{\mu}, \bar{\sigma}^2)$$

$$\bar{\mu} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2)$$

$$\bar{\sigma}^2 = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

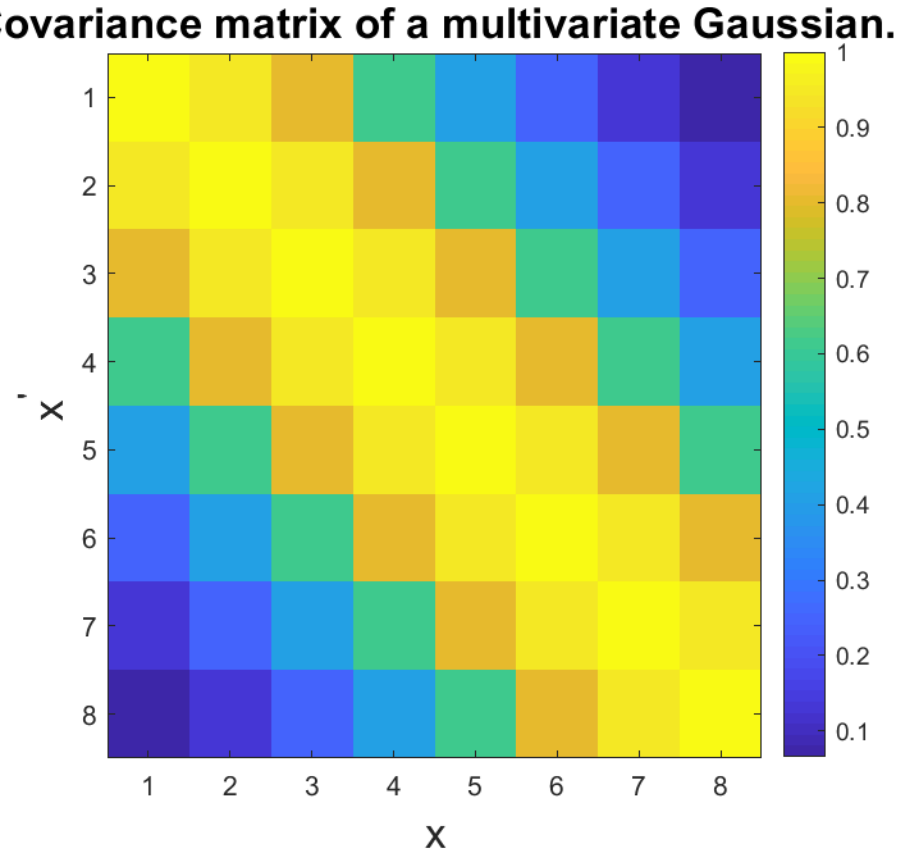
- This is one of the main ideas behind GPs

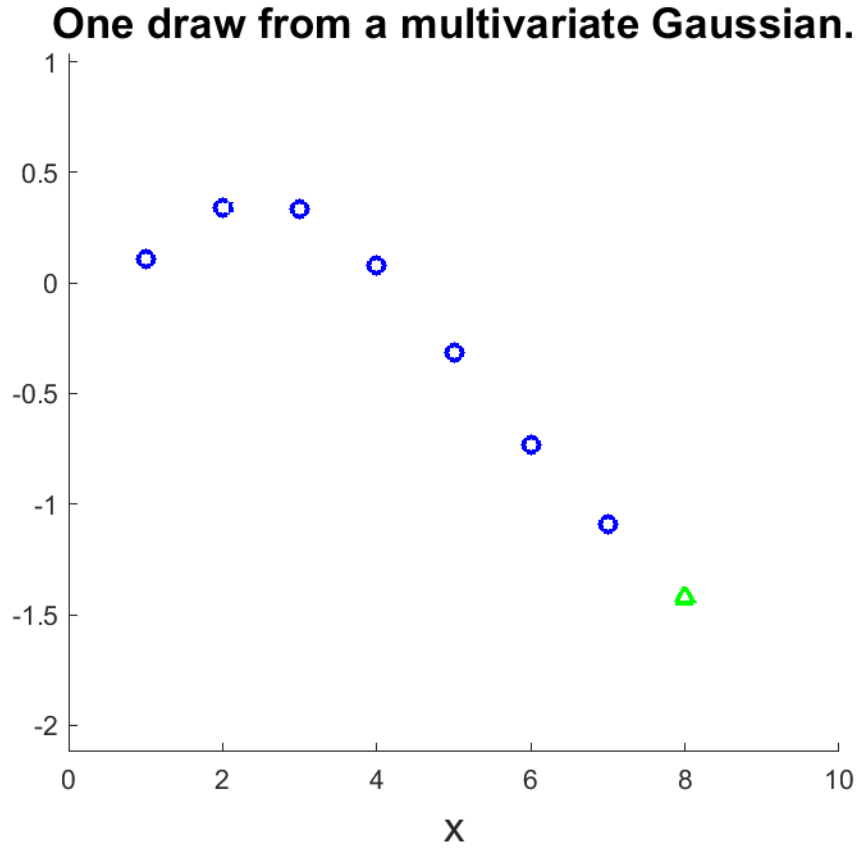


Gaussian distribution

- We have an 8-dimensional multivariate Gaussian distribution

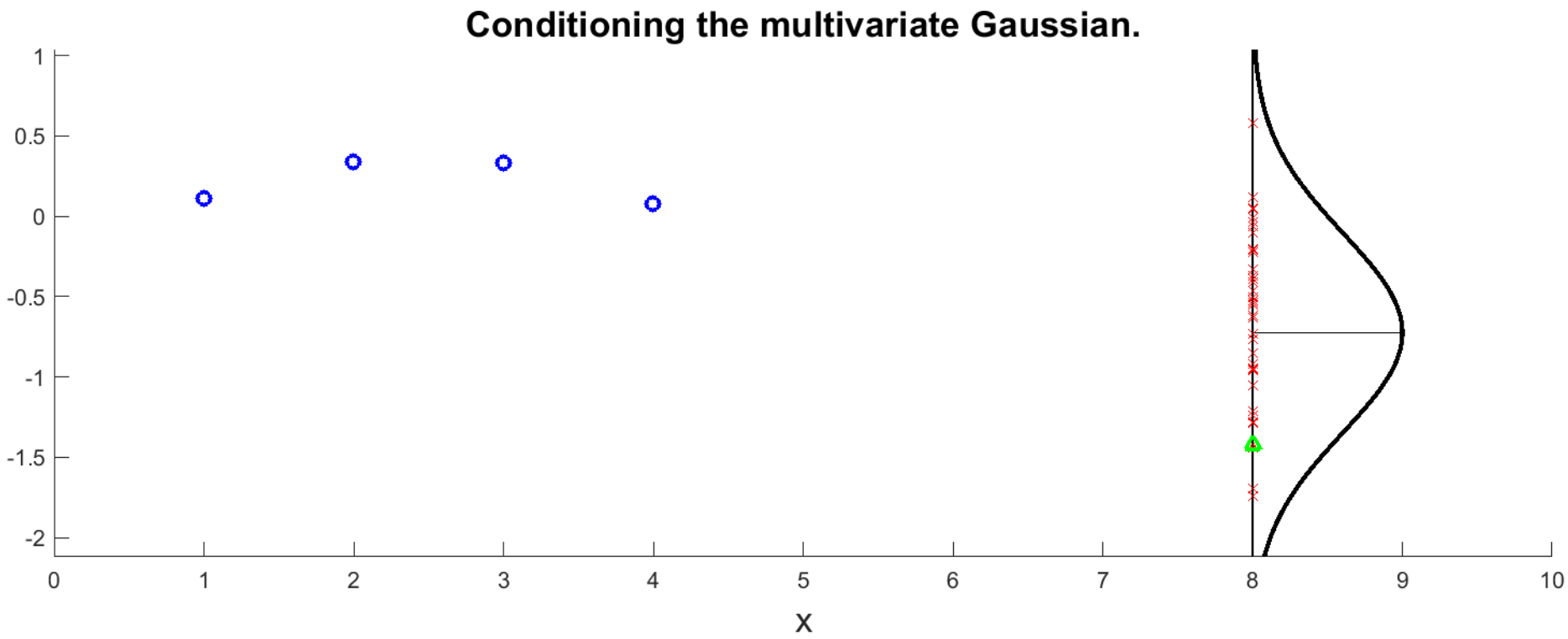
$$p(x_{1:8})$$





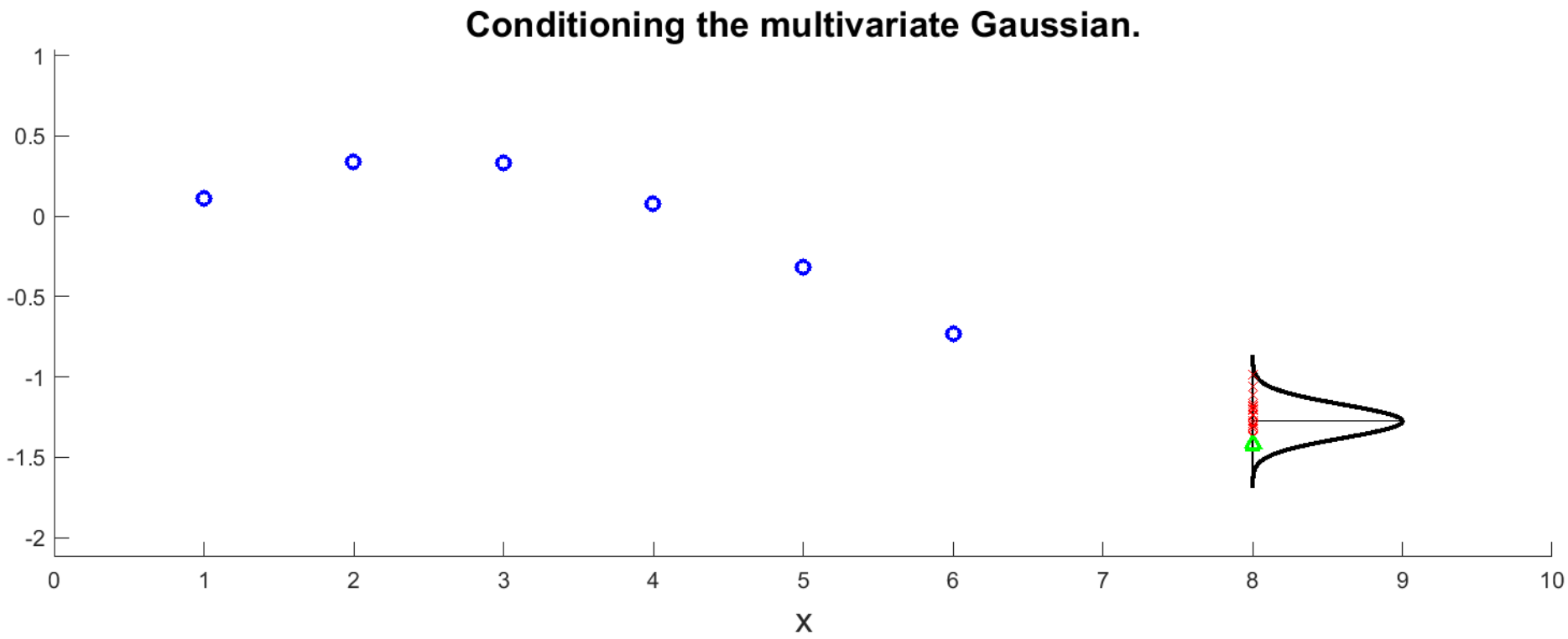
Gaussian distribution

$$p(x_8|x_{1:4})$$



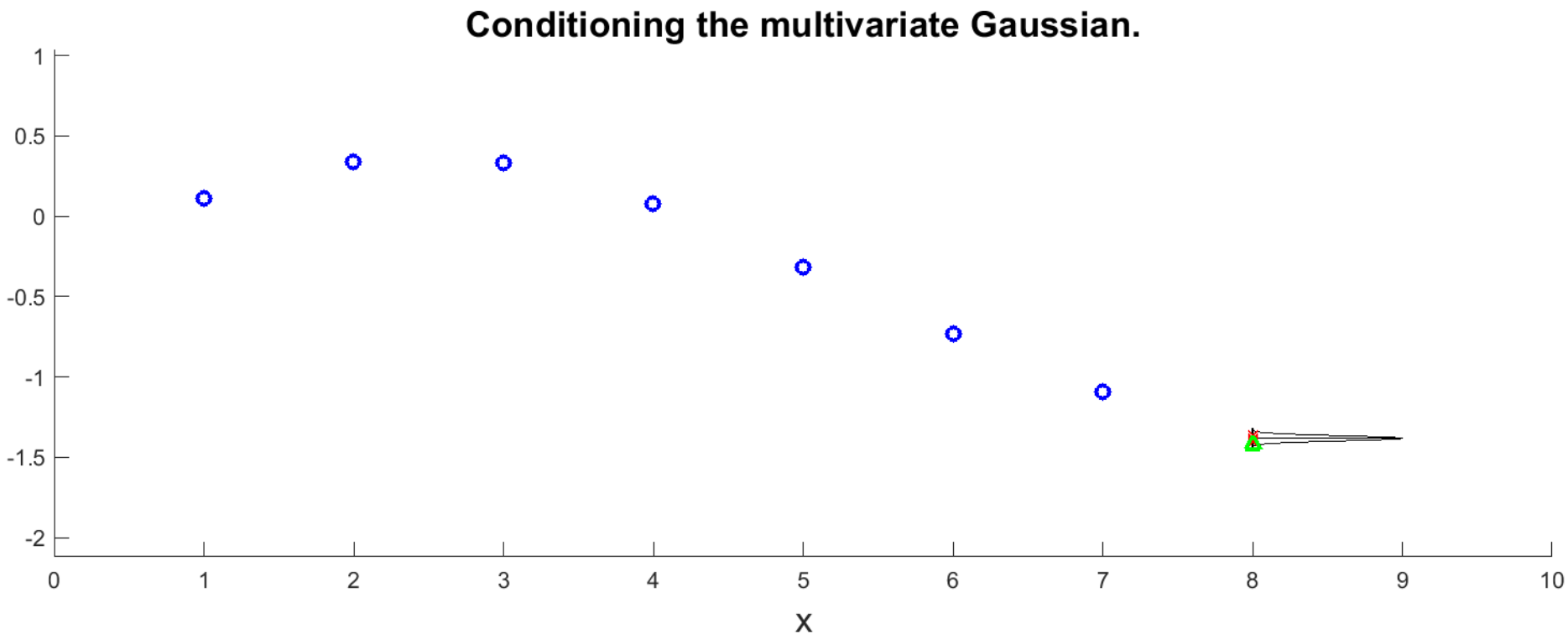
Gaussian distribution

$$p(x_8|x_{1:6})$$



Gaussian distribution

$$p(x_8|x_{1:7})$$



Gaussian Processes

- **D:** A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

- Completely described by it's mean and covariance function

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

  - Usually we assume $m(\boldsymbol{x}) = 0$, but this isn't mandatory

- Nonparametric Bayesian model
- We will discuss GPs for regression

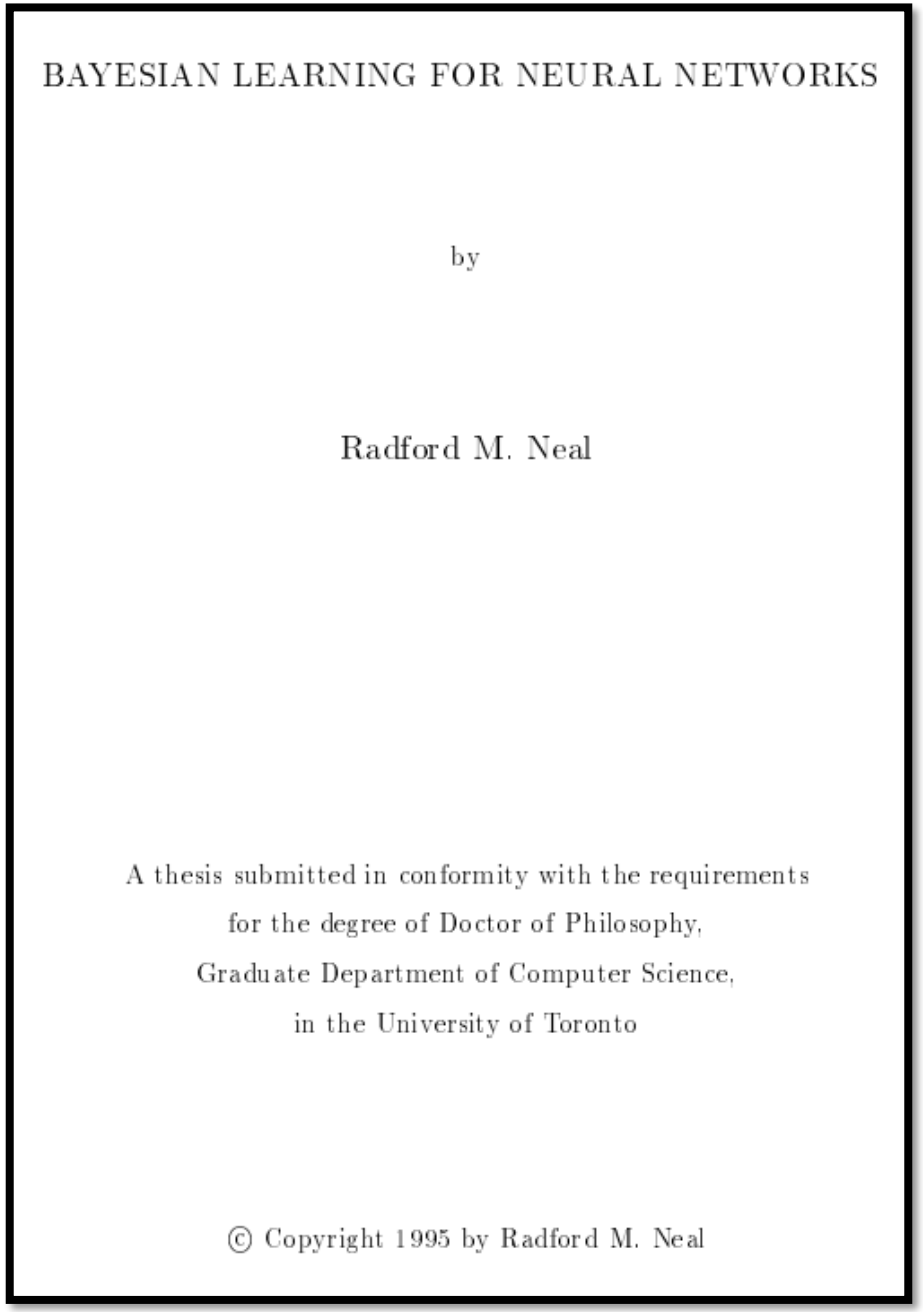
Parametric and nonparametric models

- Parametric models encode the information from the training set into a set of parameters $\mathbf{w}$
- Usually the $\mathbf{w}$ is of a much lower dimension than the number of datapoints
- To make a prediction we don't need the training set, we just need $\mathbf{w}$

- Nonparametric models don't make this assumption
- They usually use all the training data to make a prediction

Expressiveness of GPs

- One-hidden-layer Bayesian NN converges to a Gaussian Process as the number of the neurons in the layer goes to infinity. (Chapter 2.1)



BAYESIAN LEARNING FOR NEURAL NETWORKS

by

Radford M. Neal

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy,
Graduate Department of Computer Science,
in the University of Toronto

Gaussian Processes

- Model:

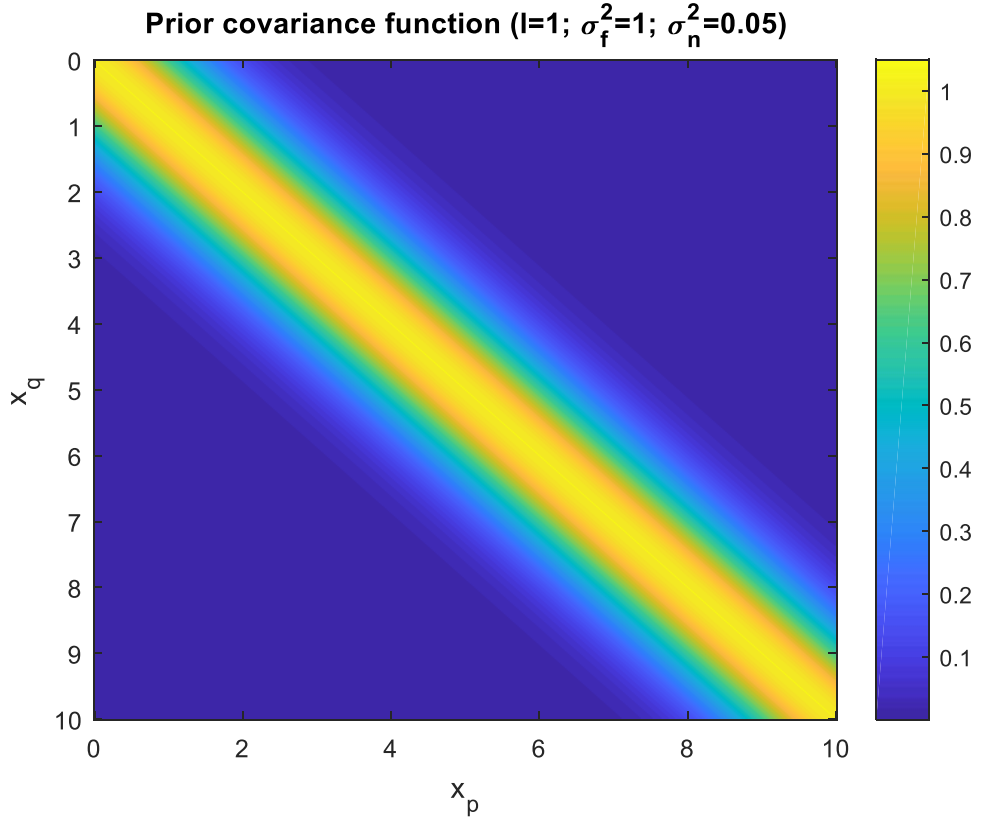
$$y_i = f(\boldsymbol{x_i}) + \epsilon_i$$
$$\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$$

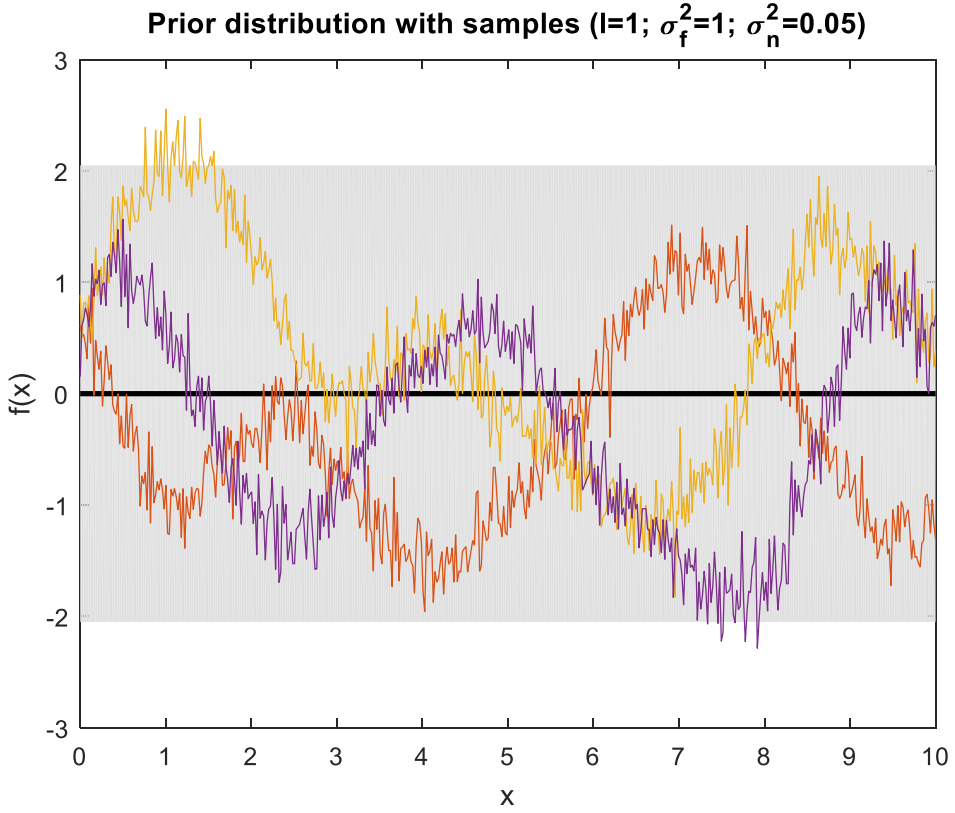
- We first need to specify a prior for $f(\boldsymbol{x_i})$

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

Prior distribution

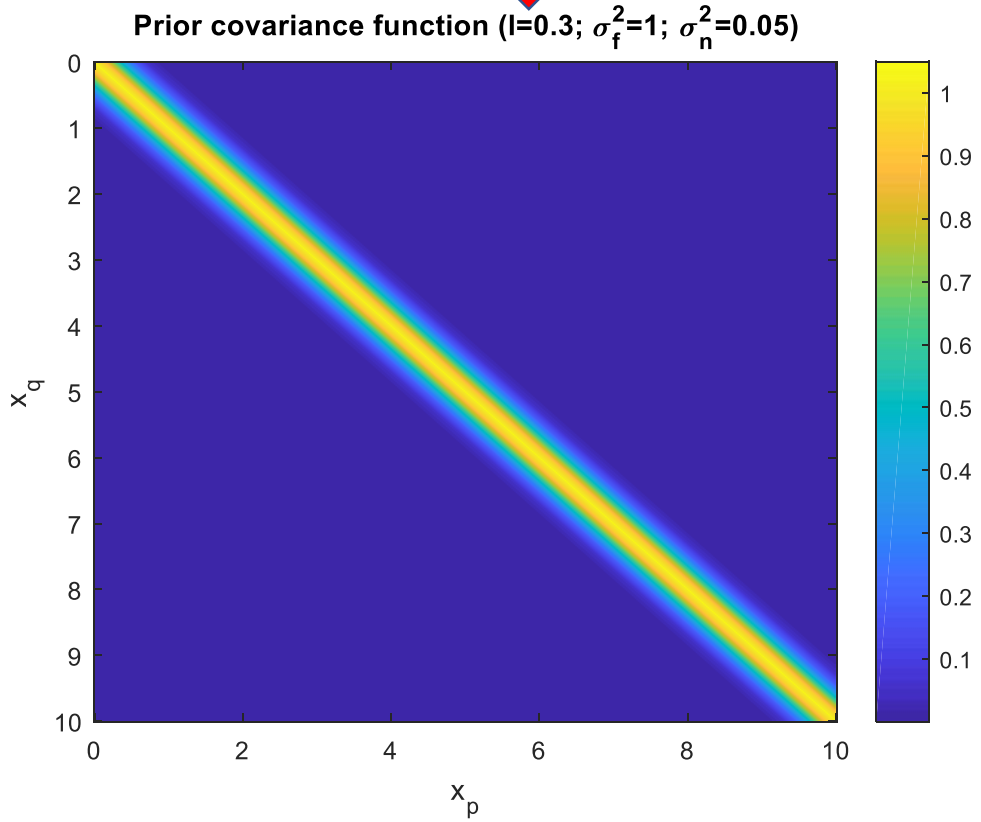
$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2}{2l^2}\right)$$

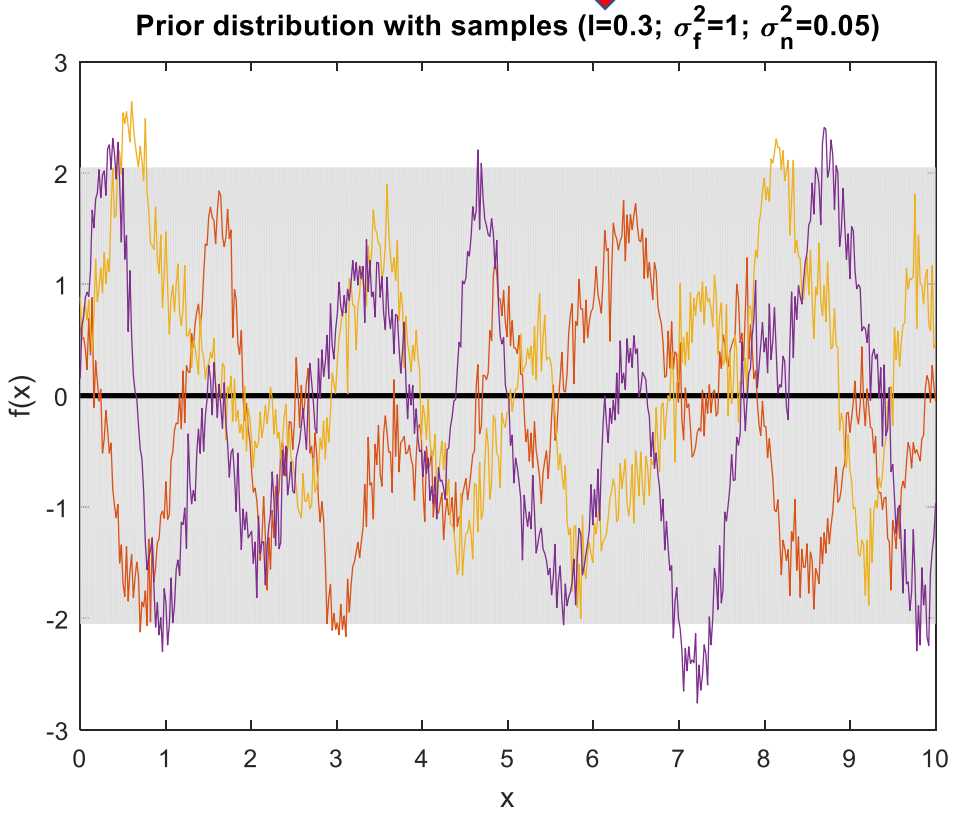




Prior distribution

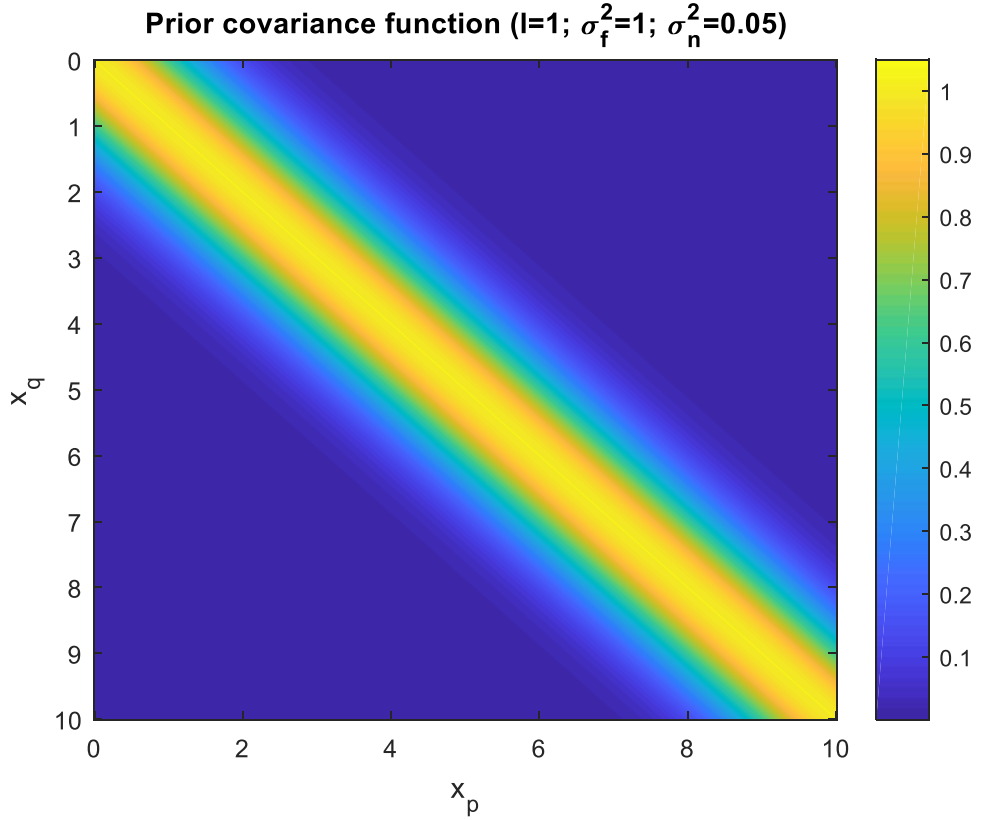
$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2}{2l^2}\right)$$

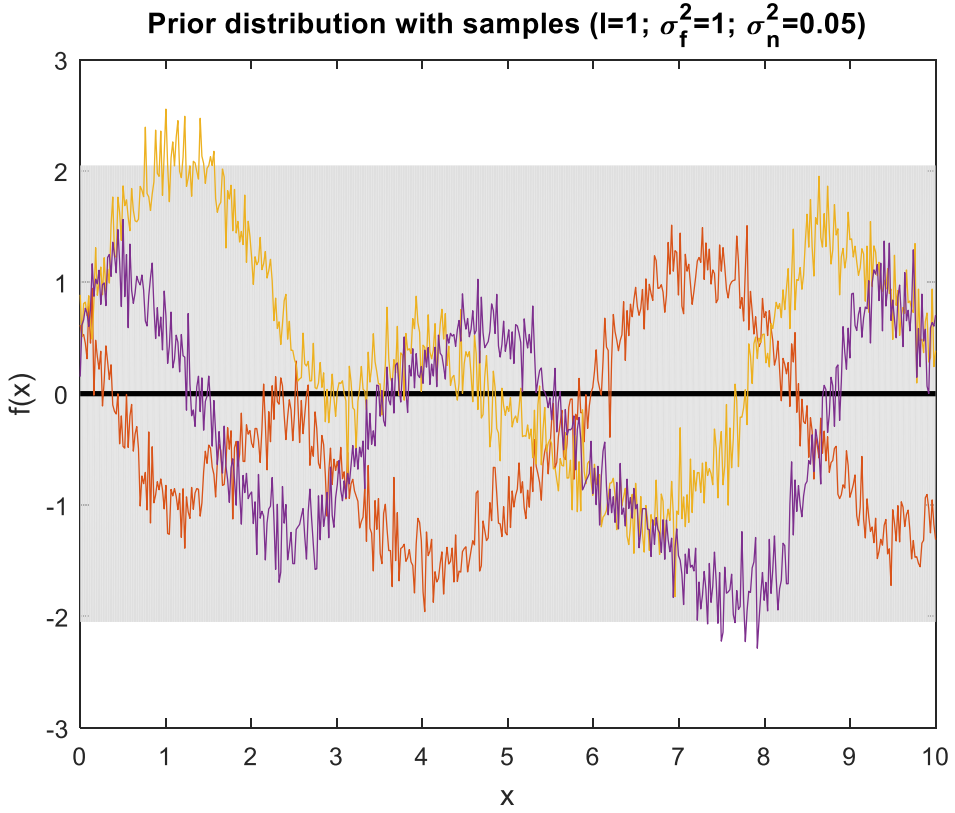




Prior distribution

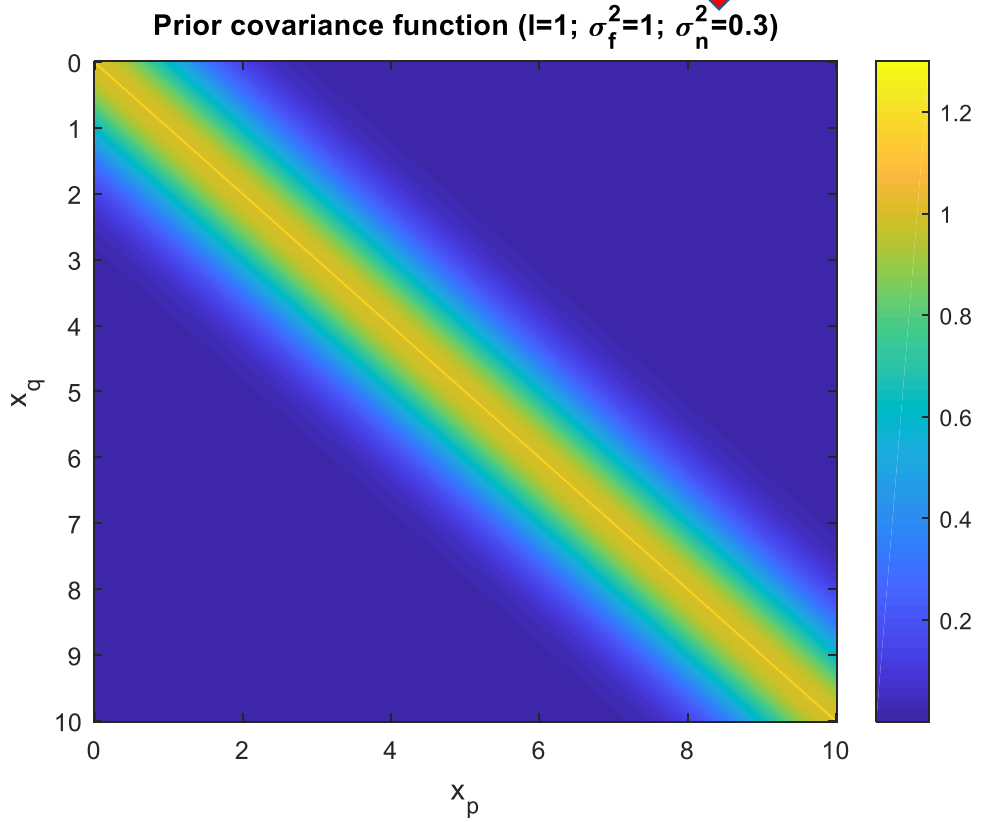
$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2}{2l^2}\right)$$

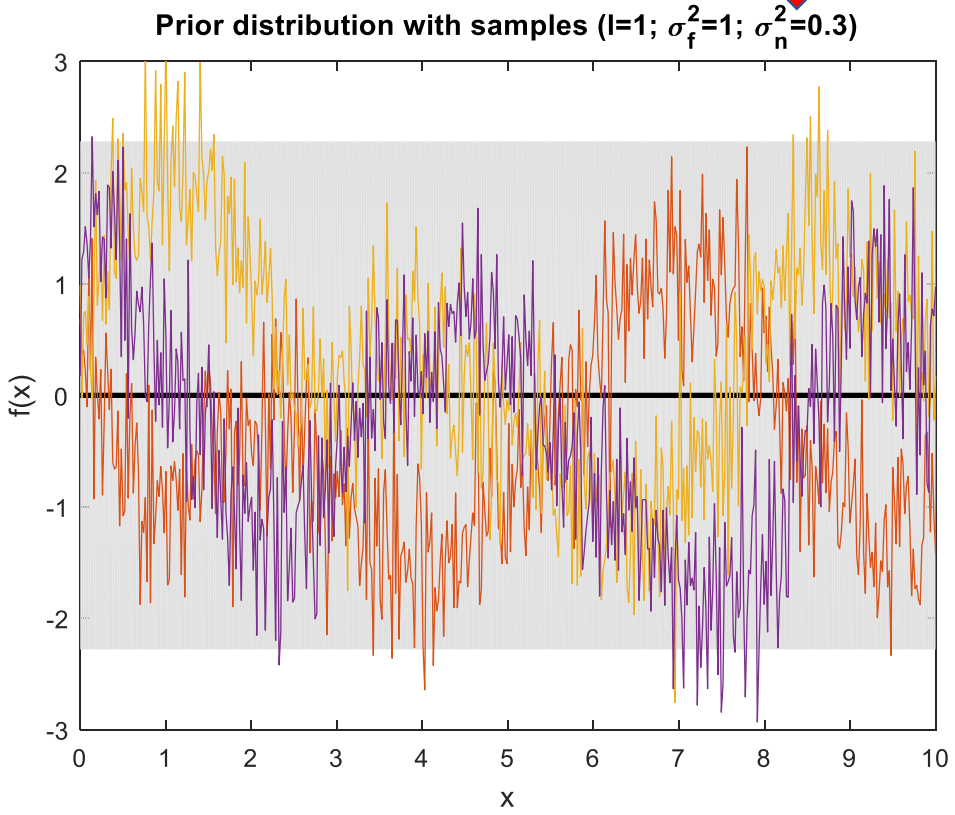




Prior distribution

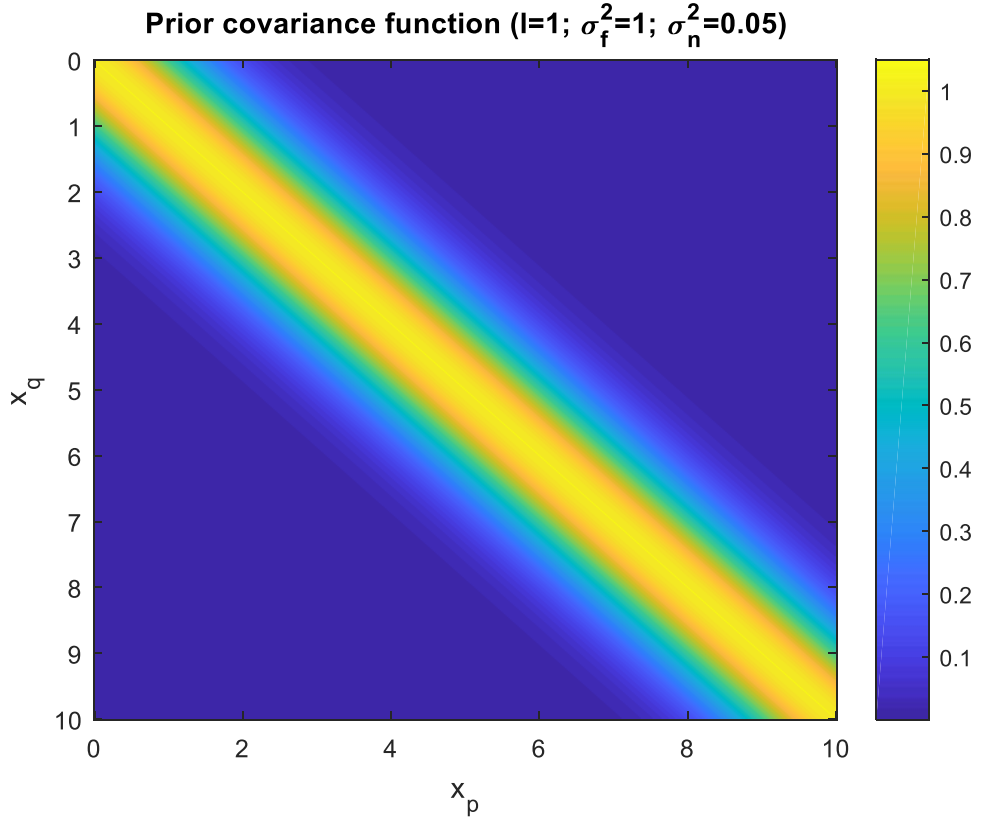
$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2}{2l^2}\right)$$

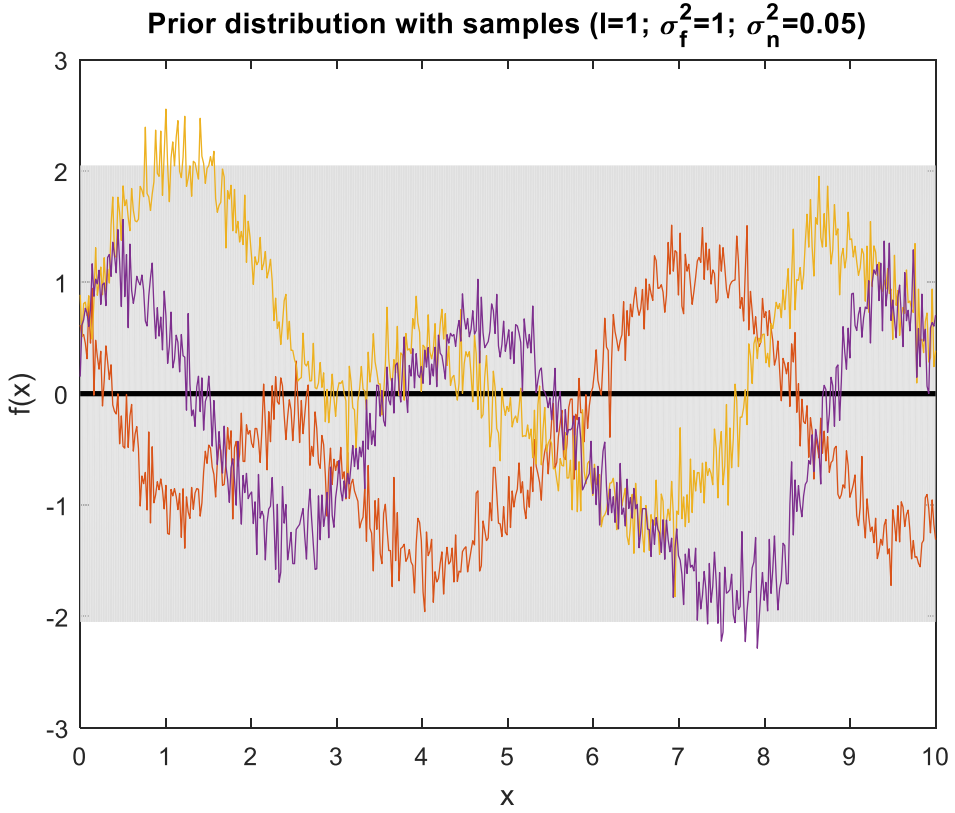




Prior distribution

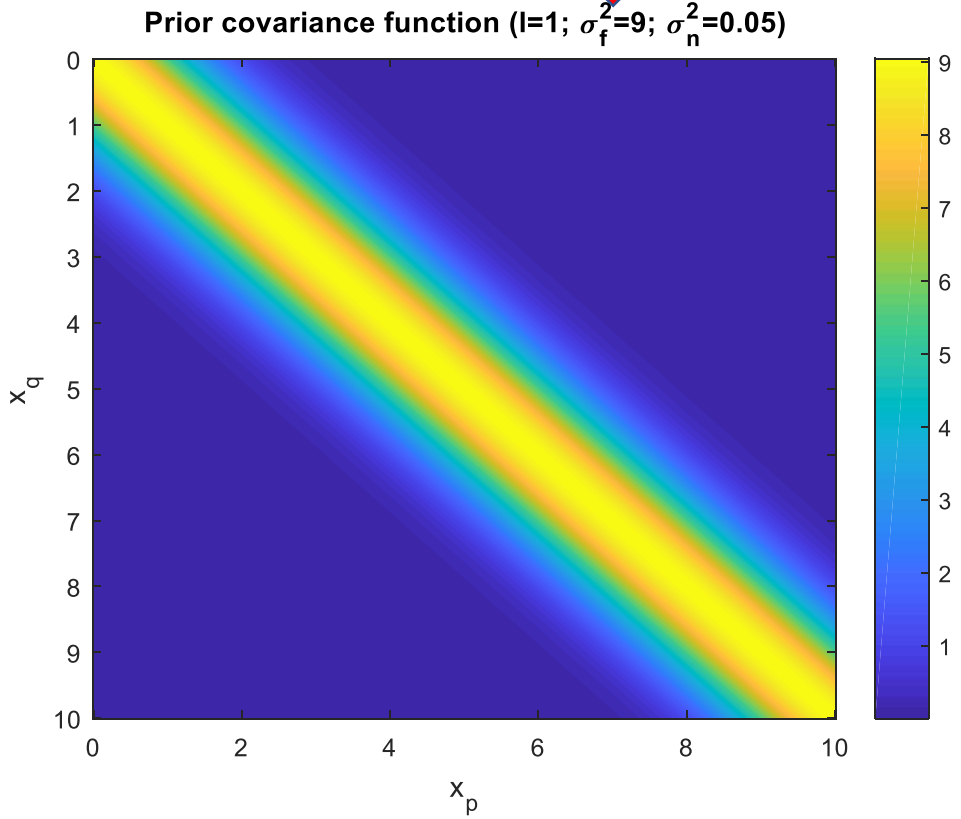
$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2}{2l^2}\right)$$

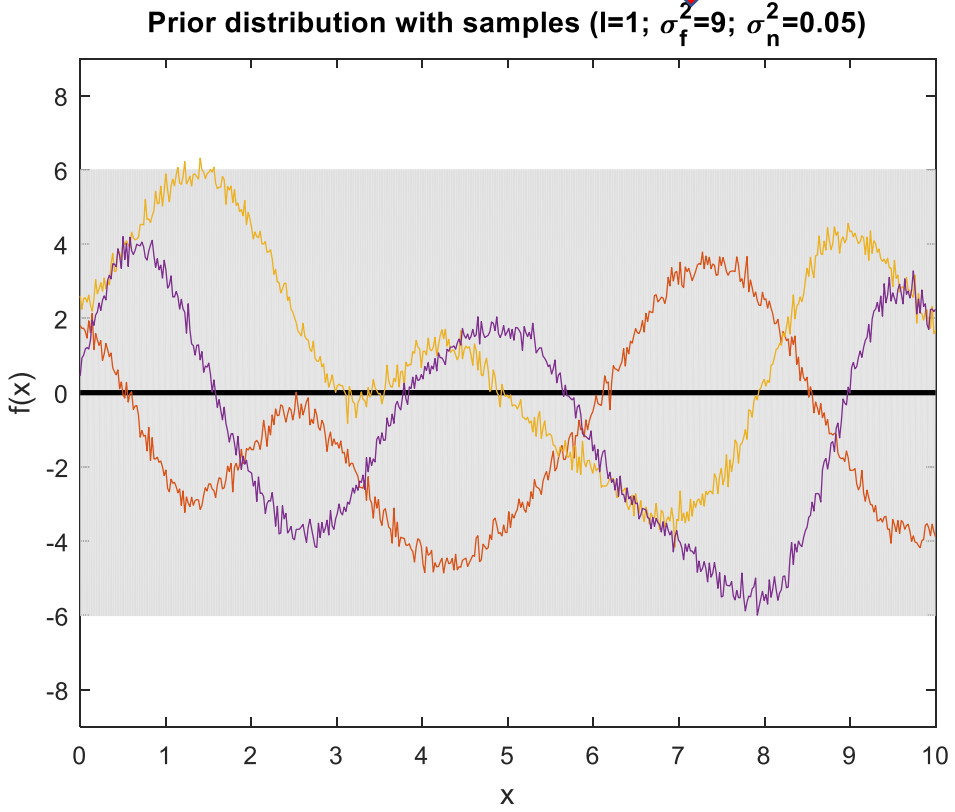




Prior distribution

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2}{2l^2}\right)$$





Gaussian Processes

- We have a training set $\mathcal{D} = (\mathbf{X}, \mathbf{y})$
- We want to predict function values f_* at the input location $\mathbf{x}_*$

- From the specified prior we can write the joint distribution

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I} & \mathbf{k}(\mathbf{X}, \mathbf{x}_*) \\ \boldsymbol{k}(\mathbf{x}_*, \mathbf{X}) & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right)$$

- In shorter notation

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma^2 \mathbf{I} & \mathbf{k}_* \\ \mathbf{k}_*^{\mathrm{T}} & k_{**} \end{bmatrix}\right)$$

Gaussian Processes

- Posterior (predictive)distribution for a new input $\boldsymbol{x}_*$, using the training set $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$

$$f_* | \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$$

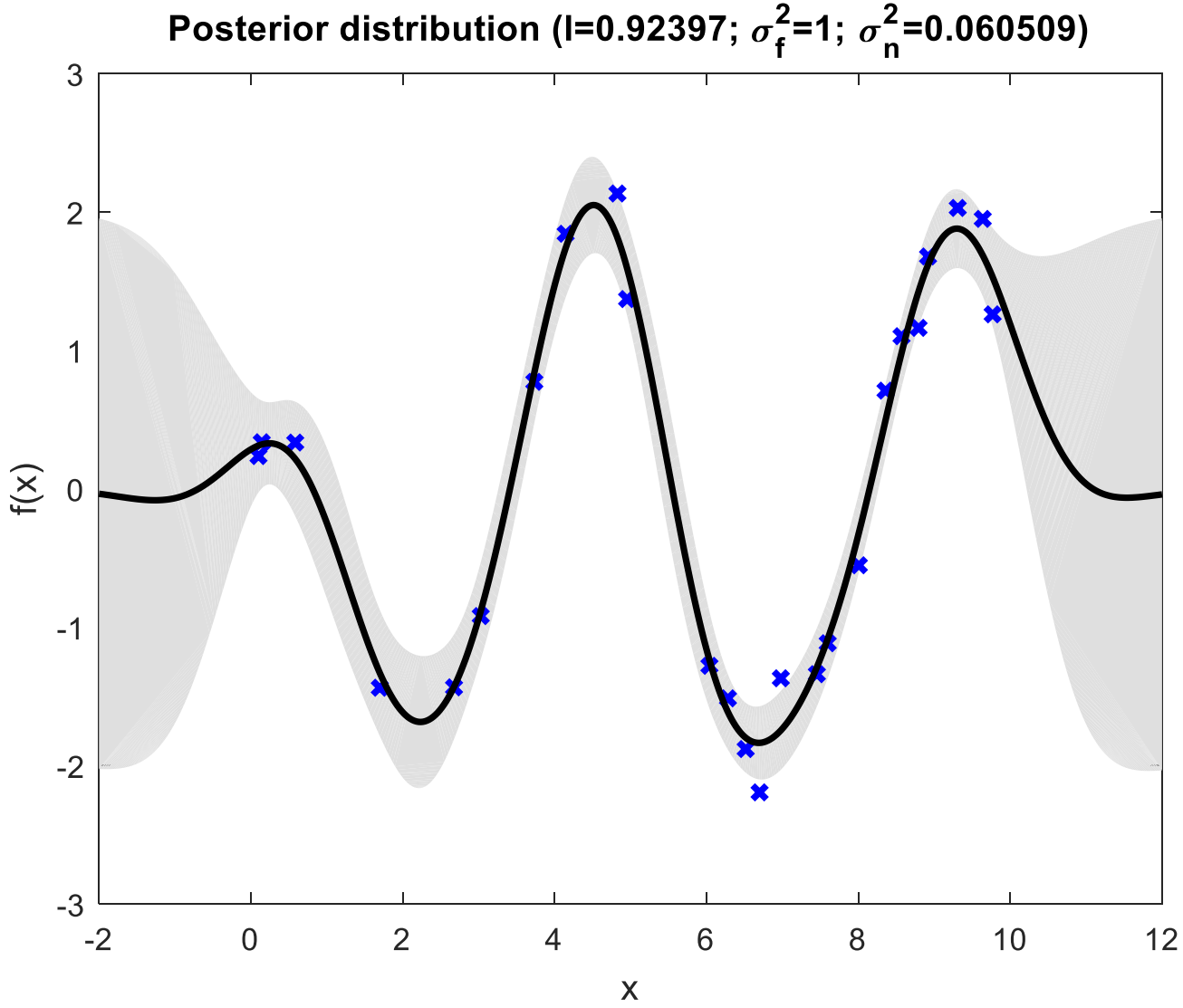
$$\mu(\boldsymbol{x}_*) = \boldsymbol{k}_*^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \mathbf{y}$$

$$\sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{k}_*$$

- $(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1}$ has complexity $O(n^3)$
- There is a good and numerically stable way of implementig these equations ([1], page 19.)

Posterior (predictive) distribution

$$p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})$$



Gaussian Processes in a different view

- $f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}$ looks like Bayesian linear regression in a feature space

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \boldsymbol{\Phi}(\mathbf{X})^{\mathrm{T}} \boldsymbol{\Sigma}_{\boldsymbol{p}} \boldsymbol{\Phi}\ (\mathbf{X})$$

- For every transformation in the feature space we can calculate the corresponding kernel

- For each positive-definite covariance function k, there exists some feature space ϕ

Making the final decision

- When we need to give a point-wise prediction we could take the mean of the predictive distribution

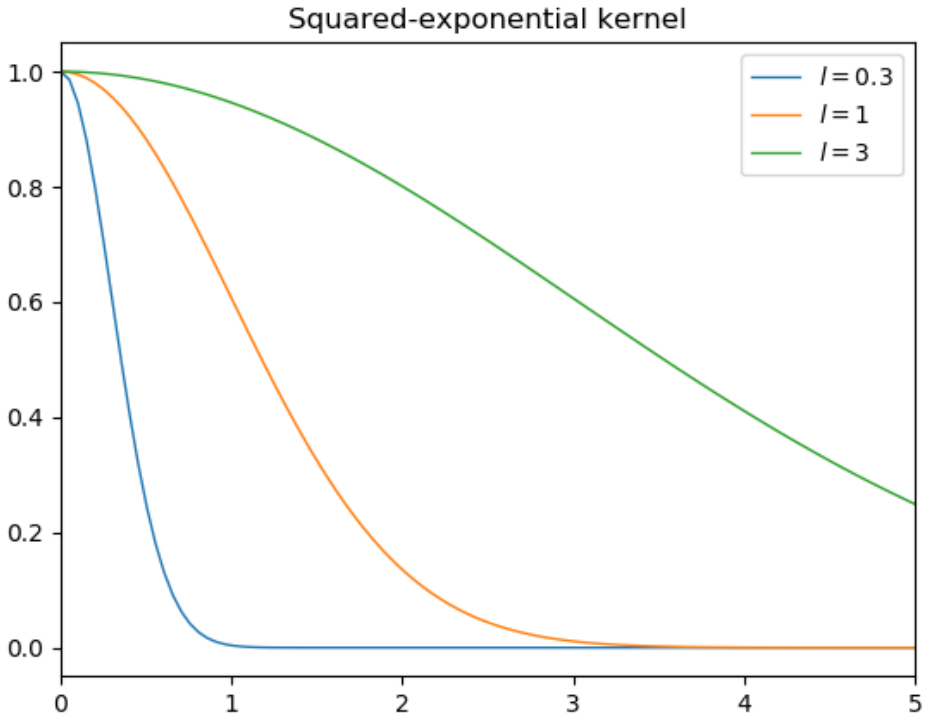
$$\mu(\boldsymbol{x}_*)$$

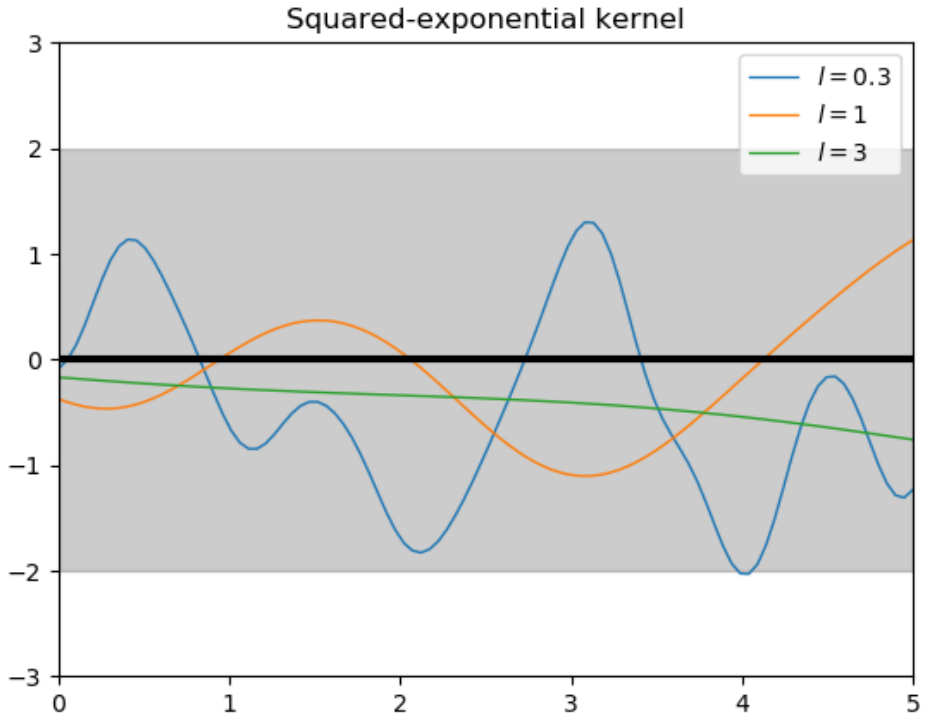
Squared-exponential kernel

- Very smooth
- l is the characteristic length-scale
- Each dimension could have its own lenght-scale (ARD)

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \exp\left(-\frac{\|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2}{2l^2}\right)$$

Realizations without noise



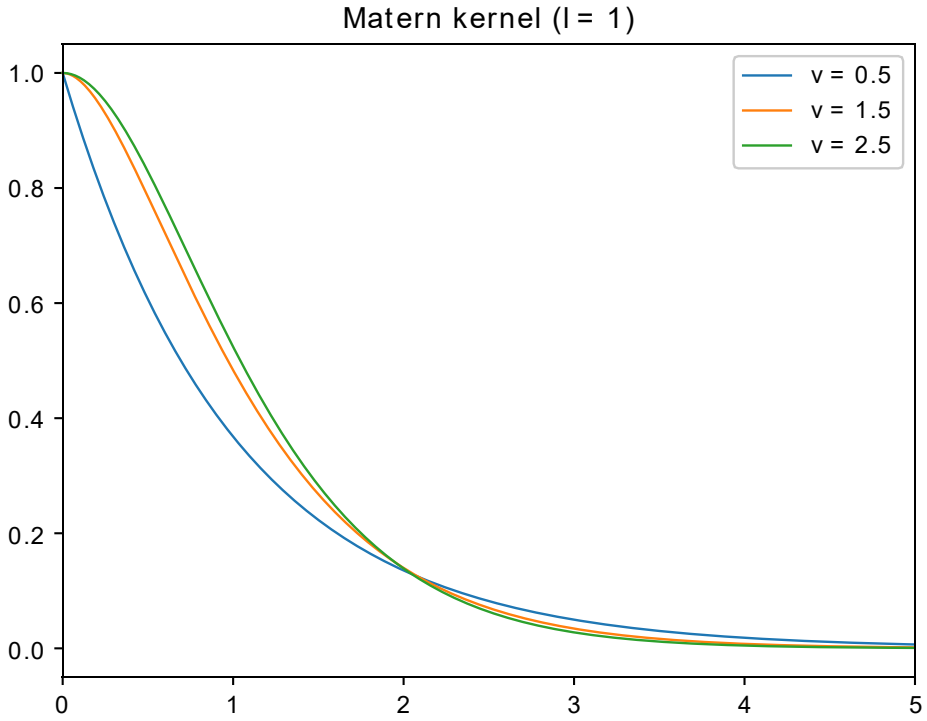


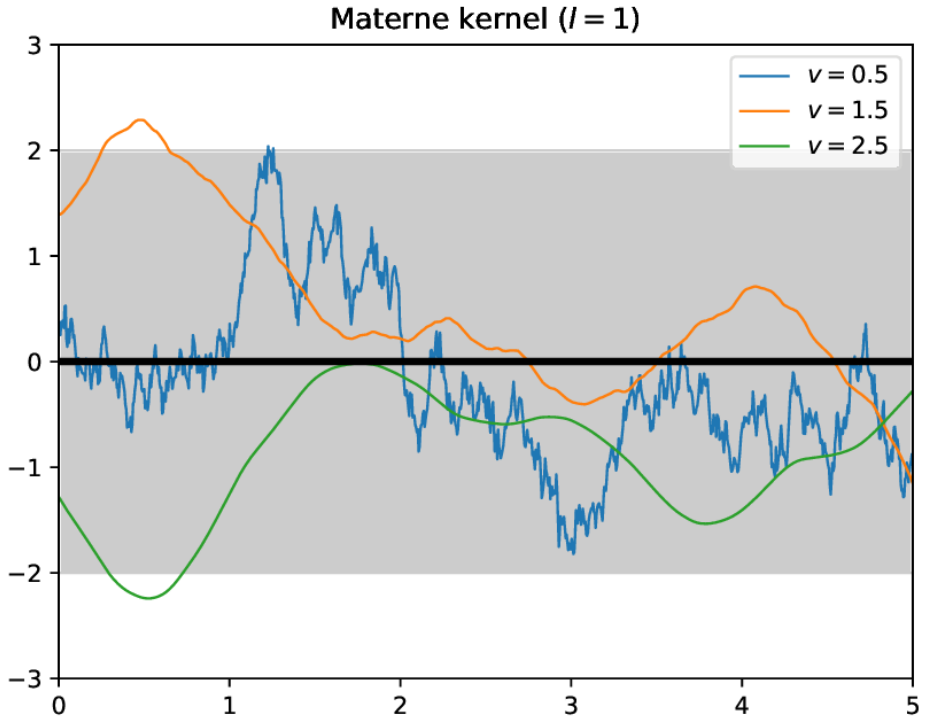
Matérn class kernels

- $v \to \infty$ leads to a squared-exponential kernel
- v determines the smoothness
- l is the length-scale

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \frac{2^{1-v}}{\Gamma(v)} \left(\frac{\sqrt{2v}\|\boldsymbol{x}_p - \boldsymbol{x}_q\|}{l} \right)^{v} \mathrm{K_v} \left(\frac{\sqrt{2v}\|\boldsymbol{x}_p - \boldsymbol{x}_q\|}{l} \right)$$

Realizations without noise



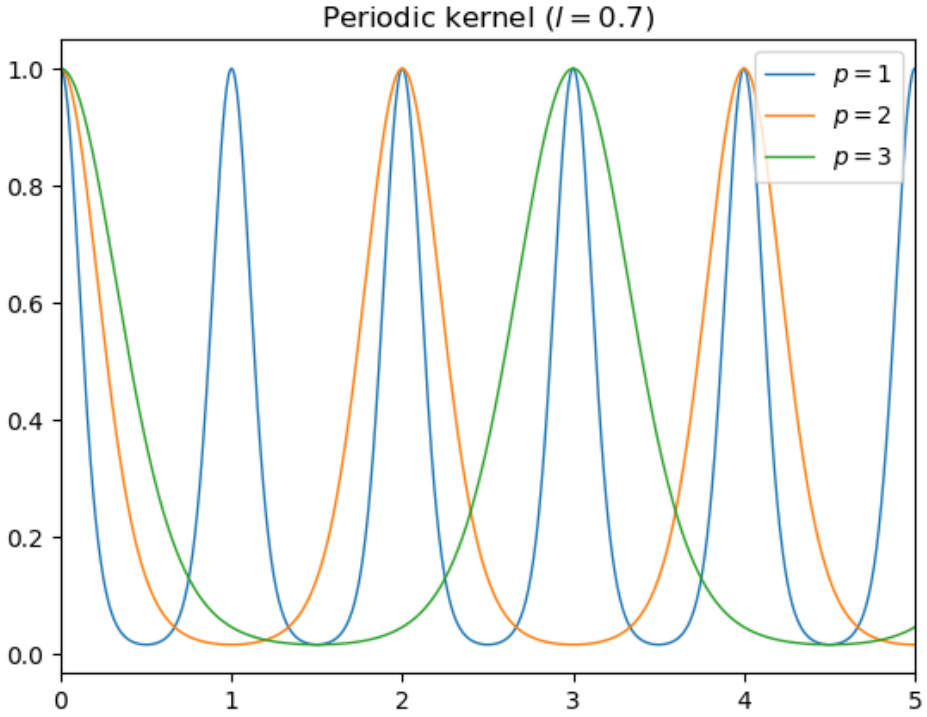


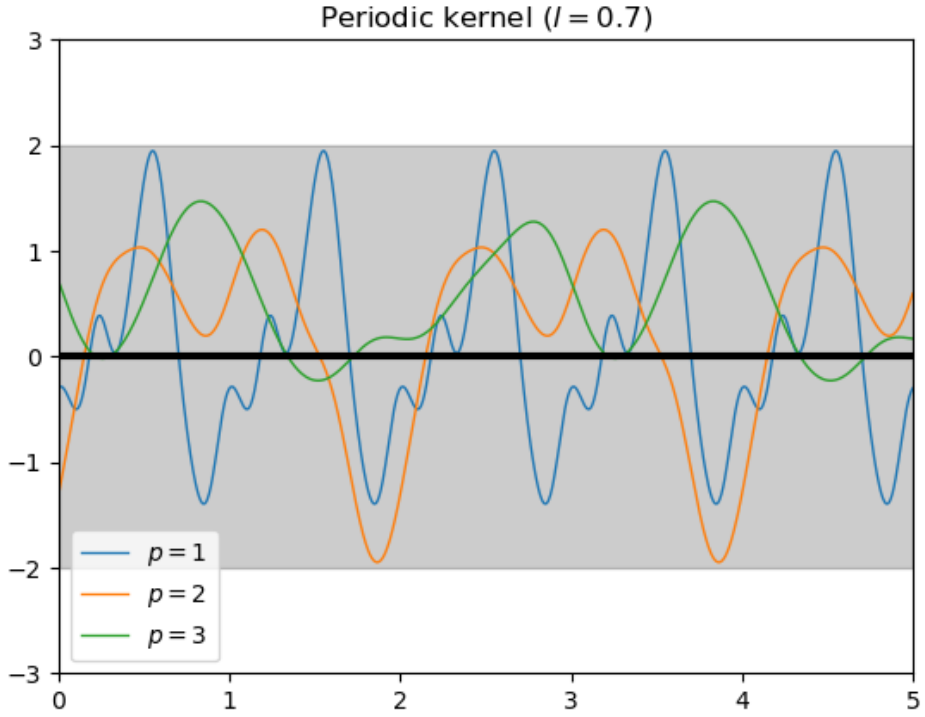
Periodic kernel

- p determines the period
- l is the length-scale

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \exp\left(-\frac{2\sin^2\left(\pi\|\boldsymbol{x}_p - \boldsymbol{x}_q\|/p\right)}{l^2}\right)$$

Realizations without noise





Generating covariance functions

- There are a lot of different kernels

- Kernels can be combined
  - A sum of two kernels is a valid kernel
  - A product of two kernels is a valid kernel
    - We could multiply each kernel with the process variance σ_f^2

Model selection (training)

- The model here will be the mean $m(\boldsymbol{x})$ and the covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$, along with their hyperparameters θ

- Model selction could give an interpretation of the data

- Bayesian model selection
  - Maximum likelihood type-II approximation
- Crossvalidation

General Bayesian model selection

$$\boxed{p(\mathbf{w}|\mathbf{y},\mathbf{X},\boldsymbol{\theta},\mathcal{H}_i)} = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w},\mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta},\mathcal{H}_i)}{p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta},\mathcal{H}_i)}$$

$$\boxed{p(\boldsymbol{\theta}|\mathbf{y},\mathbf{X},\mathcal{H}_i)} = \frac{p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta},\mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)}{p(\mathbf{y}|\mathbf{X},\mathcal{H}_i)}$$

$$\boxed{p(\mathcal{H}_i|\mathbf{y},\mathbf{X})} = \frac{p(\mathbf{y}|\mathbf{X},\mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y}|\mathbf{X})}$$

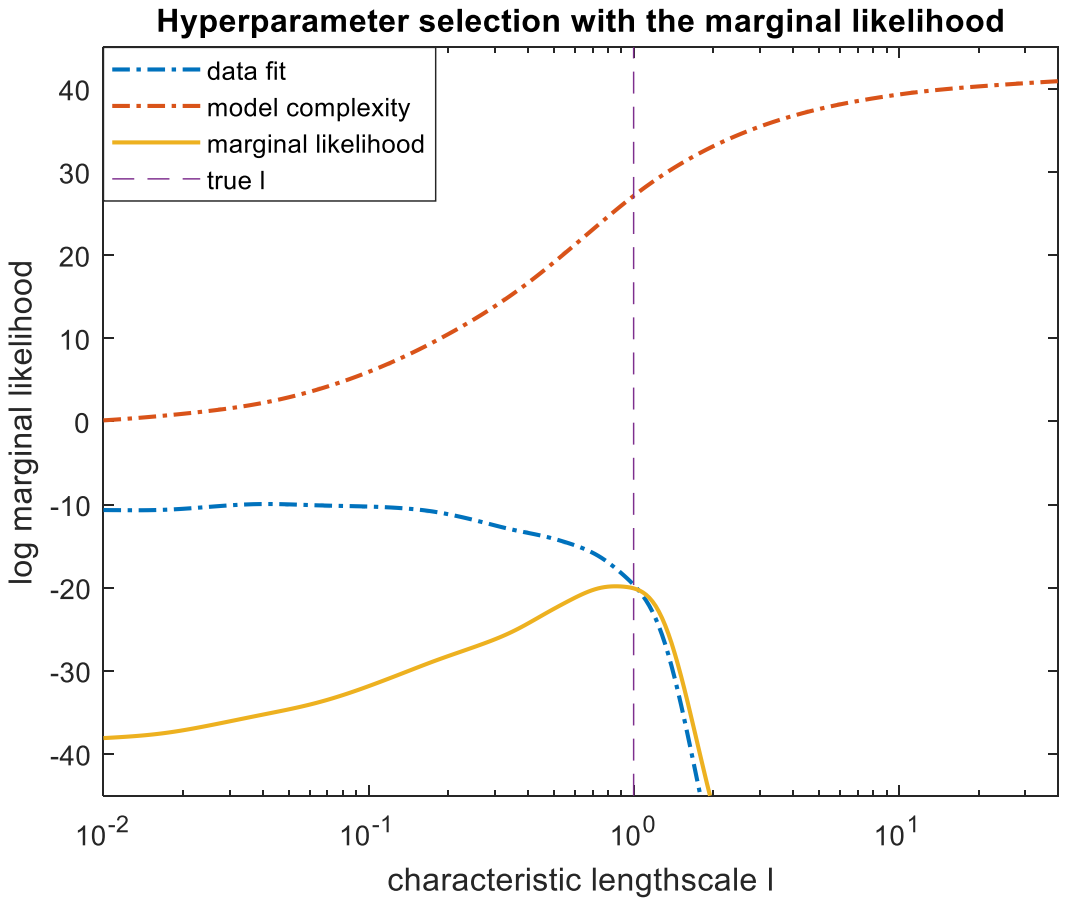
Bayesian model selection (training)

penalty for model complexity

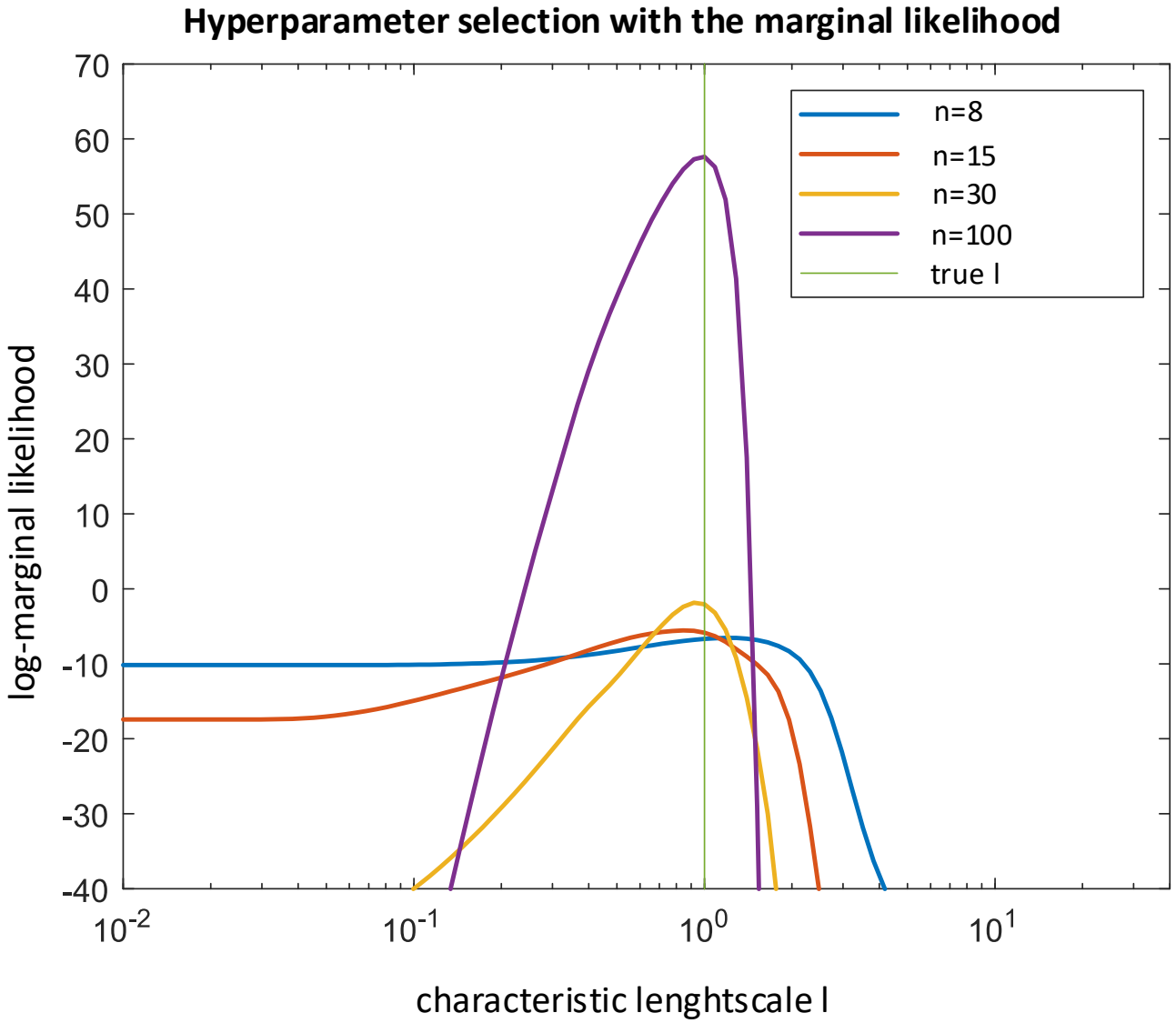
$$\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta},\mathcal{H}_i) = -\frac{1}{2}\mathbf{y}(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}+\sigma_n^2\mathbf{I}| - \frac{n}{2}\log(2\pi)$$

data fit

- Each evidence evaluation costs $O(n^3)$ (matrix inversion)
- Numerically stable implementation of the equation ([1], page 19.)

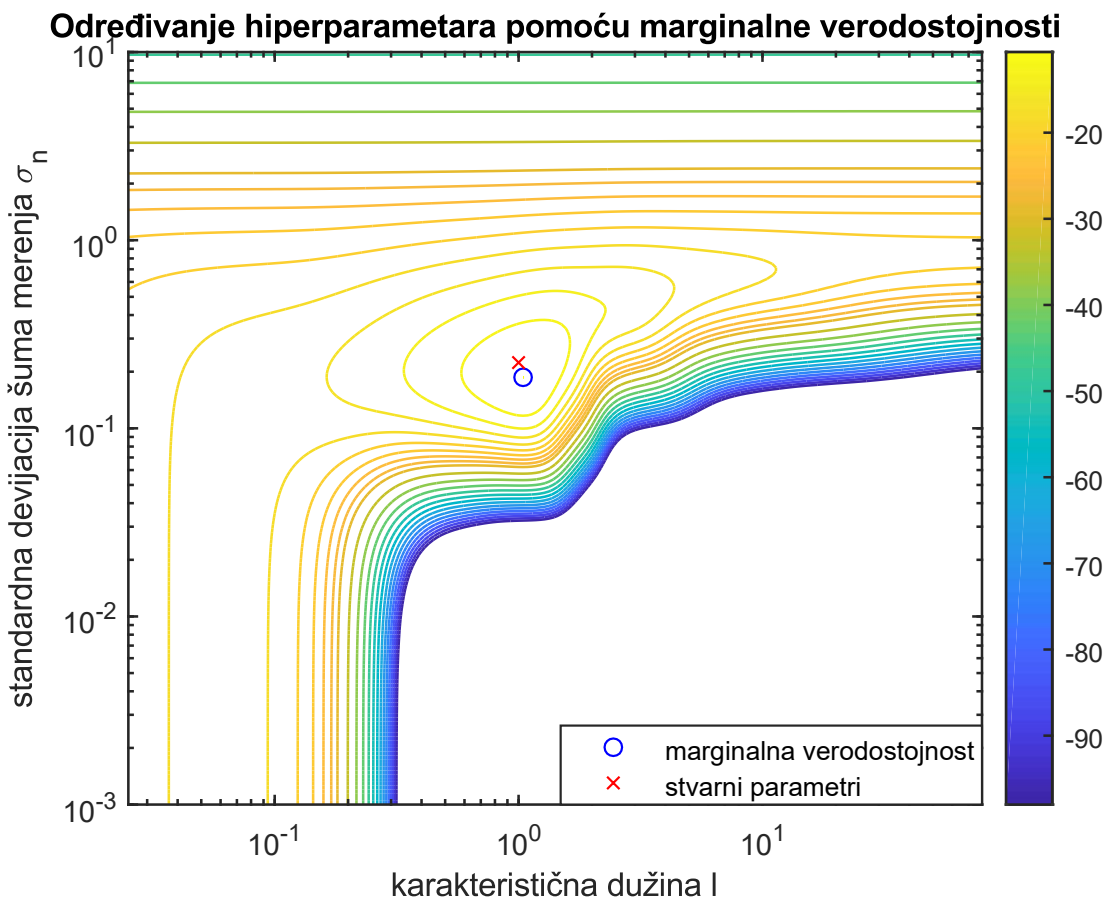


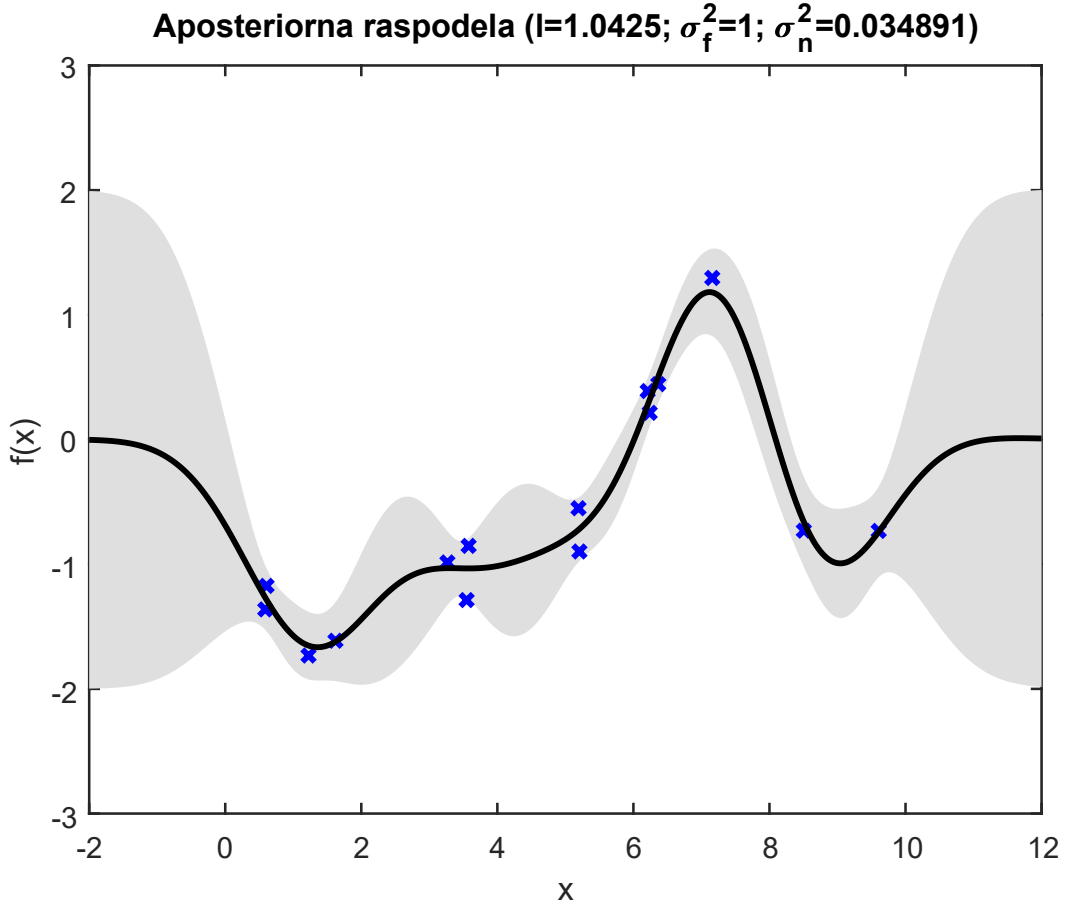
Bayesian model selection (training)



Bayesian model selection (training)

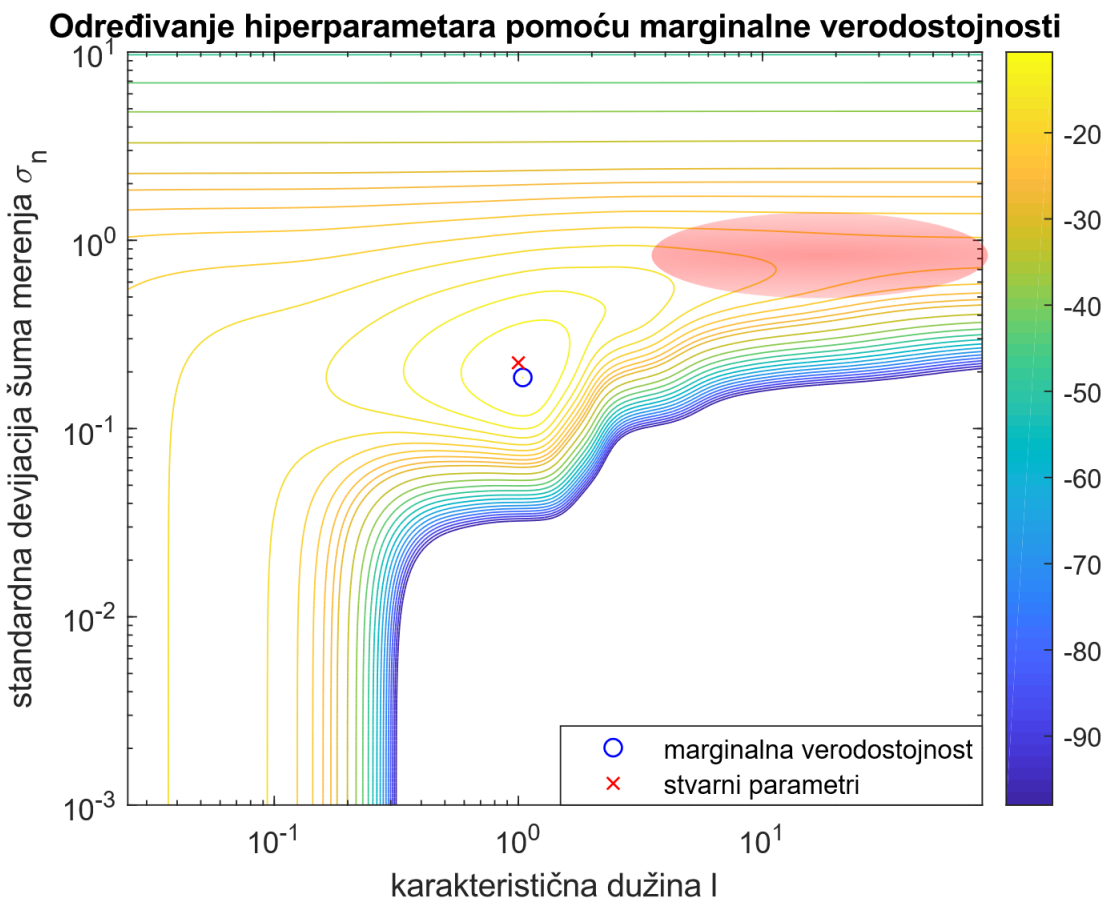
- 15 training datapoints

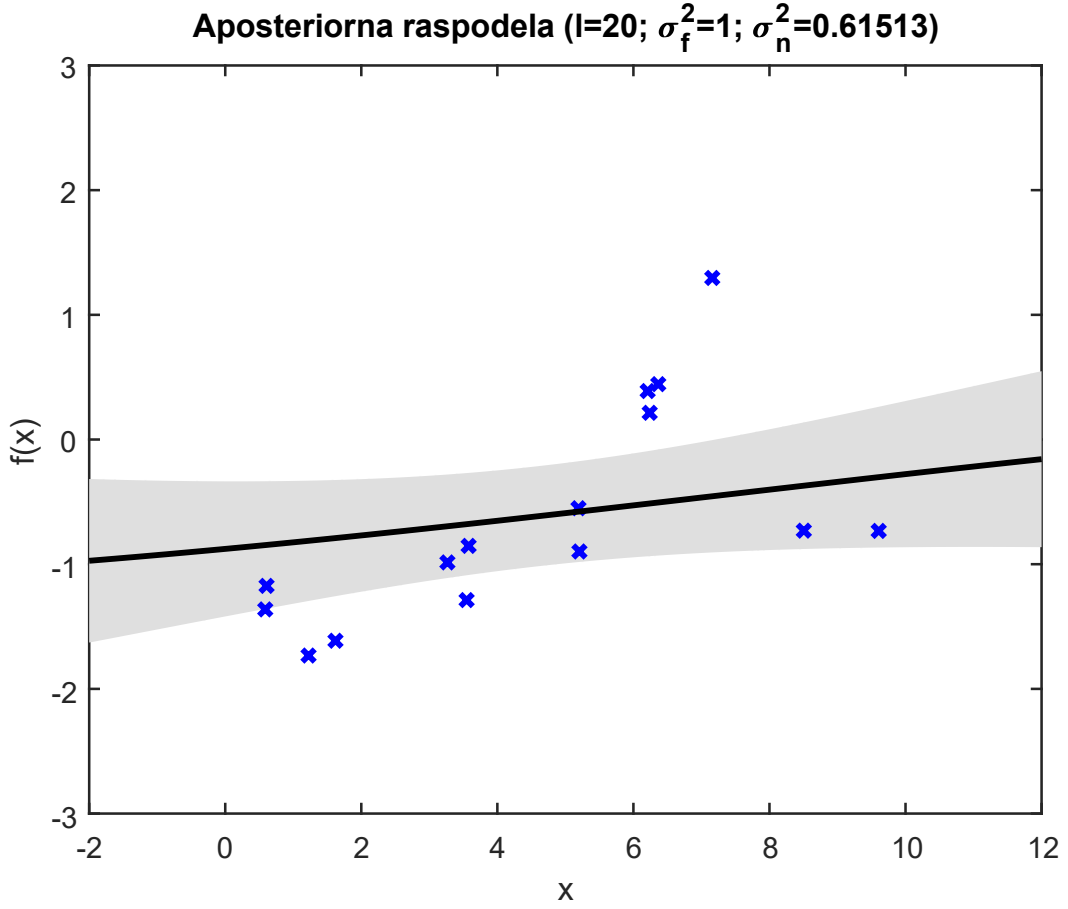




Bayesian model selection (training)

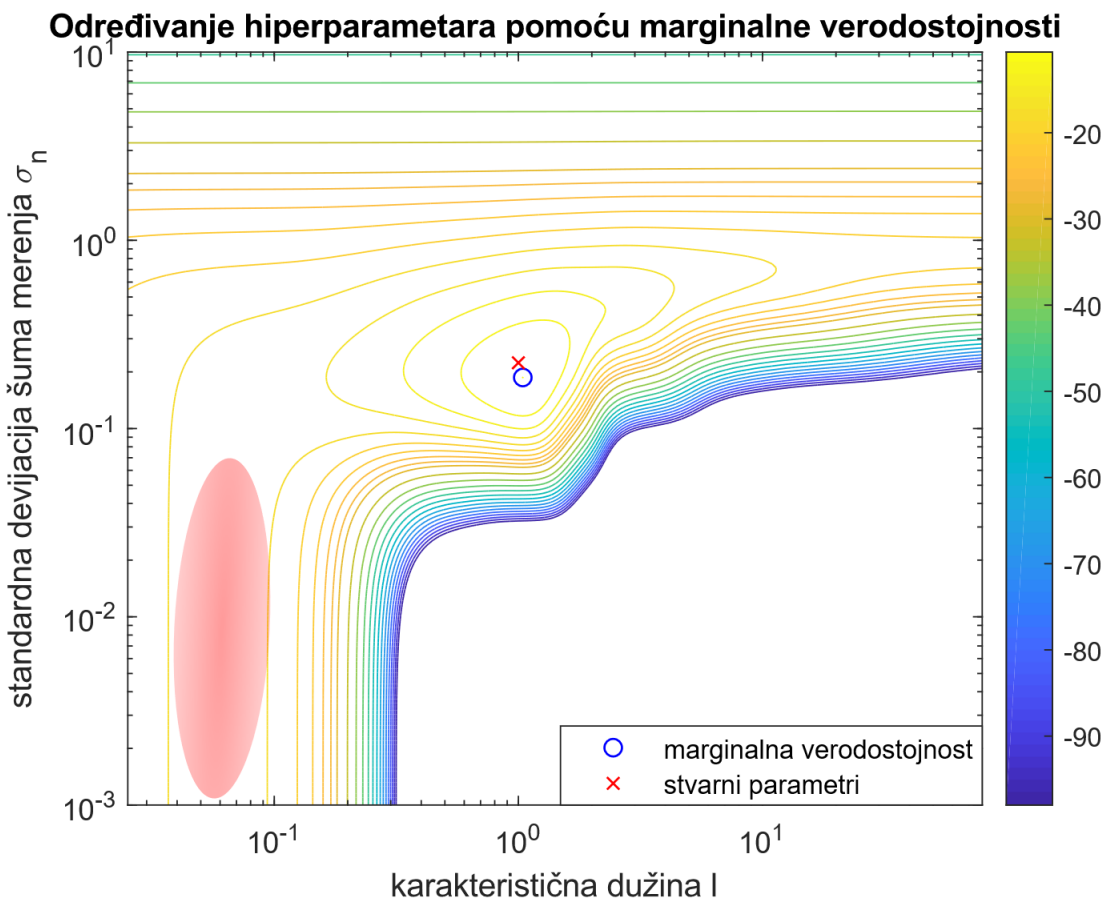
- 15 training datapoints

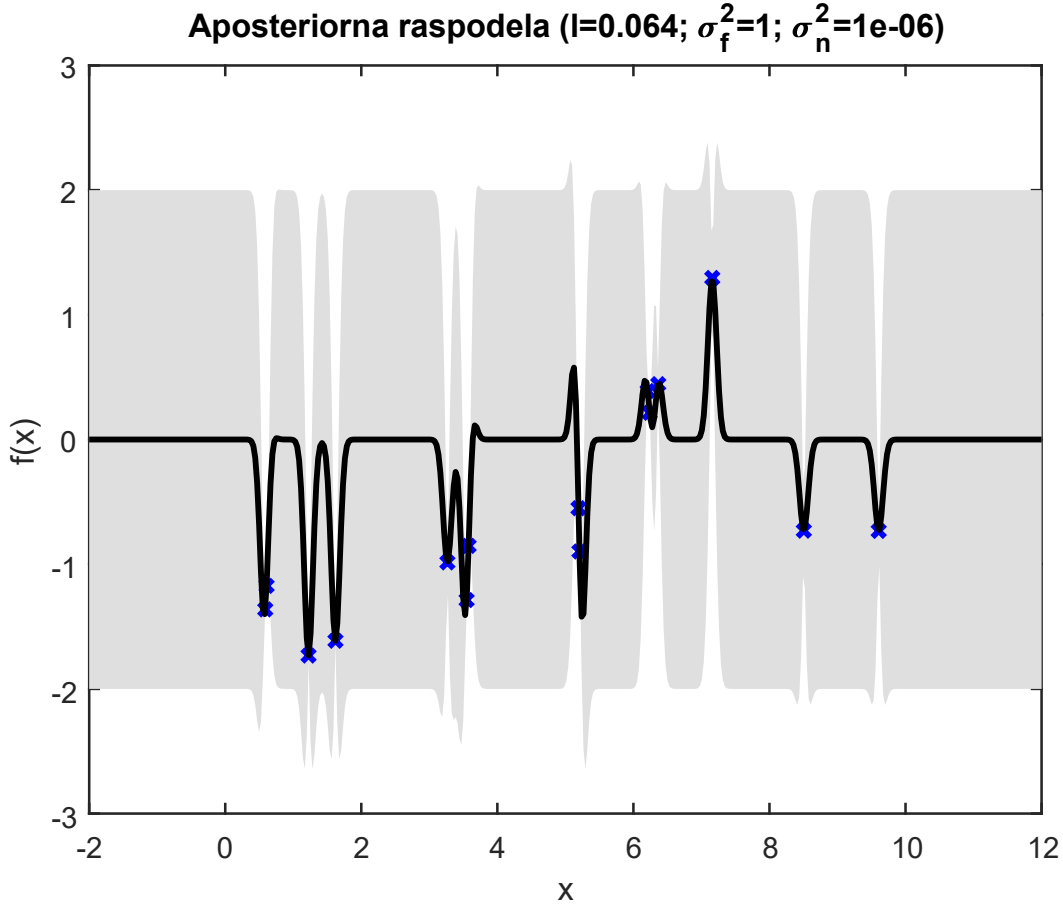




Bayesian model selection (training)

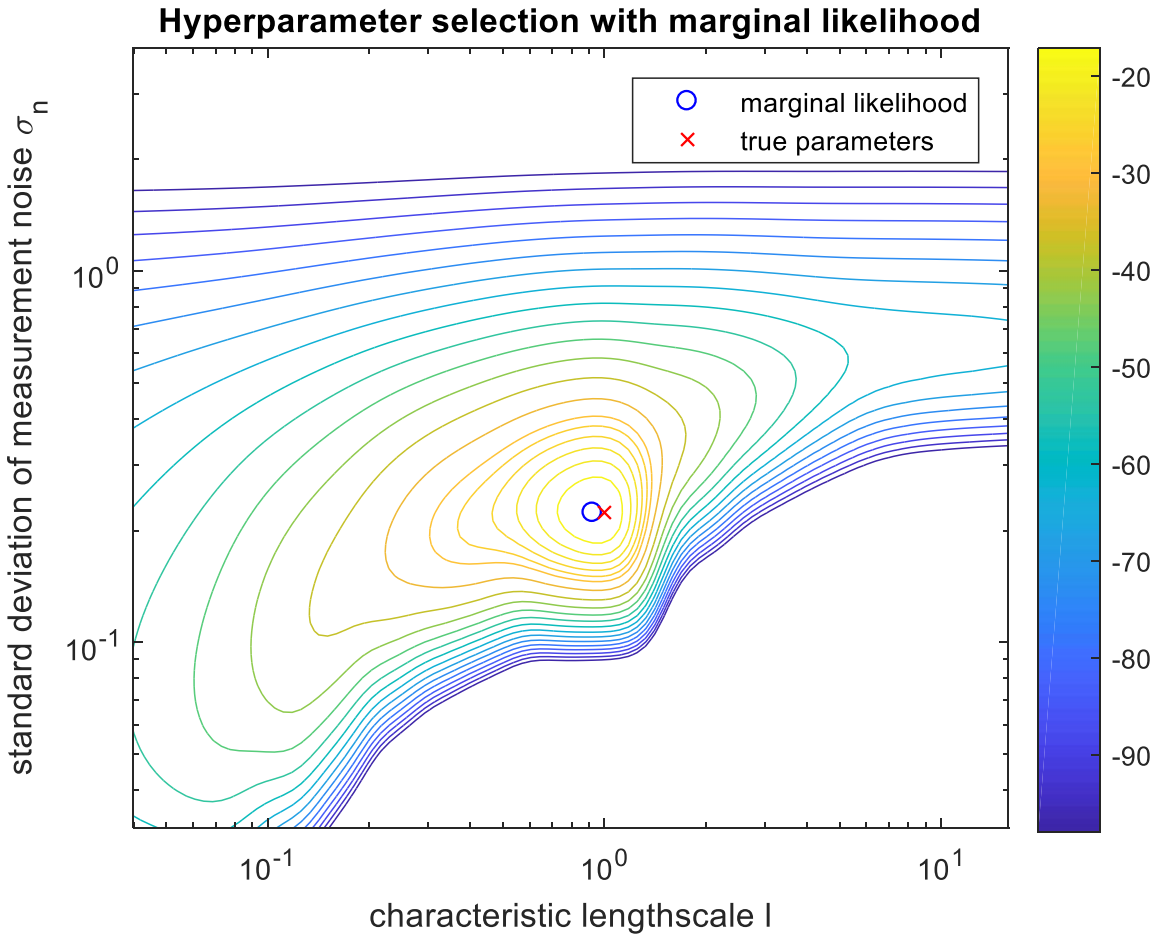
- 15 training datapoints





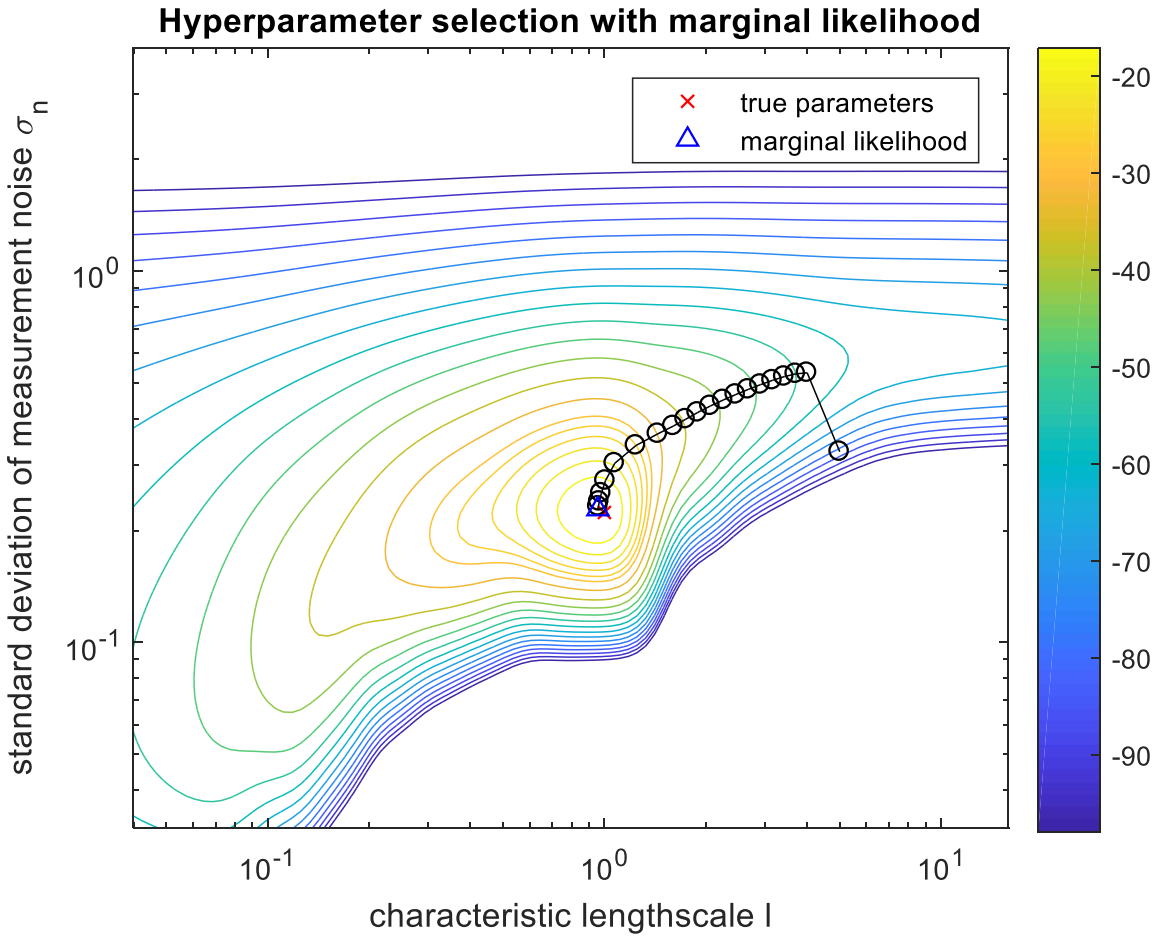
Bayesian model selection (training)

- 60 training datapoints



Bayesian model selection (training)

- Gradient descent (implementation trick: learn $\log(\cdot)$ of parameters)



Crossvalidation

- We break the training set into the new training set and the validation set
  - Or use k-fold crossvalidation
- Find the $\boldsymbol{\theta}$ and $\mathcal{H}_i$ which have the best performance on the validation set
- Metrics

$$L(\mathbf{X}_v, \mathbf{y}_v, \boldsymbol{\theta}, \mathcal{H}_i) = \frac{1}{n_v} \sum_{i=1}^{n_v} \left(y_{pred}^{(i)} - y_v^{(i)} \right)^2$$

$$L(\mathbf{X}_v, \mathbf{y}_v, \boldsymbol{\theta}, \mathcal{H}_i) = \frac{1}{n_v} \sum_{i=1}^{n_v} -\log p(y_v^{(i)} | \mathbf{x}_v^{(i)}, \mathbf{y}_t, \mathbf{X}_\mathrm{t}, \boldsymbol{\theta}, \mathcal{H}_i)$$

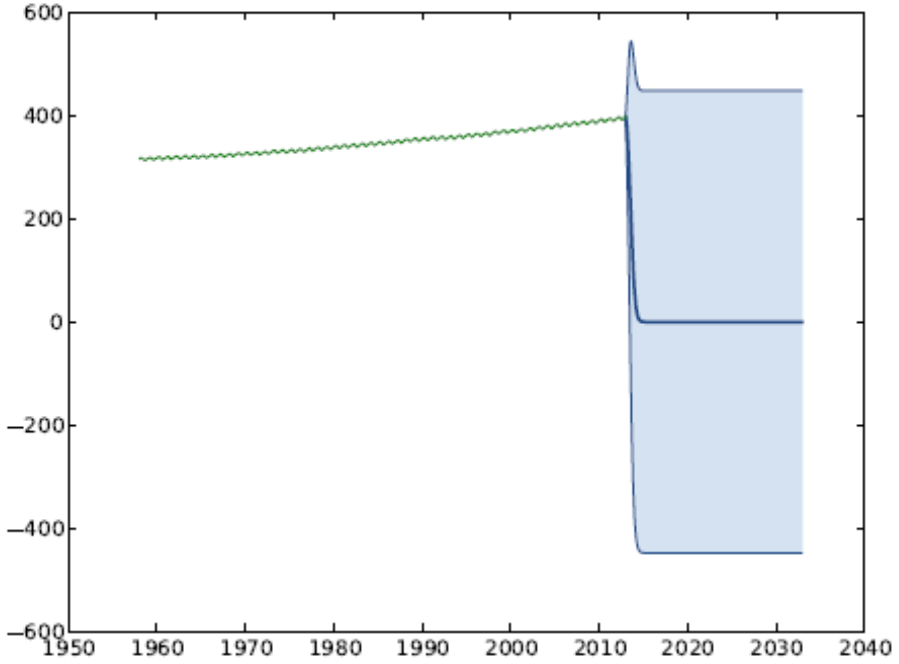
$$= \frac{1}{n_v} \sum_{i=1}^{n_v} \left(\frac{1}{2} \log \sigma_v^{(i)} + \frac{\left(\mu_v^{(i)} - y_v^{(i)} \right)^2}{2\sigma_v^{2(i)}} + \frac{1}{2} \log 2\pi \right)$$

Kernel design

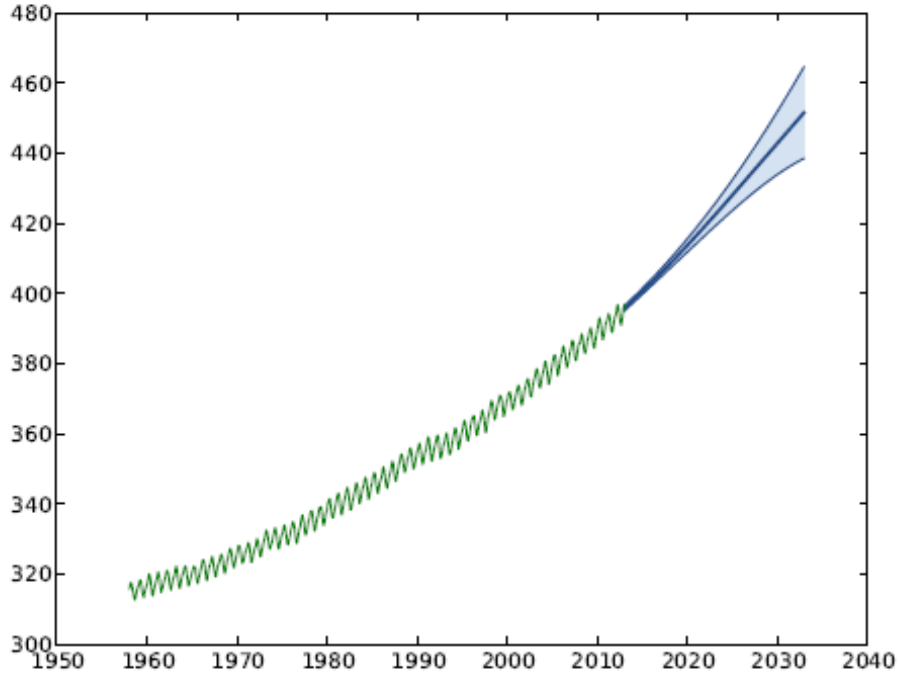
- The Mauna Loa observatory dataset – monthly CO_2 concentration in Hawai

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = k_{SE}(\boldsymbol{x}_p, \boldsymbol{x}_q)$$

small lengthscale



big lengthscale



Figures taken from http://gpss.cc/gpss18/slides/Durrande2018.pdf

Kernel design

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = k_{SE}(\boldsymbol{x}_p, \boldsymbol{x}_q) + k_{SE}(\boldsymbol{x}_p, \boldsymbol{x}_q) + k_{PER}(\boldsymbol{x}_p, \boldsymbol{x}_q) + k_{QUADRATIC}(\boldsymbol{x}_p, \boldsymbol{x}_q)$$

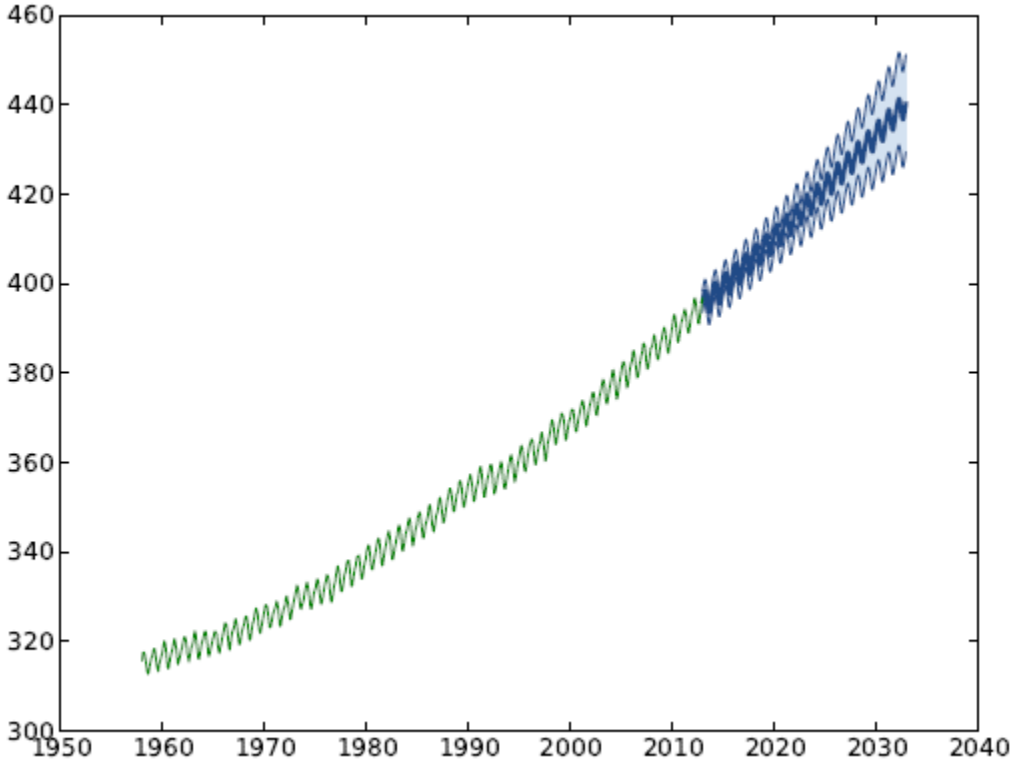
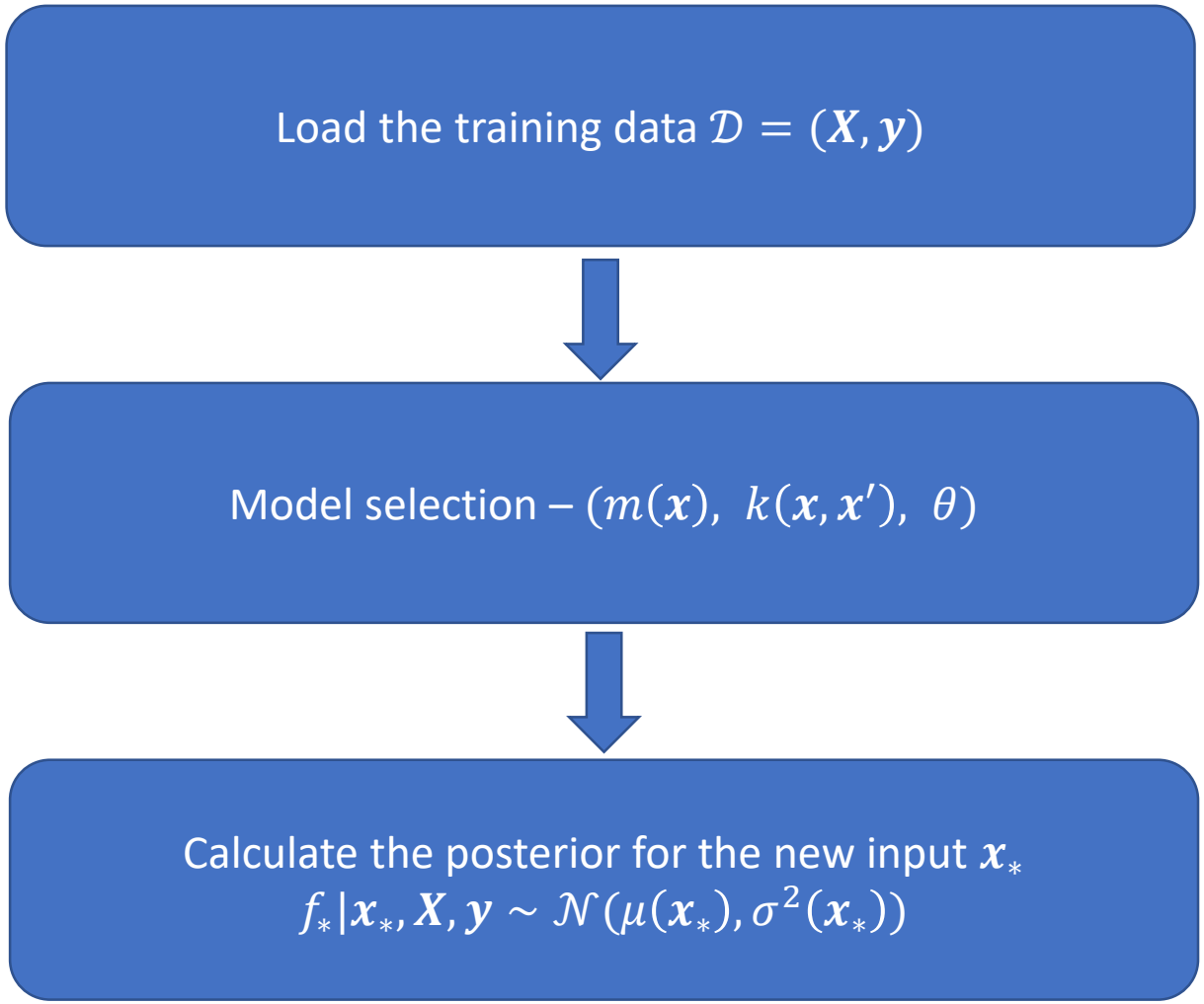


Figure taken from http://gpss.cc/gpss18/slides/Durrande2018.pdf

Gaussian Processes for Regression



What are some advantages of using Gaussian Process Models vs Neural Networks?

Answer · Follow · 130 · Request · 1

4 Answers

Yoshua Bengio, My lab has been one of the three that started the deep learning approach, back in 2006, along with Hinton's...

Answered Apr 6, 2011

I would add the following to David Warde-Farley's excellent answers. An advantage of Gaussian Processes is that, like other kernel methods, they can be optimized exactly, given the values of their hyper-parameters (such as the weight decay and the spread of a Gaussian kernel), and this often allows a fine and precise trade-off between fitting the data and smoothing. On small datasets they are very good because of this well-tuned smoothing and because they are still computationally affordable. They are my method of choice for small regression datasets (less than 1000 or 2000 examples). On the other hand, if you want to capture a complicated function (with many many ups and downs, i.e., not necessarily very smooth), then you need a model that can scale to large datasets and that can generalize non-locally (which kernel machines with standard generic kernels, typically local, do not provide). Modern variants of neural networks (so-called Deep Learning, Deep Learning) are more attractive with respect to these two properties, so I would prefer them for larger datasets where there is a lot of structure to be extracted from the data (the target function is not smooth).

49.7k views · View Upvoters

Sparse GPs

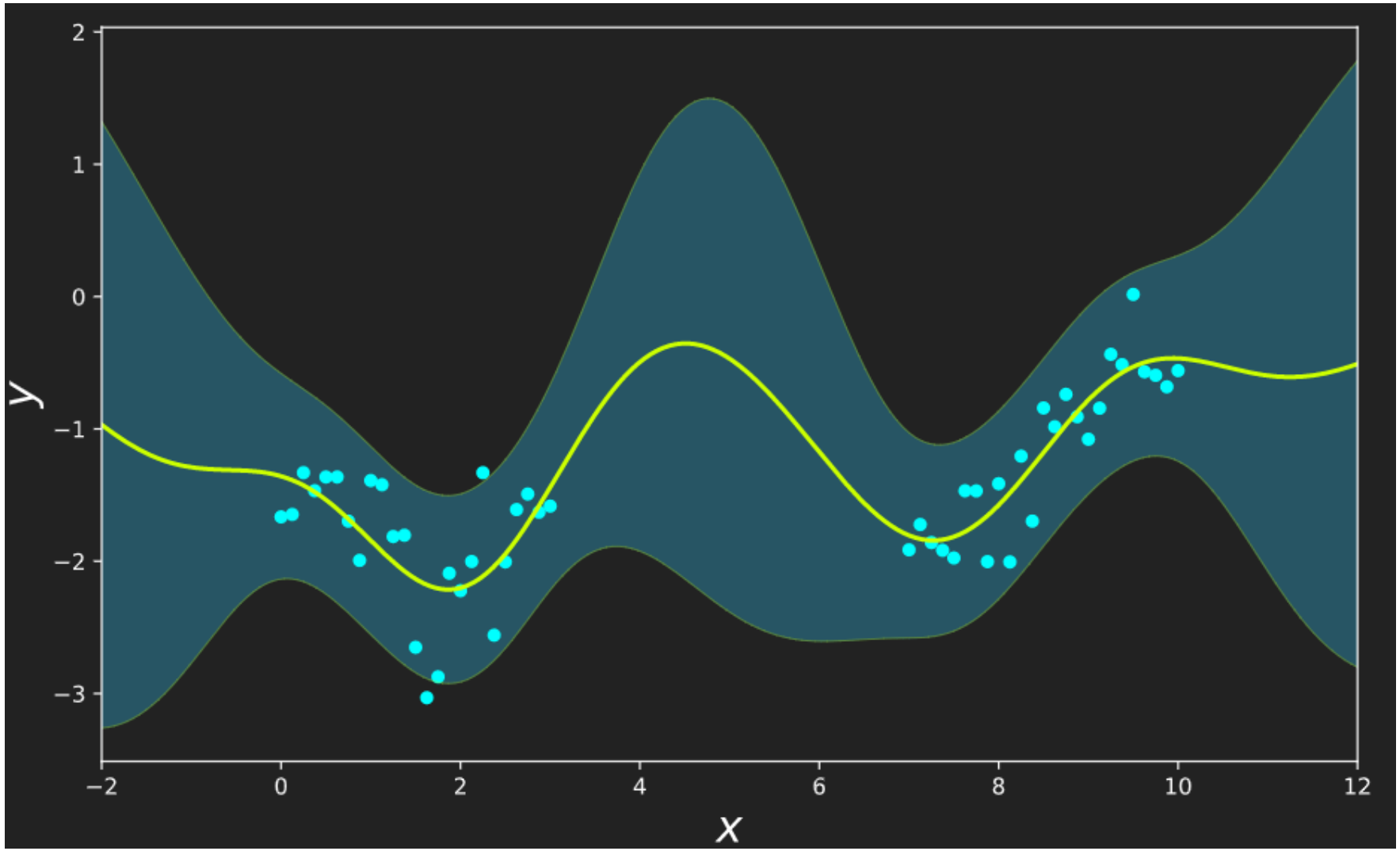


Figure taken from http://inverseprobability.com/talks/notes/deep-gaussian-processes.html

Sparse GPs

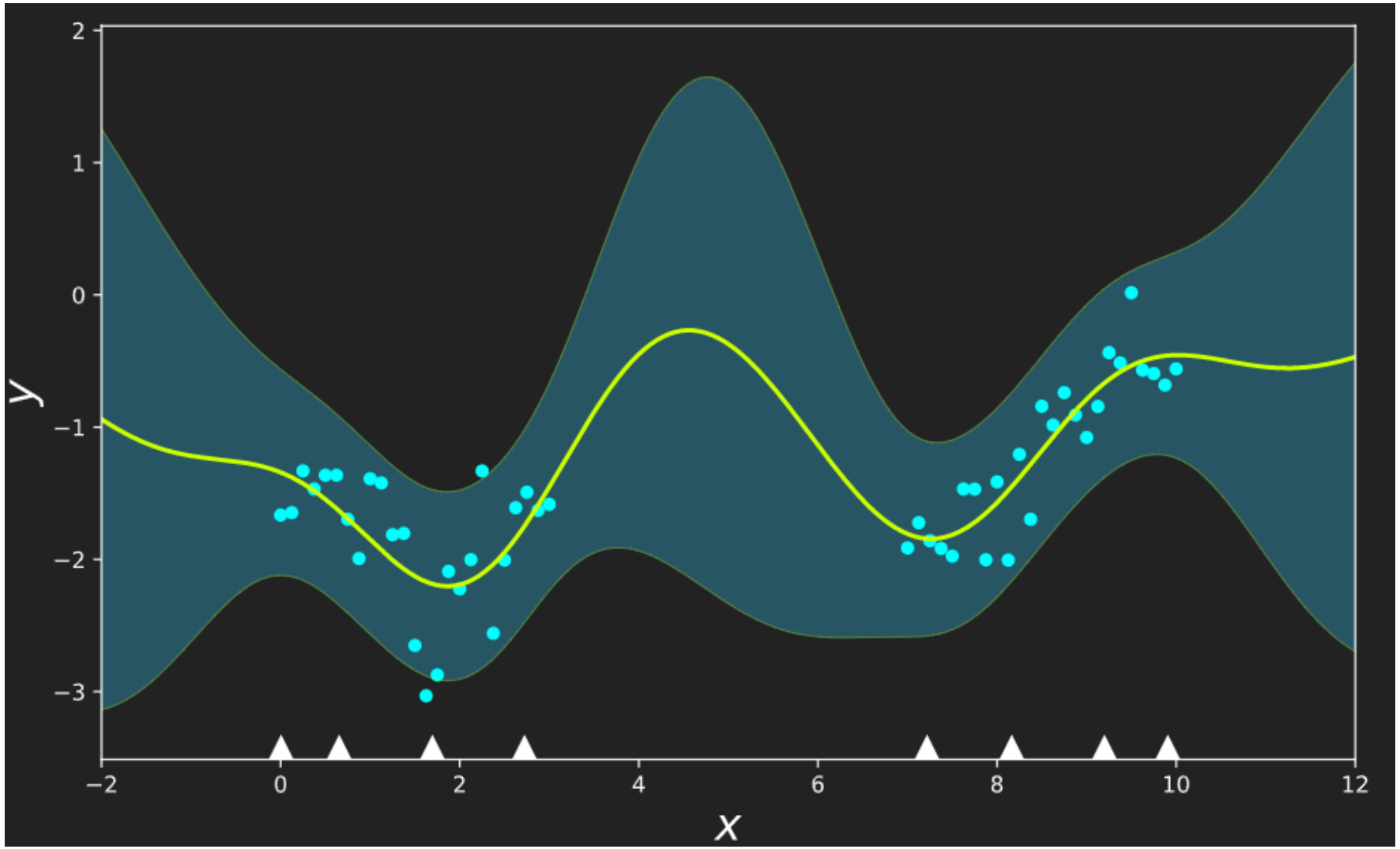
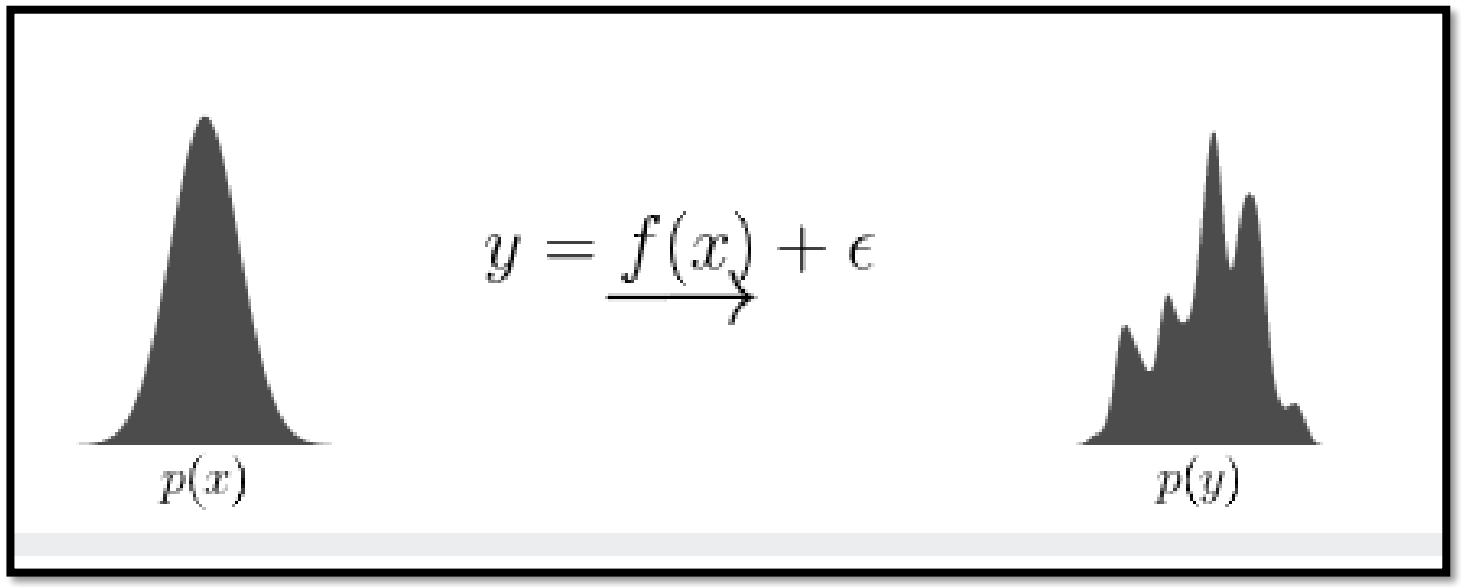


Figure taken from http://inverseprobability.com/talks/notes/deep-gaussian-processes.html

Deep GPs

- DL revolution: hierarchical learning
- Analytically intractable
- Variational approaches are used

$$p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{f}_5)p(\mathbf{f}_5|\mathbf{f}_4)p(\mathbf{f}_4|\mathbf{f}_3)p(\mathbf{f}_3|\mathbf{f}_2)p(\mathbf{f}_2|\mathbf{f}_1)p(\mathbf{f}_1|\mathbf{x})$$



Figures taken from http://inverseprobability.com/talks/notes/deep-gaussian-processes.html

Some applications

- Bayesian optimization
  - Tuning the Swiss Free Electron Laser
  - Tuning Reinforcement Learning Hyperparameters
- Crowd counting from an image
- Geostatistics
- Modelling of facial expressions
- Learning a model of a dynamic system
  - Applying model-based RL

Exact Gaussian Processes on a Million Data Points

Ke Alexander Wang[1*] **Geoff Pleiss**[1*] **Jacob R. Gardner**[2]
Stephen Tyree[3] **Kilian Q. Weinberger**[1] **Andrew Gordon Wilson**[1]
[1]Cornell University, [2]Uber AI Labs, [3]NVIDIA

Abstract

Gaussian processes (GPs) are flexible models with state-of-the-art performance on many impactful applications. However, computational constraints with standard inference procedures have limited exact GPs to problems with fewer than about ten thousand training points, necessitating approximations for larger datasets. In this paper, we develop a scalable approach for exact GPs that leverages multi-GPU parallelization and methods like linear conjugate gradients, accessing the kernel matrix only through matrix multiplication. By partitioning and distributing kernel matrix multiplies, we demonstrate that an exact GP can be trained on over a million points in 3 days using 8 GPUs and can compute predictive means and variances in under a second using 1 GPU at test time. Moreover, we perform the first-ever comparison of exact GPs against state-of-the-art scalable approximations on large-scale regression datasets with $10^4 - 10^6$ data points, showing dramatic performance improvements.

1 INTRODUCTION

Gaussian processes (GPs) have seen great success in many machine learning settings, such as black-box optimization (Snoek et al., 2012), reinforcement learning (Deisenroth and Rasmussen, 2011; Deisenroth et al., 2015), and time-series forecasting (Roberts et al., 2013). These models offer several advantages – principled uncertainty representations, model priors that require little with few observations; they also have great promise to exploit the available information in increasingly large datasets, especially when combined with expressive kernels (Wilson and Adams, 2013) or hierarchical structure (Wilson et al., 2012; Damianou and Lawrence, 2013; Wilson et al., 2016a; Salimbeni and Deisenroth, 2017).

In practice, however, exact GP inference can be intractable for large datasets, as it naïvely requires $\mathcal{O}(n^3)$ computations and $\mathcal{O}(n^2)$ storage for n training points (Rasmussen and Williams, 2006). Many approximate methods have been introduced to improve scalability, relying on mixture-of-experts models (Deisenroth and Ng, 2015), inducing points (Snelson and Ghahramani, 2006; Titsias, 2009; Wilson and Nickisch, 2015; Gardner et al., 2018b), random feature expansions (Rahimi and Recht, 2008; Le et al., 2013; Yang et al., 2015), or stochastic variational optimization (Hensman et al., 2013, 2015; Wilson et al., 2016b; Cheng and Boots, 2017; Salimbeni et al., 2018). However, choosing a suitable scalable approach involves many design choices, such as numbers of random features, types of features, and numbers and locations of inducing points. Performance is often sensitive to these choices and depends on the properties of a given dataset. Due to the historical intractablity of training exact GPs on large datasets, it is an open question how approximate methods compare to an exact approach when much more data is available.

In this paper, we develop a methodology to scale exact GP inference well beyond what has previously been achieved: we train an exact Gaussian process *on over a million data points without approximations*. Such a result would be intractable with standard implementations, which rely on the Cholesky decomposition. The scalability we demonstrate is made feasible through the

Where to begin?

- C. E. Rasmussen and C. K. I. Williams, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, 2005.
  - **Beginners guide to GP for regression (1., 2., 4. briefly, 5.)**

- http://gpss.cc/gpss18/program
  - Gaussian Process and Uncertainty Quantification Summer School (video lectures)

- https://deepbayes.ru/
  - DL and Bayesian methods summer school, should have video lectures

Where to begin?

- http://inverseprobability.com/talks/notes/deep-gaussian-processes.html
  - Neil Lawrence on sparse GPs and deep GPs
- https://www.prowler.io/blog/sparse-gps-approximate-the-posterior-not-the-model
  - Blog about the trends in sparse GPs (has highlighted some important papers)
- A. Damianou, N. Lawrence, Deep Gaussian Processes, Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, PMLR 31:207-215, 2013.

Thank you for your attention!